End-to-End QoS Network Design

Quality of Service for Rich-Media & Cloud Networks

Second Edition

Tim Szigeti
Christina Hattingh
Robert Barton
Kenneth R. Briley, Jr.

ciscopress.com

End-to-End QoS Network Design

Second Edition

Tim Szigeti, CCIE No. 9794
Robert Barton, CCIE No. 6660
Christina Hattingh
Kenneth Briley, Jr., CCIE No. 9754

Cisco Press

800 East 96th Street

Indianapolis, IN 46240

End-to-End QoS Network Design
Quality of Service for Rich-Media & Cloud Networks
Second Edition

Tim Szigeti, CCIE No. 9794
Robert Barton, CCIE No. 6660
Christina Hattingh
Kenneth Briley Jr., CCIE No. 9754

Published by:
Cisco Press
800 East 96th Street
Indianapolis, IN 46240 USA

Printed in the United States of America

First Printing November 2013

Library of Congress Control Number: 2013950000

ISBN-13: 978-1-58714-369-4

ISBN-10: 1-58714-369-0

Warning and Disclaimer

This book is designed to provide information about designing a network with end-to-end quality of service. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an “as is” basis. The authors, Cisco Press, and Cisco Systems, Inc., shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the author and are not necessarily those of Cisco Systems, Inc.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Cisco Press or Cisco Systems, Inc. cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Corporate and Government Sales

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests . For more information, please contact: **U.S. Corporate and Government Sales** 1-800-382-3419 corpsales@pearsontech-group.com

For sales outside of the U.S. please contact: **International Sales international@pearsoned.com**

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through e-mail at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Publisher: Paul Boger
Business Operation Manager, Cisco Press: Jan Cornelssen
Associate Publisher: Dave Dusthimer
Executive Editor: Brett Bartow
Senior Development Editor: Christopher Cleveland
Managing Editor: Sandra Schroeder
Copy Editor: Keith Cline
Project Editor: Seth Kerney
Technical Editors: John Johnston, Roland Saville
Editorial Assistant: Vanessa Evans
Proofreader: Jess DeGabriele
Cover Designer: Mark Shirar
Indexer: Christine Karpeles
Composition: Jake McFarland

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**

About the Authors

Tim Szigeti, CCIE No. 9794, is a senior technical leader in the Systems Design Unit at Cisco Systems, where his role is to design network architectures for enterprise mobility solutions. He has specialized in quality of service technologies for the past 15 years, during which time he has authored many technical papers, design guides, and two Cisco Press books: *End-to-End QoS Network Design* (version 1) and *Cisco TelePresence Fundamentals*.

Robert Barton, CCIE No. 6660, is located in Vancouver, where he lives with his wife and two children. He graduated from the University of British Columbia with a degree in engineering physics, and is a registered professional engineer. Rob holds dual CCIEs, in Routing and Switching and Security, and was also the first CCDE in Canada. Rob joined Cisco from ArrowPoint Communications, where he worked as a data center specialist supporting many of the largest corporations in Canada. In the time since ArrowPoint was acquired by Cisco, Rob has worked as a public sector systems engineer, primarily focused on wireless and security architectures. Currently, Rob is working on SmartGrid network technologies, including smart meter and intelligent substation design.

Christina Hattingh spent 13 years as a senior member of the technical staff in Unified Communications (UC) in the Enterprise Networking Routing Group (formerly Services Routing Technology Group or SRTG) at Cisco Systems. The SRTG products, including the Cisco 2900/3900 and 2800/3800 series ISR platforms and their predecessors, were the first Cisco platforms to converge voice, data, and video traffic and services on IP networks by offering TDM gateway interfaces, WAN interfaces, call control, and QoS features. The ISR series of routers often live at smaller remote offices and therefore at the edge of the WAN, where the need for QoS services is most sensitive. In this role, Christina spoke at Cisco Live conferences, trained Cisco sales staff and Cisco resale partners on router-based UC technologies, authored several Cisco Press books, and advised customers on UC network deployment and design, including QoS designs and helping them through the TDM to SIP trunk industry transition.

Kenneth Briley, Jr., CCIE No. 9754 is a technical lead in the Network Operating Systems Technology Group at Cisco Systems. For over 10 years, he has specialized in quality of service design and implementation in customer environments, alignment of QoS features and functions, and the marketing of new products that leverage QoS technologies. During this time, he has written several deployment guides and whitepapers, presented at Cisco Live, and most recently has focused on the convergence of wired and wireless quality of service.

About the Technical Reviewers

John Johnston, previously CCIE No. 5232, is a technical marketing engineer for Cisco Systems. His focus is on mobile security technology and design validation. John has more than 19 years of experience in IP internetworking, including the design and implementation of enterprise networks. Before joining Cisco Systems, John provided network design support for Fortune 500 companies. He holds a BSEE from the UNC-Charlotte.

Roland Saville is a Technical Leader for the Systems Development Unit (SDU) at Cisco, focused on developing best-practice design guides for enterprise network deployments. He has more than 18 years of experience at Cisco as a Systems Engineer, Product Manager, Consulting Systems Engineer, Technical Marketing Engineer, and Technical Leader. During that time, he has focused on a wide range of technology areas, including the integration of voice and video onto network infrastructures, network security, wireless LAN networking, RFID, energy management, Cisco TelePresence, and BYOD. He has also spent time focusing on the retail market segment. Prior to Cisco, he spent eight years as a Communications Analyst for Chevron Corporation. Roland holds a bachelor's of science degree in electrical engineering from the University of Idaho and an MBA degree from Santa Clara University. He co-authored the book *Cisco TelePresence Fundamentals*, is a member of the IEEE, and has 12 U.S. patents.

Dedications

Tim Szigeti:

I find myself in a dedication dilemma.

On the one hand, I already went to great lengths to explain why *not* dedicating the first edition of this book to my wife would have been a fatal mistake. Since then, I've gone on to dedicate my second book to my son, and now I have a beautiful daughter who deserves a dedication too (and whose arrival, incidentally, actually delayed the release of this edition by a couple of months).

So, the question becomes: Are dedications—as their definition implies—*exclusive* to a given book? Or can these be edition-specific? Or perhaps the more important question is: Do I really think it wise to get into a debate over semantics with my wife, who has a double-major in both English and philosophy?

So I'll play the political game and try to weakly rationalize a compromise: The first edition of this book was dedicated to Lella. The second will be to Lella 2.0, or as she's more commonly known, Isla.

Besides, I've already witnessed how much my daughter values my books. For example, over the past few months, she's had two copies of my previous book under her crib, slightly elevating one end to alleviate nighttime gas. Since she wakes up happy and smiling every morning, I'll infer from this her appreciation of the practical benefits of my work. Furthermore, she's always ready to gnaw and drool on my books until they're nice and soggy, and since pure happiness is expressed during the process, I'll attribute this to her esteem of the quality of the authorship.

And so, to my beautiful little girl, I wish to dedicate to you this work. I really don't know how I ever managed to finish it, seeing as how little you let me sleep over the past few months! I know you'll probably never read it, but that's not the point. I just want you to know you were always on my mind and made working on it virtually impossible! And I'm so very happy it's all done with now, so that I can spend more time playing with you and letting you continue wrapping me tightly around your tiny little finger!

Rob Barton:

This book is dedicated to my two wonderful boys, Adrian and Matthew. It's not that I expect you to actually pick up the book and try to become QoS experts, or that I am even trying to encourage you toward a career in network design or engineering, although these are noble pursuits. Rather, the lesson that writing this book has reminded me of is that you only grow as a person when you recognize the space you are in and make the decision to do something new. Oftentimes, we don't know what direction our efforts will take us in, but when you make the mindful choice to do something that is difficult, challenging, and can cause you more than a little pain along the way, you grow. No muscle ever grew without the fibers being damaged through exercise, and so is it too with all aspects of life. My hope is that this book will inspire you throughout your life to look for opportunities for growth—be it artistic, mental, professional, physical, or spiritual. This book is for you.

Christina Hattingh:

To Robert Verkroost and my parents for their unfailing encouragement and support.

Kenneth Briley, Jr.:

As this is my first book, I'd like to heed Tim's advice and dedicate it to my beautiful wife Mirah for fear of the aforementioned transgression. To Mirah, who incidentally read and approved this dedication, and her countless hours devoted to resolving numerous grammatical errors and listening to me drone on about how incredibly interesting QoS is. To our growing family; Lukas, Erik and Max: please don't grow up too fast, and remember that all things are possible.

Acknowledgments

Tim Szigeti:

First, I'd like to thank all the readers of the first edition who made it the success that it has become. There aren't many technology books that are still being steadily purchased nearly 10 years after their release. And a special thanks to the reviewers who have posted comments for it; I cannot express the pride and appreciation I felt when I saw five-star ratings for our book on Amazon. Thank you!

Thanks to my director, Neil Anderson, for long recognizing the critical role of QoS across all our networking systems and solutions and ensuring that it was always properly addressed. Thanks, too, to Greg Edwards in helping to define and articulate various end-to-end QoS strategies.

Thank you Fred Baker for your guidance and direction in both defining and interpreting various QoS standards. Thanks, too, to James Polk for continuing to push the envelope in the IETF to define what tomorrow's QoS models are going to look like.

I'd like to thank the Vancouver Cisco office lab administrator, Mojan Mobasser, for all her diligence in sourcing and arranging access to equipment. Similar thanks are extended to Dawid Brink for letting me use his Nexus boxes—no questions asked!

Farther east, I'd also like to extend thanks to the Toronto Bell Canada team for allowing me extended access to their ASR and CRS labs. Similar thanks, but in the opposite geographic direction, go out to Lim Fung in our Singapore office for providing me access to his labs also.

I'd like to extend sincere thanks to Tim Stevenson for his amazing technical expertise, particularly relating to data center platforms. You really helped demystify a lot of hardware architectural questions I was grappling with. Thanks, Tim!

Also I'd like to thank Lukas Krattiger in Switzerland for hours of research, testing, and correspondence to ensure that we properly wrapped our arms around Nexus 7000 QoS. Thanks for all your insight, patience, and hard work, Lukas!

Additionally, I'd like to thank Lucien Avramov for sharing his work on data center QoS and allowing me to borrow from it. Thank you too, Mike Herbert—wherever you are—for getting the ball rolling in this area.

I'd like to thank also the Cisco product teams that listened to the feedback we offered as a result of producing this book so as to continue to improve our QoS feature sets. This includes Albert Mitchell and the Catalyst 2K/3K team for implementing our latest designs into a new version of AutoQoS. Thanks also to Sarath Subrahmanya and Ramachandra Murthy in India for taking to heart our suggestions on WLC QoS feature enhancements. Kudos also go out to Oz Ben-Rephael and team in Israel for continuing to develop NBAR2 application signatures, including for our own Jabber application.

Thanks to the Cisco Press team. Brett Bartow: Thanks for taking on this project and allowing us to thoroughly update and expand on our original work in a comprehensive manner. We appreciate that you didn't blow a gasket when we exceeded our targeted

page count again, again, and again—to a final tune of target +50%! Thanks also for delaying this publication by a couple of months, letting me focus on my family as my daughter was born.

Thank you Chris Cleveland for making the review process so easy. Your comments and accommodation were very much appreciated and really helped polish this work. Thank you, too, Seth Kerney for coordinating the copy review. And also thanks to Vanessa Evans for ensuring that we always had everything we needed at every step of the way.

I'd like to extend exceptional thanks to our technical editors Roland Saville and John Johnston. Roland: You're one of the smartest persons I've had the pleasure of working with—and in so many fields. I don't know how your brain doesn't explode! You know I like to think of you as a "philosopher engineer," because you can take almost any design recommendation and find the corner-case counterargument where it breaks downs and doesn't apply. That's critically important to the process because by seeing from a distance where things can break you continually save us tremendous amounts of time in the lab, as well as ensuring the overall quality of the final designs. Thank you, too, JJ! You allowed me unfettered access to your massive labs in RTP and helped me along every step of the way. Your attention to detail is so impressive that I'm nearly spooked by your ability to catch the tiniest errors while reviewing hundreds of pages of configurations!

Finally, I owe a great deal of gratitude to my co-authors:

Ken: Thanks for your impressive knowledge and flexibility that you demonstrated by being able to jump right in and seamlessly adapt your research to our work in such an intuitive and cohesive manner. I've enjoyed working with you on many projects for the past decade and look forward to many more collaborations. Thanks again, Ken!

Christina: Thanks so much for coming out of retirement to work on one more project. Even though you're on the road more than Jack Kerouac these days, it was a real pleasure working with you again! Thanks for donning your QoS hat for us once again and bringing all your knowledge and experience to the table to help make this such a solid work.

Rob: Over the past 20 years we've been friends, classmates, roommates, workmates, "second-best" men at each other's weddings, and now co-authors. Your courage and determination are very inspiring. I honestly don't know if I would have taken my CCIE if I hadn't watched you do it. Same goes with running half-marathons (and one-day marathons!) Thanks for all your tremendous work on this project. It certainly was not for the faint-hearted, as every time we turned around we seemed to uncover yet another rabbit hole of technical issues that required yet more research and testing to be done. Thanks for sticking with it and seeing it through, Rob. But then again, that's just the kind of friend you are.

Rob Barton:

To begin, I would like to thank my very forgiving colleagues in the Cisco Vancouver office who have suffered through two years of trying to depend on an attention divided systems engineer who was more interested in solving theoretical QoS problems than in helping his customers. Special thanks to my Cisco account team partner, Mike MacDonald, for his long-suffering patience, my manager, Ralph Wright, who enthusiastically supported this effort and always offered many words of encouragement, and to my director, Bill Kastelic, who eagerly gave me the flexibility to do this project. None of this would have been possible without the support from you guys.

I would also like to thank my lab administrator, Mojan Mobasser, for helping to get lab gear when I needed it the most. Testing these QoS solutions involved a lot of lab time, and without your support we would not have been able to build and test these solutions.

Special thanks goes out to Ian Procyk and my co-author Ken Briley who helped test some of the more difficult wireless scenarios. As well, I would like to thank Larry Ross for the many hours of emails and phone conversations discussing various wireless QoS solutions with me. Also thanks goes out to Kangwarn Chinthammit for helping with the AVC section review, and Scott Wainer who helped with the GET VPN work. All you guys were like my technical conscious during this project.

I'd also like to thank Bruno Wollmann from Terra-Comm Solutions who, while discussing my presentation at Cisco Live last year, introduced me to the concept of combining DMVPN with GET VPN to solve a real-world performance issue related to VoIP, which I think has made a great addition to the GET VPN chapter.

Chris Cleveland and Brett Bartow, thanks so much for your hard work on this project and supporting us all the way through. This project turned into a much bigger undertaking than any of us had expected, and instead of trying to apply your own QoS mechanism on our output, you let the creative juices flow, and in the end helped support a substantial work of technical literature.

Lastly, I'd like to thank Tim Szigeti. Not only have you been one of my closest friends for more than 20 years, you are also an inspiring engineer. Yes, I said engineer, the word you always tease me with. I can clearly remember the day this project started two and a half years ago; we were rewarding ourselves with a well-earned breakfast at the White Spot after one of our half-marathon training runs. I was complaining that your first edition of the End-to-End QoS book, while being a great book, was hopelessly out of date. Your response to me was unforgettable: "So why don't you help me write a new one?" That day was the start of this project, and although it was a long and difficult undertaking, it has also been an immensely rewarding experience. Thanks, Tim!

Kenneth Briley, Jr.:

First off I'd like to thank Roland Saville, for his guidance and clever insight when we worked through QoS on the Converged Access platforms.

To Stephen Orr, wireless is now awesome, before it was an illusion – thanks for the brilliant and oh so colorful commentary.

Many thanks to Tripti Agarwal, Saravanan Radhakrishnan, Anuj Kumar, and Bhavana Kudipudi without that team we would have never been able to deliver such a versatile platform.

Contents at a Glance

Part I: QoS Design Overview

Part II: QoS Design Strategies

Part III: Campus QoS Design

Part IV: Wireless LAN QoS Design

Part V: Data Center QoS Design

Part VI: WAN and Branch QoS Design

Part VII: MPLS VPN QoS Design

Part VIII: IPsec QoS Design

Part XI: Appendixes (Online)

Contents

Part II: QoS Design Strategies

Part III: Campus QoS Design

Part IV: Wireless LAN QoS Design

Part V: Data Center QoS Design

Part VI: WAN and Branch QoS Design

Part VII: MPLS VPN QoS Design

Part VIII: IPsec QoS Design

Introduction

"Aren't we done with QoS yet?"

That's a question I get from time-to-time, which I like to answer along the lines of "As soon as we're done with availability and security, we'll be done with QoS also."

What I'm trying to express—although cheekily—is that although QoS has been around for a while, it is a foundational network infrastructure technology (the same as high-availability technologies and security technologies). And these foundational technologies will always prove to be integral components of any networking system, being present at the platform level, at the place in-the network (PIN) level and ultimately at the end-to-end network level.

Furthermore, such foundational network technologies are constantly evolving and expanding to meet new business and technical requirements. Such has been the case with QoS since the first edition of this work was published nearly 10 years ago.

For example, consider just one QoS-dependent application: video.

In 2004, there were really only two flavors of video traversing most enterprise networks: streaming video (unidirectional flows that benefited from both network- and application-level buffering to offset variations in transmission delays) and video conferencing (bidirectional 384-Kbps or 768-Kbps streams between dedicated hardware-based systems). So, we went into our massive Cisco Validation Labs in Research Triangle Park in North Carolina and hammered out best-practice designs to accommodate these two categories of video. We were done, right?

Wrong.

In the years that followed, codec and hardware advances made video production more cost-efficient and accessible, such that today nearly everyone with a smartphone has the ability to shoot high-definition video anytime and anywhere. Similarly, with the advent of social networking websites, video sharing and distribution suddenly became possible by anyone, anywhere (and that on a global scale!). Finally, video consumption also became possible anytime, anywhere, and on any device—thanks to advances in hardware and in wireless networking technologies.

That being the case, video is now the most dominant type of network traffic on the Internet and is expected to reach 90 percent within in a few years. Furthermore, there are many new forms and variations of video traffic, such as TelePresence, IP video surveillance, desktop video, and digital signage (just to name a few). And each of these types of video has unique service level requirements that must be met to ensure a high quality of experience by the end user. And thus, we circle back to QoS, which represents the enabling technologies to provide this quality of experience.

And that's just one application.

Advances in areas of data center and cloud networking, in addition to wireless networking, all have had corresponding impacts on QoS network designs.

Hence, a new edition of this book.

Another reason behind this second edition is to reflect the evolution of industry standards relating to QoS. Cisco has long advocated following industry standards and recommendations whenever deploying QoS, because this simplifies QoS designs, extends QoS policies beyond an administrative domain, and improves QoS policy effectiveness between administrative domains. Therefore, new standards, RFCs, and proposals have had—and will continue to have—a major impact on current and future strategic QoS designs.

A third key reason behind this new edition is that every network platform detailed in the original book has been replaced or significantly upgraded. So, the latest platforms (at the time of this writing) have been featured in this second version, with over a dozen Cisco product families being represented. In fact, nearly every design chapter features a different Cisco platform that suits the role being discussed, whether the role is a data center virtual switch, a branch router, a wireless LAN controller, a campus distribution switch, a WAN aggregator, a service provider core router, or so on.

And finally, QoS is a comprehensive and complex subject, one that entails a significant amount of fundamental technological concepts as well as platform-specific implementation detail. Therefore, it is often valuable for network administrators to have a single common reference on the subject, such as this book, which overviews all the relevant tools, presents various end-to-end strategies, and details platform-specific design recommendations for every major shipping Cisco platform.

And no, we're not done with QoS yet!

Objectives of This Book

The main objective of this book is to present—in a comprehensive and cohesive manner—the many aspects of quality of service design, including an overview of the tools, strategic and tactical design recommendations, and platform-specific configuration details. Therefore, novice to advanced network administrators alike can benefit from this volume as a single handy reference on this topic.

In addition, this exercise has produced multiple platform-specific configurations that can be viewed as QoS templates. As such, these templates can be considered roughly 80 percent of a generic enterprise or service provider QoS solution (borrowing from Pareto's 80/20 rule), to which another 20 percent of customizing and tailoring can be done to reach a final customer-specific solution. Considerations and rationales behind the presented designs are all explained so that administrators are fully informed of the rationale behind the designs and therefore can confidently modify these to meet their own specific requirements and constraints.

A key approach that we've used throughout this configuration-rich book is to incorporate inline explanations of configurations. In this way, QoS-relevant commands are highlighted and detailed line-by-line to explicate the function of each element and clarify how these parts make up the solution as a whole.

To complement these line-by-line design recommendations, related verification commands are also incorporated. These verification commands are presented in context with the design examples, with specific details of what-to-look-for being highlighted and explained. These verification examples are therefore significantly richer in relevance than most such examples presented in hardware/software documentation, and they allow network administrators to confirm quickly whether the recommended designs have been deployed correctly.

Finally, each design section has a case study chapter at the end that ties together many of the strategic principles, tactical recommendations, and platform-specific considerations that have been presented within the section. These case studies illustrate how to take generic and abstract design concepts and mold them to meet specific customer requirements. These case studies are indicative of what can be expected in real-life production environments. Each of these case study examples spans multiple devices, thus highlighting critical interrelationships. Furthermore, all case study chapters form respective parts of a single integrated end-to-end QoS network design.

Who Should Read This Book?

The primary reader of this book is the network administrator tasked with deploying QoS technologies. By extension, this group may also include other related IT professionals, such as systems administrators, audio/video specialists, VoIP specialists, and operations staff.

In addition, some readers may include technical decision makers tasked with evaluating the strategy and feasibility of QoS deployments, in addition to the drafting of implementation plans and phases toward these goals.

Yet another group of readers includes system engineers, partners, trainers, and other networking professionals who need to ramp-up technically on QoS technologies and designs, both for practical deployment purposes and to achieve various Cisco certifications.

Prerequisites are minimal, as the opening section of this book covers QoS technologies in high-to-mid-level technical detail, including protocols, tools, and relevant standards. In addition, each chapter includes extensive references for Additional Reading for more detailed information for readers unfamiliar with specific concepts discussed.

Because the content of the book ranges from a high level to a very low level of technical detail, it is suitable for a wide range of audiences, from intermediate to expert.

How This Book Is Organized

This book is organized into 39 chapters distributed across 8 parts, and includes 2 appendixes. Although this book can be read cover to cover, this organization allows readers to easily identify chapters of direct interest, thus facilitating the use of this book as a handy reference work. The eight parts of this book are described below:

Part I, "QoS Design Overview," introduces readers to QoS technologies, presenting a brief history and an architectural framework for these tools. Following this, groups of QoS tools are overviewed, including classification and marking tools, policing and shaping tools, queuing and dropping tools, bandwidth-reservation tools, and advanced tools like Medianet and application visibility and control.

Part II, "QoS Design Strategies," breaks away from a purely technical discussion to take a higher-level view of how business requirements drive QoS design. Application service-level requirements are analyzed, as are strategic QoS design best practices. This section concludes with the first case study chapter, illustrating the considerations that factor into defining an end-to-end QoS design strategy.

Part III, "Campus QoS Design," begins the exercise applying strategic QoS models to a tactical place in the network (PIN), which in this case is the enterprise campus. Campus-specific design considerations and recommendations are discussed at length, and subsequent chapters specialize in design recommendations for the access, distribution, and core layers of the campus network. A campus QoS design case study chapter completes the section.

Part IV, "Wireless LAN QoS Design," applies the strategic QoS models to the enterprise wireless LAN. Because WiFi is a unique media, as compared to the rest of the network, additional concepts need to be covered to explain how QoS can be achieved over-the-air. These considerations include the introduction of the Enhanced Distributed Coordination Function as well as IEEE 802.11e/Wireless Multimedia QoS. Following this, QoS design chapters address both the centralized wireless LAN controller deployment model and the new wired-and-wireless converged access deployment model. The section finishes with a WLAN QoS design case study.

Part V, "Data Center QoS Design," continues the application of QoS strategies, but this time to the data center network. Because of the convergence of storage-area networks and local-area networks within the data center, certain protocols require a completely lossless service that traditional QoS tools cannot guarantee. Therefore, data center-specific QoS tools are discussed, including the data center bridging toolset, which can be leveraged to guarantee such a lossless service. Following this, QoS design chapters address the virtual access layer, access and aggregation layers, and the core layer of data center networks. This part closes with a data center QoS design case study.

Part VI, "WAN and Branch QoS Design," expands the scope of discussion beyond the local area and applies strategic QoS principles to the wide-area network. QoS designs are presented for both WAN aggregation routers and for branch routers. This part ends with a WAN QoS design case study.

Part VII, "MPLS VPN QoS Design," continues the wide-area discussion but addresses QoS strategies for MPLS VPN networks, taking the perspectives of both the enterprise customer and the service provider into account in the end-to-end design. Design chapters are presented for the enterprise customer-edge router, the provider-edge router and the provider core routers. This section finishes with a case study on MPLS VPN QoS design.

Part VIII, "IPsec QoS Design," concludes the discussion by applying strategic QoS principles to IPsec VPNs. QoS designs are detailed for both Dynamic Multipoint VPNs and Group Encrypted Transport VPNs.

An overview on each of the 39 chapters (and the 2 appendixes) follows.

- **Chapter 1, "Introduction and Brief History of QoS and QoE":** Provides a brief history lesson on quality of service and quality of experience evolution, introducing fundamental QoS concepts, standards, and the evolutionary changes necessitating a second edition of this book.
- **Chapter 2, "IOS-Based QoS Architectural Framework and Syntax Structure":** Overviews how QoS tools interrelate, and introduces Cisco's IOS-based Modular QoS command-line interface (MQC), the common syntax structure for configuring QoS across most Cisco platforms.
- **Chapter 3, "Classification and Marking Tools":** Describes the various classification options for distinguishing one packet from another, which is the requisite first step in providing differentiated services. Also discussed are various marking options so that packets do not have to be reclassified at every network node.
- **Chapter 4, "Policing, Shaping, and Markdown Tools":** Discusses various tools that can be used to meter and regulate packet flows, including policers (which drop excess traffic), shapers (which delay excess traffic) and markers (which re-mark excess traffic).
- **Chapter 5, "Congestion Management and Avoidance Tools":** Considers options on how to deal with bottlenecks in the network, by addressing both queuing tools (to determine which packets get priority or preferential treatment during congestion), and early-dropping tools (to reduce the probability of congestion).
- **Chapter 6, "Bandwidth-Reservation Tools":** Introduces the concepts of bandwidth reservations and endpoint/infrastructure signaling to communicate how and when such reservations are to be made.
- **Chapter 7, "QoS in IPv6 Networks":** Examines IPv6 packet formats, classification and marking options, and how QoS tools are to be configured in IPv6 networks or in mixed IPv4 and IPv6 networks.
- **Chapter 8, "Medianet":** Gives a brief overview of the Medianet architecture, with particular focus on the aspects of Medianet specific to QoS configuration and monitoring.
- **Chapter 9, "Application Visibility and Control":** Presents deep packet inspection technologies for application identification, classification, and monitoring and how these can be used within the network.
- **Chapter 10, "Business and Application QoS Requirements":** Examines current business trends impacting QoS designs and various application-class QoS requirements.

- **Chapter 11, "QoS Design Principles and Strategies":** Combines the QoS tools and business requirements presented in preceding chapters and formulates these into QoS strategic models to address basic, intermediate, and advanced requirements.
- **Chapter 12, "Strategic QoS Design Case Study":** This first case study in the series introduces a fictional company, Tifosi Software, and discusses the business and technical considerations that come into play when defining an end-to-end QoS strategy.
- **Chapter 13, "Campus QoS Design Considerations and Recommendations":** Overviews various considerations and recommendations relating to campus QoS design, including trust boundaries, per-port versus per-VLAN design options, and EtherChannel QoS considerations.
- **Chapter 14, "Campus Access (Cisco Catalyst 3750) QoS Design":** This first platform-specific design chapter details best practice QoS designs at a configuration level for Cisco Catalyst 3750 series switches in the role of a campus access layer edge switch.
- **Chapter 15, "Campus Distribution (Cisco Catalyst 4500) QoS Design":** This design chapter details configuration recommendations for a Cisco Catalyst 4500 series switch in the role of a campus distribution layer switch. Additional designs include details on how this switch can be configured as a campus access-edge switch also.
- **Chapter 16, "Campus Core (Cisco Catalyst 6500) QoS Design":** This design chapter details configuration recommendations for a Cisco Catalyst 6500 series switch in the role of a campus core layer switch. Additional designs include details on how this switch can be configured as a campus access-edge or distribution layer switch as well.
- **Chapter 17, "Campus QoS Design Case Study":** This case study chapter describes how Tifosi Software has applied their strategic QoS design model to their campus network consisting of Cisco Catalyst 3750, 4500 and 6500 series switches.
- **Chapter 18, "Wireless LAN QoS Considerations and Recommendations":** Overviews various considerations and recommendations relating to wireless LAN QoS design and introduces WLAN QoS tools such as the Enhanced Distributed Coordination Function and Wireless Multimedia QoS.
- **Chapter 19, "Centralized (Cisco 5500 Wireless LAN Controller) QoS Design":** This design chapter details both GUI and CLI configuration recommendations for centralized wireless LAN controller (WLC) deployment models, featuring the Cisco 5500 WLC.
- **Chapter 20, "Converged Access (Cisco Catalyst 3850 and the Cisco 5760 Wireless LAN Controller QoS Design":** This design chapter details configuration recommendations for converged access WLAN deployment models, featuring the Cisco Catalyst 3850 series switch and the Cisco 5760 WLC.

- **Chapter 21, "Converged Access QoS Design Case Study":** This case study chapter describes how Tifosi Software has applied their strategic QoS design model to their wired-and-wireless converged access LAN network consisting of Cisco Catalyst 3850 series switches and the Cisco 5760 WLC.
- **Chapter 22, "Data Center QoS Design Considerations and Recommendations":** Overviews various considerations and recommendations relating to data center QoS design and introduces the data center bridging toolset.
- **Chapter 23, "Data Center Virtual Access (Nexus 1000V) QoS Design":** This design chapter details configuration recommendations for a Cisco Nexus 1000V series virtual switch in the role of a data center access layer switch.
- **Chapter 24, "Data Center Access/Aggregation (Nexus 5500/2000) QoS Design":** This design chapter details configuration recommendations for a Cisco Nexus 5500 series switch, which may include Cisco Nexus 2000 series Fabric Extenders, in the role of a data center access/aggregation switch.
- **Chapter 25, "Data Center Core (Nexus 7000) QoS Design":** This design chapter details configuration recommendations for a Cisco Nexus 7000 series switch in the role of a data center core switch. QoS designs for both M-Series and F-Series modules are detailed.
- **Chapter 26, "Data Center QoS Design Case Study":** This case study chapter describes how Tifosi Software has applied their strategic QoS design model to their data center network, consisting of Cisco Nexus 1000V, 5500/2000 and 7000 series switches.
- **Chapter 27, "WAN and Branch QoS Design Considerations and Recommendations":** Overviews various considerations and recommendations relating to WAN QoS design, including hardware versus software considerations, latency and jitter targets, and bandwidth-reservation options.
- **Chapter 28, "WAN Aggregator (Cisco ASR 1000) QoS Design":** This design chapter details configuration recommendations for a Cisco ASR 1000 series router in the role of a WAN aggregation router. WAN media featured includes leased lines, ATM, and Packet-Over-SONET.
- **Chapter 29, "Branch Router (Cisco ISR G2) QoS Design":** This design chapter details configuration recommendations for a Cisco ISR G2 series router in the role of a branch router, featuring Medianet and AVC designs.
- **Chapter 30, "WAN and Branch QoS Design Case Study":** This case study chapter describes how Tifosi Software has applied their strategic QoS design model to their wide-area network, consisting of Cisco ASR 1000 and ISR G2 series routers.
- **Chapter 31, "MPLS VPN QoS Design Considerations and Recommendations":** Overviews various considerations and recommendations relating to MPLS VPN QoS design, both from an enterprise and from a service provider perspective, including enterprise-to-provider mapping models and MPLS DiffServ tunneling modes. In

addition, this design section features carrier Ethernet as a WAN media.

- **Chapter 32, "Enterprise Customer Edge (Cisco ASR 1000 and ISR G2) QoS Design":** This design chapter details configuration recommendations for a Cisco ASR 1000 or ISR G2 series router in the role of an enterprise customer-edge router interfacing with a MPLS VPN service provider.
- **Chapter 33, "Service Provider Edge (Cisco ASR 9000) QoS Design":** This design chapter details configuration recommendations for a Cisco ASR 9000 series router in the role of a service provider edge router.
- **Chapter 34, "Service Provider Core (Cisco CRS) QoS Design":** This design chapter details configuration recommendations for a Cisco CRS-3 series router in the role of a service provider core router.
- **Chapter 35, "MPLS VPN QoS Design Case Study":** This case study chapter describes how Tifosi Software has adapted their strategic eight-class enterprise QoS model to integrate with their service provider's six class-of-service model, featuring Cisco ISR G2, ASR 1000, ASR 9000, and CRS-3 series routers.
- **Chapter 36, "IPsec VPN QoS Considerations and Recommendations":** Overviews various considerations and recommendations relating to IPsec VPN QoS design, including classification of encrypted packets, MTU considerations, and anti-replay implications.
- **Chapter 37, "DMVPN QoS Design":** This design chapter details configuration recommendations for Cisco ASR 1000 and ISR G2 routers in the roles of DMVPN hub-and-spoke routers (respectively).
- **Chapter 38, "GET VPN QoS Design":** This design chapter details configuration recommendations for Cisco ISR G2 routers in the roles of GET VPN routers.
- **Chapter 39, "Home Office VPN QoS Case Study":** This case study chapter describes how Tifosi Software has adapted their strategic QoS model over a DMVPN to provide telecommuting services to employees in their home offices. This case study features Cisco ASR 1002 series routers at the headend and ISR 881 series routers connected behind a broadband modem via Ethernet at the home office.
- **Appendix A, "AutoQoS for Medianet":** This online appendix overviews the latest evolution of the AutoQoS feature, which is based on the same QoS designs presented in this book. Detailed syntax is presented for the first platforms to support this feature, including the Cisco Catalyst 3750 and 4500 series switches.
- **Appendix B, "Control Plane Policing":** This online appendix overviews the control plane policing feature, which applies a QoS function (of policing) to a virtual interface (the control plane) to harden the network infrastructure from denial-of-service or worm attacks. Best-practice recommendations and configurations are presented for this feature.

This page intentionally left blank

Chapter 1

Introduction and Brief History of QoS and QoE

This chapter provides a brief history of rich media and cloud network evolution, illustrating the background of the concepts of quality of service (QoS) and quality of experience (QoE). It introduces the fundamental QoS/QoE concepts, standards, and the recent networking changes giving rise to the second edition of this book. The chapter specifically covers the following topics:

- Evolution of traffic types and classes
- QoS and QoE
- Circuit-switched and packet-switched network technologies
- Integrated services (IntServ) and differentiated services (DiffServ)

QoS is a fundamental network infrastructure technology—in the same class as high-availability and security technologies. Like these other technologies, the basics of QoS have remained fairly steady for many years now, but there has always been a continuing evolution in the refinement and sophistication of specific QoS mechanisms and in the breadth of platforms where these tools are available. In addition, network and user requirements, and application types and volume, have all changed dramatically in the past few years and are continuing to do so.

However, the real changes—and the impetus for the second edition of this book—lie in several factors in QoS network design that are rapidly changing and evolving, including the following:

- The traffic mix and characteristics on a network continue to become more complex and sophisticated. This trend is continuing unabated as mobile communications devices, wireless network access, and business and consumer applications continue to evolve rapidly, placing ever more requirements on the transport and management of network traffic.

- Video applications are growing explosively. Video traffic is bandwidth hungry, and erratic, as compared to voice traffic, and constitutes many different traffic subtypes (for example, passive streaming video, interactive video, immersive video conferences), each of which require very different QoS network design techniques. Video traffic volume surpassed peer-to-peer network traffic in 2010. Globally, Internet video traffic is expected to be 86 percent of all consumer Internet traffic by 2016.
- Bandwidth in both the core and access layers of the network have changed rapidly as older technologies like Frame Relay and ATM disappear and newer IP access types (fiber, wireless) become commonplace. Bandwidth constraints and congestion points have not disappeared, but how and where in the network they manifest themselves are evolving.
- The platforms available in all places in the network (PINs) continue to evolve, and combined with the continuous refinement and increasing sophistication of QoS tools and features, the design and implementation of QoS techniques continue to change.
- Standards (Internet Engineering Task Force [IETF] Request For Comments [RFC]) and design guidelines in communications networks continue to evolve as the industry grapples with changing traffic mixes and policy behaviors and how best to accommodate these consistently on modern networks to optimize the user experience regardless of device used or network access technologies.
- Wireless network access is rapidly becoming ubiquitous and continues to widen its reach as more consumer and business devices become untethered and users expect appropriate communications access regardless of what network they have access to from their current location.
- QoS is used increasingly not just as a traffic management tool but also as a security method. Traffic patterns and characteristics monitored and analyzed by QoS tools to prioritize/drop traffic also provide valuable information about the security characteristics of the traffic.

All these factors imply that while QoS concepts, tools and techniques have reached maturity, the design of networks and the application of the tools during a network deployment are continuing to change rapidly.

History and Evolution

Electronic communications networks have existed for over a century and a half, starting with the first telegraph wiring that connected different locations. The technologies used to build these networks have changed dramatically and completely over this time. The entire network infrastructure has been replaced and superseded by waves of increasingly advanced technologies multiple times.

Then

Over a century ago, the public switched telephone network (PSTN) started building out a worldwide circuit-switched network. This network consisted of fixed-bandwidth dedicated circuits and was ideally suited to carrying real-time traffic, such as voice.

Some five decades later, networking experts from military and educational environments introduced packet-switched networks to circumvent any single points of failure, common in circuit-switched networks. Packet switching chops the information flow into small chunks, which can be addressed and routed over independent paths to the same destination.

The increased resilience of packet-switched networks caused a shift toward connectionless communications protocols that can handle packets that might arrive out of order. However, for many applications, this was complicated and insufficient. So, connection-oriented protocols such as X.25 and Systems Network Architecture (SNA), and later Frame Relay and Asynchronous Transfer Mode (ATM), were soon developed. In these protocols, a logical circuit (permanent virtual circuit [PVC] or switched virtual circuit [SVC]) is defined over the underlying connectionless packet network to handle a session of communication between two endpoints.

In theory, real-time communications such as voice and video can use “circuits” regardless of the underlying networking technology; however, real-time traffic transport over virtual circuits using packet-switched network technologies did not become practical until network processing and memory power became fast and cost-effective enough to enable application deployment. This became a reality in the mid-1990s.

However, other issues with carrying real-time communications over a packet-switched network soon manifested themselves. Packets delayed or dropped en route because of buffering, overflows, or other unexpected network events fell short of satisfying the end-user experience of the real-time session. Voice calls clipped, fax and modem calls dropped, and video frames pixilated or froze. The intelligent protocols defined in the seven-layer International Organization for Standardization (ISO) model recover over these network events with error-detection and correction capabilities, such as timeouts and retransmissions. Although sufficient for non-real-time data applications, these techniques do not generally suffice for real-time traffic types.

Both circuit-switched networks, like the PSTN, and packet-switched networks, like X.25, SNA, and later IP, attempted to evolve to accommodate real-time and non-real-time traffic types. Ultimately the flexibility, cost-effectiveness, and ubiquity of packet networks won out over the rigidity and wasteful bandwidth allocations of transporting data traffic on circuit networks. Even the traditional PSTN is by now predominantly built on packet-switched backbone technologies.

Now

To create the logical environment of a circuit for non-delayed delivery of real-time traffic on packet-switched IP networks, the concept of traffic classes evolved, and with that the concept of differentiated treatment of packets belonging to these different classes. Early on, SNA networks introduced the concept of a class of service (CoS) in its Advanced Peer-to-Peer Networking (APPN) architecture, and although APPN CoS used a very different approach from modern QoS, it was the forerunner of the concept that networking equipment could provide different levels of treatment to particular types of network traffic.

Several basic characteristics define a traffic class and thereby dictate the network's handling of packets belonging to that traffic class. Crucial among these characteristics are the following:

- **Delay (or latency):** This is the finite amount of time that it takes a packet to reach the receiving endpoint after being sent from the sending endpoint.
- **Jitter (or delay variation):** This is the variation, or difference, in the end-to-end delay in arrival between sequential packets.
- **Packet drops:** This is a comparative measure of the number of packets faithfully sent and received to the total number sent, expressed as a percentage.

Every traffic class has the potential of being sensitive to any of these characteristics. If the network applies processing, buffering, bandwidth, and memory resources to minimize the characteristics that each traffic class is sensitive to, then all traffic types can be transported over the same network and maximize the end-user experience of each type of session.

Evolution of QoS

In the early 2000s, the predominant traffic types managed on IP networks were voice and data. Voice traffic was real time and comprised constant and predictable bandwidth and packet arrival times. Data traffic was non-real-time and comprised unpredictable (or bursty) bandwidth and widely varying packet arrival times.

Since that time, various types of video traffic became increasingly important to business communications and operations. Video traffic comprises several traffic subtypes, including passive streaming video, real-time interactive video, and immersive video conferences. Video traffic can be real time (but isn't always), uses varied bandwidth requirements, and comprises different types of packets with different delay and loss tolerance within the same end-user experience (session). Figure 1-1 gives a broad characterization of the major categories of traffic.

Wireless access has also become increasingly popular over the past few years, and today a plethora of different devices roam and register with a variety of public and private wireless networks. 802.11 WiFi access networks have variable bandwidth and throughput characteristics depending on the location and power of the end device with respect to the access point (AP), and are based on a completely different Media Access Control (MAC) layer architecture than typical 802.3 Ethernet. This has introduced a whole new set of QoS challenges beyond those of fixed-bandwidth wired networks.

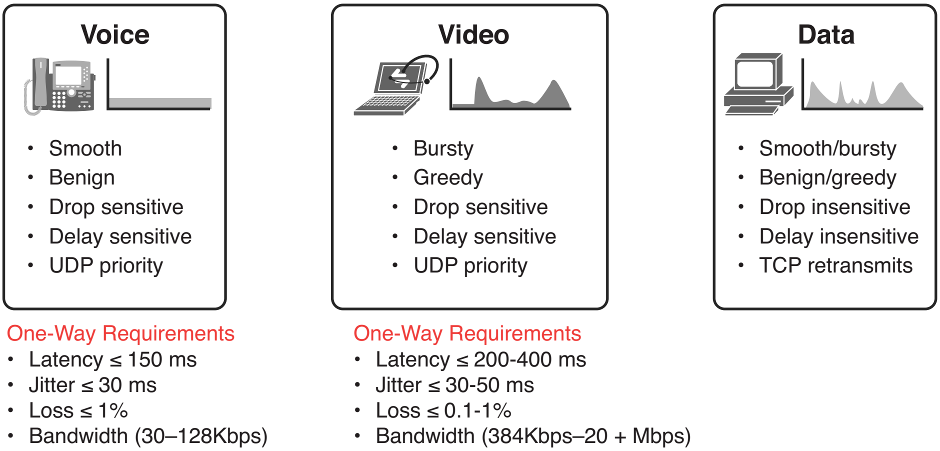


Figure 1-1 *Summary Characteristics of the Major Categories of Traffic*

Some period of time after the initial deployment of QoS features to provide different levels of services to end users, network administrators started implementing QoS features to manage traffic for security and business purposes. An enterprise would typically want to give its own internal training videos much better network treatment than a non-business-related Internet video played by an employee, even if both videos use the same technology and are from a technical aspect the same type of video traffic (stored broadcast or multicast video).

QoS Basics and Concepts

The fundamental purpose of QoS is to manage contention for network resources to maximize the end-user experience of a session—any kind of session. Because not all packets are equal, they should not be treated equally.

QoS features implement a system of *managed unfairness* in the network. Some sessions receive priority over other sessions; delay-sensitive sessions bypass queues of packets holding sessions less sensitive to delay; when queuing buffers overflow, packets are dropped on sessions that can recover from the loss or on those that can be eliminated with minimal business impact. To make space for the packets belonging to high-business-impact sessions that cannot tolerate loss without affecting the end-user experience, other sessions are *managed* (that is, packets are selectively delayed or dropped when contention arises) based on QoS policy decisions implemented in the network.

User Expectations: QoS, QoE, and QoX

The original and fundamental concept of QoS—different levels of network treatment for different packets—describes technical network performance and can be measured numerically in latency, jitter, and packet loss. The term *quality of experience* evolved later as network administrators became more attuned to end-user perception of the network's performance, instead of just to the technical metrics of the network. QoE is subjective and cannot be numerically measured but is nevertheless representative of user perception or assessment of the network and therefore remains an important goal for a network administrator. QoS features and techniques are deployed to maximize QoE for the end user. The term *QoX* followed later to encompass a broader concept of all subsidiary quality of *anything* (service, experience, and so on) terms.

The user expectation that "applications just work" transcends the type of device used or the type of network access available to the user. To build networks that deliver to this user expectation is not an easy or static task because bandwidth availability and traffic mixes continuously change with new applications and devices.

An unspoken implication of the concept of QoE is that QoS must be implemented and managed end to end on the network: The session runs between user A and user B, and QoE is what these users experience regardless of the number, type, or ownership of the interconnected networks that may separate them. QoS is near meaningless when implemented on only a segment of the network because the QoE perception is equal to the impairment imposed by the worst-performing segment of the network. QoS is an excellent example of the "only as strong as the weakest link" cliché.

QoS Models: IntServ and DiffServ

The first attempt to standardize QoS came in the mid-1990s, when the IETF published the integrated services (IntServ) RFCs (RFCs 1633, 2211, and 2212). These RFCs centered on a signaling protocol called the Resource Reservation Protocol (RSVP). RSVP signals bandwidth and latency requirements for each discrete session to each node along a path (logical circuit) from the sending endpoint to the receiving endpoint. Initially, RSVP required every node to heed its reservations, which was highly impractical over the Internet, on which servers, switches, and routers of every description, vintage, and vendor coexist. RSVP also requires every node to keep per-flow state.

To address these challenges, another set of standards—the differentiated services (DiffServ) model—soon emerged as a second attempt at standardizing QoS (RFCs 2474, 2597, 2598, 3246, 4594). The DiffServ model describes various behaviors to be adopted by each compliant node. The nodes can use whatever features are available (proprietary or otherwise), as chosen by the vendor, to conform. Packet markings, such as IP precedence (IPP) and its successor, differentiated services code points (DSCPs), were defined along with specific per-hop behaviors (PHBs) for key traffic types.

As the IntServ and DiffServ models have evolved, the general popularity of one method versus the other has swung back and forth with committed advocates on both sides. While the intellectual debate remains unresolved (neither model provides a complete solution), QoS implementations on today's network architectures have settled primarily on the DiffServ model, occasionally including an overlay of select IntServ features. However, the proliferation of variable-bandwidth access links such as 802.11 wireless networks and increasing volumes of delay- and jitter-sensitive traffic (video), for which the IntServ model has superior capabilities, has led to a reemergence of RSVP on modern networks.

The IntServ and DiffServ models are conceptually competing, but in practice they often prove more complementary. Therefore, these models are often co-deployed in network QoS implementations. IntServ is the only tool with dynamic network awareness for making per-flow admission control (AC) decisions, which makes it practical on network links where bandwidth is at a premium; on the other hand, DiffServ is far more flexible and scalable, and as such can be pervasively deployed throughout the network to ensure that PHBs are being met at every node.

Fundamental QoS Concepts and Toolset

A fair amount of literature exists on QoS concepts, technologies, and features. Therefore, the purpose of this book is not to reiterate in depth how each of these tools work, but to provide a quick overview of the key tools available and to show how these interact with each other and what combination of tools can be used to achieve the optimal QoS design objectives of a network.

Generally, QoS tools fall into the following categories:

- **Classification and marking tools:** Sessions, or flows, are analyzed to determine what traffic class they belong to and therefore what treatment the packets in the flow should be given. Once determined, the packets are marked so that analysis happens only a limited number of times, usually at the ingress edge of a network. A packet may traverse several different networks to its destination endpoint, and so reclassification and re-marking is fairly common at the hand-off points upon entry to a new network.

- **Policing, shaping, and markdown tools:** Different classes of traffic are allotted certain portions of network resources (which may be expressed in absolute or relative percentage terms). When traffic exceeds available network resources, some traffic may be selectively dropped, delayed, or re-marked to avoid congestion. Sessions are monitored to ensure that they do not use more than their allotment, and if they do, traffic is dropped (policing), slowed down (shaped), or re-marked (markdown) to conform.

- **Congestion management or scheduling tools:** When traffic exceeds available network resources, traffic is queued to await availability of resources. Traffic classes that do not react well to delay (queuing) fare better with the AC techniques that strive to keep traffic off the network unless there are “guaranteed” resources to transport the traffic without delay.

- **Link-specific tools:** Some link types require special handling and tools, such as fragmentation and interleaving techniques. Some links also have contractual bandwidth agreements (lower than the physical speed) not to be exceeded, and if a burst of traffic exceeds these limits, it is shaped (or slowed down) to conform to the agreement.

Figure 1-2 illustrates the QoS toolset and the typical sequence in which the functions are applied on traffic streams.

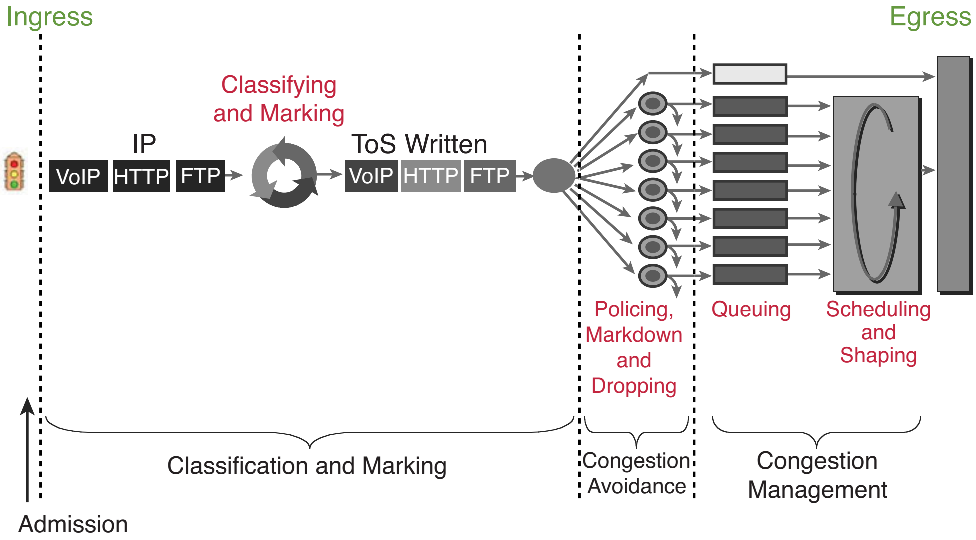


Figure 1-2 *QoS Toolset*

Packet Headers

Both IPv4 and IPv6 have space set aside in the packet headers for various markings. These bits are used to carry the packet marking assigned by the QoS classification tools to ensure that the rest of the network affords the packet the appropriate priority and packet-loss behaviors.

Figure 1-3 illustrates the IPv4 and IPv6 packet headers.

The IPv4 packet carries an 8-bit Type of Service (ToS) byte, whereas the IPv6 header carries an 8-bit Traffic Class field. The first 3 bits of both these fields are the IPP bits, giving a total of eight possible classes of service. These first 3 bits combined with the next 3 bits are known collectively as the DSCP bits. These 6 bits offer a maximum of 64 possible classes of service.

In addition to Layer 3 packet markings, there are also marking fields at the Layer 2 frame level, including Ethernet 802.1p CoS bits, the MAC bridge 802.1D user priority bits, Multiprotocol Label Switching Experimental values (MPLS EXP), virtual local-area network (VLAN) identification, Frame Relay Discard Eligible (FR DE), and Asynchronous Transfer Mode cell loss priority (ATM CLP) values. These fields are covered in more detail in Chapter 3, “Classification and Marking.”

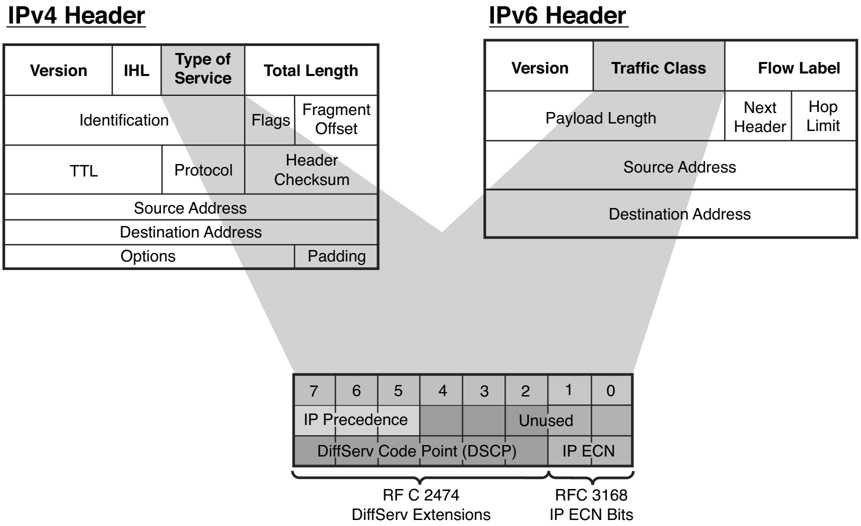


Figure 1-3 *IPv4 and IPv6 Packet Headers*

Simplifying QoS

In the late 1990s, Cisco spent significant development effort to implement an extensive QoS tool and feature set. The portfolio of QoS mechanisms and options became very rich and, simultaneously, very complicated to deploy. During the early 2000s, a number of QoS simplification and automation efforts were undertaken, including the following:

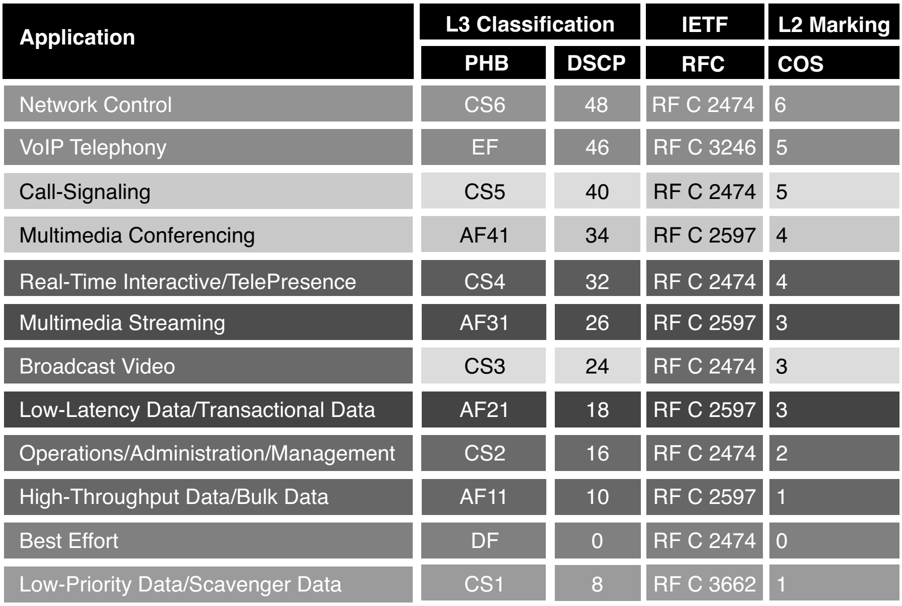
- The creation of a Modular QoS command-line interface (MQC)
- The establishment of a QoS baseline describing feature operation and default behaviors, an effort that was subsequently subsumed in various RFCs (RFCs 2474, 2597, 2598, 3246)
- Cross-platform feature consistency by implementing adherence to the MQC, QoS baselines, and RFC compliance
- Automatic QoS (AutoQoS), an intelligent macro that allows you to enter one or two simple commands to enable all the appropriate features for the recommended QoS settings for an application on a specific interface
- Hierarchical Queuing Framework (HQF) to port distributed QoS code and features to nondistributed platforms

Standardization and Consistency

In the late 1990s, when QoS features first became available, they varied considerably in configuration and operation. Early efforts toward increased consistency were in the form of a series of Cisco-developed *baselines*.

Over time, the IETF's suite of RFCs evolved to be comprehensive enough to, for the most part, guide the consistent operation of features across vendors' equipment and network components. A key contribution of the RFCs was the characterization of classes of traffic and how these should be identified and treated: the RFC-specified behaviors for the traffic classes.

These have evolved over time, and major traffic classes are now generally identified by RFC 4594 (as of 2006), including the classes shown in the first column in Figure 1-4.



Application	L3 Classification		IETF	L2 Marking
	PHB	DSCP	RFC	COS
Network Control	CS6	48	RF C 2474	6
VoIP Telephony	EF	46	RF C 3246	5
Call-Signaling	CS5	40	RF C 2474	5
Multimedia Conferencing	AF41	34	RF C 2597	4
Real-Time Interactive/TelePresence	CS4	32	RF C 2474	4
Multimedia Streaming	AF31	26	RF C 2597	3
Broadcast Video	CS3	24	RF C 2474	3
Low-Latency Data/Transactional Data	AF21	18	RF C 2597	3
Operations/Administration/Management	CS2	16	RF C 2474	2
High-Throughput Data/Bulk Data	AF11	10	RF C 2597	1
Best Effort	DF	0	RF C 2474	0
Low-Priority Data/Scavenger Data	CS1	8	RF C 3662	1

Figure 1-4 *RFC Guidelines for Traffic Classes*

The implementation and configuration of QoS features, however, continue to be vendor-specific. Cisco platforms have made great strides toward consistency as a result of the QoS baseline effort, and Cisco continues to strive to make the configuration of QoS features as consistent as possible across different product lines and software suites.

Because the 12 classes of traffic identified in RFC 4594 are too granular for the practical realities of many network implementations, the 12-class model can be simplified into 8- or 4-class models, as illustrated in Figure 1-5.

The most recent addition to the standards suite is RFC 5865, which pioneers the use of DiffServ markings to indicate an AC decision (which before was an IntServ function). This standard serves to blur the line between the originally separate and competing IntServ and DiffServ models. Although this RFC's implementation complicates QoS deployment and configuration, AC is a welcome addition because of the increasing requirement of real-time traffic types, because it is better that these are denied access to the network rather than being delayed/dropped.

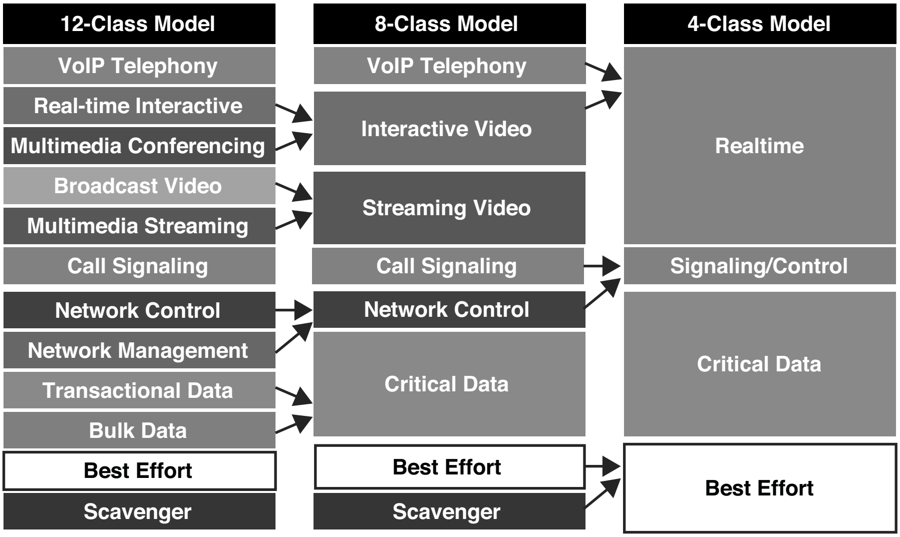


Figure 1-5 *The 4-, 8-, and 12-class Traffic Models*

Summary

This chapter introduced the concept of quality of service and explained why QoS is still a fundamental, critical, and evolving network technology. New devices, applications, and technologies are all rapidly changing and continue to challenge the network administrator to deploy QoS in such a way that QoE for the end user is maximized.

This brief history of QoS provides a background to explain how the industry has evolved to its current state. This chapter also provided an introductory overview of QoS concepts, tools, fundamental models, and industry standards.

Further Reading

General

Cisco.com QoS page: http://www.cisco.com/go/qos

Enterprise QoS Solution Reference Network Design Guide 3.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoS-SRND-Book.html

Enterprise Medianet Quality of Service Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html

IntServ

Cisco IntServ white papers: http://www.cisco.com/en/US/products/ps6611/prod_white_papers_list.html

IntServ is described by a series of RFCs from the IntServ IETF working group, all of which you can find on the IETF site, http://www.ietf.org:

RFC 1633: *Integrated Services in the Internet Architecture: An Overview*

RFC 2205: *Resource Reservation Protocol (RSVP) Version 1 Functional Specification*

RFC 2210: *The Use of RSVP with IETF Integrated Services*

RFC 2211: *Specification of the Controlled-Load Network Element Service*

RFC 2212: *Specification of Guaranteed Quality of Service*

RFC 2215: *General Characterization Parameters for Integrated Service Network Elements*

DiffServ

DiffServ—The Scalable End-to-End QoS Model (Cisco.com white paper): http://tinyurl.com/cmcogtc

DiffServ is described by a series of RFCs from the DiffServ IETF working group, all of which you can find on the IETF site, http://www.ietf.org:

RFC 2474: *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*

RFC 2475: *An Architecture for Differentiated Services*

RFC 2597: *Assured Forwarding PHB Group*

RFC 2598: *An Expedited Forwarding PHB*

RFC 2697: *A Single Rate Three Color Marker*

RFC 2698: *A Two Rate Three Color Marker*

RFC 3168: *Explicit Congestion Notification (ECN) to IP*

RFC 3246: *An Expedited Forwarding PHB* (replacing RFC 2598)

RFC 4594: *Configuration Guidelines for DiffServ Service Classes*

RFC 5865: *A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic*

Chapter 2

IOS-Based QoS Architectural Framework and Syntax Structure

This chapter covers the following topics:

- Quality of service (QoS) deployment principles
- Architectural framework of QoS features and capabilities of the IOS-based Cisco router and switch product line
- Command-line syntax and structure used to configure QoS features of the IOS-based Cisco router and switch product line

The features themselves, their operation, and their individual configuration commands are discussed in more depth in Chapters 3 through 7. The features, operation, and syntax of QoS features on non-IOS-based Cisco products, primarily the wireless product line, are discussed in Chapters 18 through 20.

QoS Deployment Principles

You should have a clear understanding of the business objectives or organizational goals of deploying QoS in your network before investigating the details of features and configurations. A general best-practice approach to a successful QoS deployment includes the following steps:

1. Clearly define the business/organizational objectives of the QoS deployment: These may include provisioning real-time services for voice/video traffic or guaranteeing the servicing of business-critical data applications or managing unwanted traffic traversing business networks. Also, it is recommended to seek executive endorsement of these business objectives before QoS design and network deployment (so that midway through the deployment political differences of opinion do not derail the process).
2. Based on these business objectives, determine how many application classes are required to meet these goals. Also, define an end-to-end strategy of how these classes are to be identified and treated across the network.

3. Analyze the service-level requirements of each application class so that appropriate QoS tools can be matched to meet these requirements.
4. Design platform-specific QoS policies to meet application requirements, taking into account appropriate place-in-the-network (PIN) considerations.
5. Test the QoS designs in a controlled environment—such as a lab—to verify that the policies are meeting the intended application requirements.
6. Begin deployment with a closely monitored and evaluated pilot rollout.
7. Deploy the tested and pilot-proven QoS designs to the production network in phases during scheduled downtime.
8. Monitor service levels to ensure that the QoS objectives are being met.

Successful QoS deployments begin not at the technical level, but at an organizational level, with determining the business objectives of QoS. Then you can identify how many classes of traffic you need to meet these objectives. It is only at this point that the technical part of the process begins, with defining a QoS strategy to identify the various applications classes and to specify how these are to be treated end-to-end across the network. Following this, the service level requirements of each traffic class needs to be analyzed, such that appropriate tools and policies can be designed to meet these.

QoS Architectural Framework

Figure 1-2 in Chapter 1, "Introduction and Brief History of QoS and QoE," introduced the principal QoS functions, including the following:

- **Classification:** Determine what class of traffic every packet belongs to.
- **Marking:** Write a value in the packet header to indicate the class of traffic the packet belongs to once it is identified and classified.
- **Policing (dropping and markdown):** Determine when and where to drop packets, or re-mark them, when traffic exceeds available resources.
- **Shaping:** Slow down traffic to fit a given bandwidth rate (often so as not to exceed a service provider contract).
- **Queuing:** Buffer packets to await transmission when ingress traffic exceeds available egress resources (bandwidth).
- **Bandwidth allocation:** Monitor and ensure that certain traffic classes use no more than configured amounts of bandwidth.
- **Admission control:** Determine whether a packet should be admitted or rejected upon entry into the network based on available resources.

The QoS functions do not operate in isolation, and every function may consist of multiple different features. The architectural framework discussed next provides context for how the features interact with one another and how they are sequenced before the discussion proceeds to the structure of the IOS-based Modular QoS command-line interface (MQC) syntax.

QoS Behavioral Model

The QoS conceptual behaviors are independent of the underlying platform-specific implementations, and it is recommended for the network administrator to design and configure QoS features along these general behaviors rather than the specifics of any one platform or its place in the network (PIN). Figure 2-1 illustrates the relationship between the human configuration interface, the underlying behavioral model, and the often platform-specific implementations.

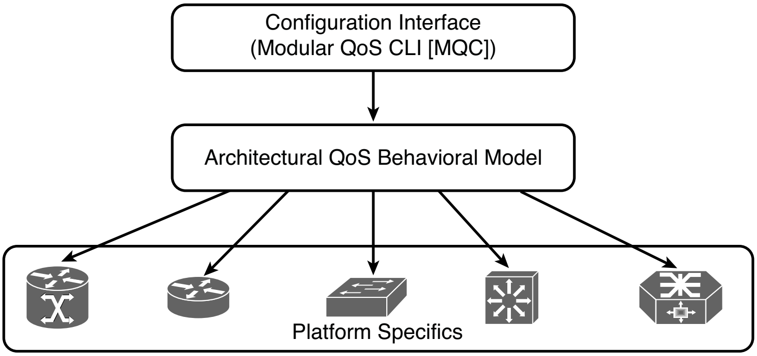


Figure 2-1 *QoS Behavioral Model*

QoS Feature Sequencing

At the next level of detail, it is useful to think of the QoS behaviors as having a general sequence to them (as suggested in Figure 1-2), including the following:

- **Classification:** The semantics for identifying each traffic stream.
- **Pre-queuing:** Admission decisions, and dropping and marking the packet, are best applied before the packet enters a queue for egress scheduling and transmission.
- **Queuing:** Decisions on the scheduling order of packets waiting for transmission.
- **Post-queuing:** Usually optional, but sometimes needed to apply actions that are dependent on the transmission order of packets, such as sequence numbering (for example, for compression or encryption), which isn't known until the QoS scheduling function dequeues packets based on its priority rules.

At yet a further level of detail, it is useful to think of the QoS behaviors and their general sequencing in terms of the flow of a packet through a network node and the interaction of the QoS features with other packet ingress and egress features. Figure 2-2 provides a high-level overview of the feature sequence for a packet, with blocks 1, 2, 6, 7, and 12 representing non-QoS feature actions.

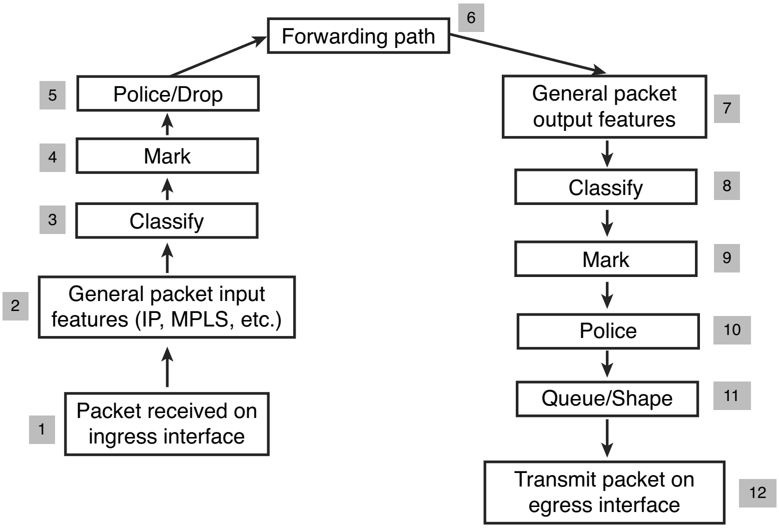

6
Forwarding path
5
Police/Drop
4
Mark
3
Classify
2
General packet input features (IP, MPLS, etc.)
1
Packet received on ingress interface
General packet output features
7
Classify
8
Mark
9
Police
10
Queue/Shape
11
Transmit packet on egress interface
12

Figure 2-2 *High-Level Packet Feature Sequence*

Classification may happen on ingress (3)/egress (8) in the packet path. In general, classification should be done as close to the source as possible (in other words on ingress), but sometimes the attributes for classification are not known until after the output features (7) have been executed. This is especially true for locally generated packets (rather than transit packets) which do not arrive via an ingress path.

Similarly, a packet may be marked on ingress (4), but perhaps requires to be re-marked on egress (9) based on changes in conditions or more knowledge about the traffic class upon egress. For example, a policer may re-mark a nonconforming packet on egress rather than drop it—the fact that the packet is nonconforming is not known until the packet reaches the egress side of the path and the decision depends on the traffic conditions at the time and not on any unique attribute of the packet itself.

Modular QoS Command-Line Framework

The MQC is the syntax structure strives to provide a consistent, platform-independent and flexible configuration interface to simplify configuring QoS features on Cisco IOS-based platforms. To this end, the MQC abstracts the QoS behavioral model to a level of simplicity where the network administrator does not have to know the details of the platforms where the syntax is executed.

MQC Syntax

The MQC defines a syntax framework for each of the following steps in implementing a QoS configuration:

1. Define traffic classes and identify what traffic belongs to each class.
2. Define the actions, or policies, that should be applied to each traffic class (for example, the marking to apply or the bandwidth to be allotted).
3. Associate the policies with specific logical or physical interfaces.

The preceding configuration steps are accomplished via the use of three basic command sets: the **class-map**, the **policy-map**, and the **service-policy** statements.

- **class-map:** This command set specifies the packet matching criteria that identifies packets that belong to a class.
- **policy-map:** This command set defines actions to be applied to each traffic class. One or more policies are specified for each traffic class. Examples of policies include policing the traffic class to a maximum bandwidth rate, or guaranteeing the traffic class a priority level in queuing.
- **service-policy:** This command is used to attach a policy map (and thereby the associated policies for each traffic class you defined) to a logical or physical interface and to specify the direction (input or output) in which the policy is to be applied.

Example 2-1 shows the generic structure of the MQC syntax.

Example 2-1 *Generic MQC Syntax Structure*

```
! Define class maps: Packet matching criteria for classification
class class-map-name-n
  <match-1>
  <match-2>
  ...
  <match-n>

! Define policy maps: Apply actions to each traffic class
policy-map policy-map-name
  class class-map-name-1
    <policy-1>
    <policy-2>
    ...
    <policy-n>

! Define service policy: Attach policy to an interface
interface serial 1/0/0
  service-policy output policy-map-name
```

Example 2-2 demonstrates a specific implementation example of a QoS configuration using the MQC structure. This example shows a configuration defining four traffic classes, three named ones (REALTIME, CONTROL, CRITICAL-DATA), and an implicit "default" class. In this example, packets are classified based on markings already applied previously to the packet (the **match** statements). Earlier in this chapter in Figure 2-2, the QoS feature sequence did classification of the packets (box 8) and then applied the markings (box 9) based on the result of the classification. Both variations of this behavior are common; edge nodes often classify packets based on analysis of the traffic and then apply markings to the packet (as shown in Figure 2-2), and all subsequent nodes use the preexisting markings to classify packets and apply policies to them (as shown in Example 2-2).

Example 2-2 *MQC Syntax Structure Example*

```
Router# show run
class-map match-any REALTIME
  match dscp ef       ! Matches VoIP bearer traffic
  match dscp cs5      ! Matches Broadcast Video traffic
  match dscp cs4      ! Matches Realtime-Interactive traffic
!
class-map match-any CONTROL
  match dscp cs6      ! Matches Network-Control traffic
  match dscp cs3      ! Matches Voice/Video Signaling traffic
  match dscp cs2      ! Matches Network Management traffic
!
class-map match-any CRITICAL-DATA
  match dscp af41 af42 af43  ! Matches Multimedia Conf. on AF4
  match dscp af31 af32 af33  ! Matches Multimedia Streaming on AF3
  match dscp af21 af22 af23  ! Matches Transactional Data on AF2
  match dscp af11 af12 af13  ! Matches Bulk Data on AF1
!
policy-map WAN-EDGE-4-CLASS
  class REALTIME
    priority percent 33      ! 33% LLQ for REALTIME class
  class CONTROL
    bandwidth percent 7      ! 7% CBWFQ for CONTROL class
  class CRITICAL-DATA
    bandwidth percent 35     ! 35% CBWFQ for CRITICAL-DATA class
    fair-queue               ! Fair-queuing on CRITICAL-DATA
    random-detect dscp-based ! DSCP-based WRED on CRITICAL-DATA
  class class-default
    bandwidth percent 25     ! 25% CBWFQ for default class
    fair-queue               ! fair-queuing on default class
    random-detect dscp-based ! DSCP-based WRED on default class
!
interface serial 1/0/0
  service-policy output WAN-EDGE-4-CLASS
```

Note that class map and policy map names are case sensitive. Thus, **class-map type1** is different from **class-map Type1**, which is different from **class-map TYPE1**. Class map names and cases must match exactly the class names specified in policy maps.

Note A convention used in this book is to type class map, policy map, and access list names in UPPERCASE, so as to better distinguish these from IOS commands.

Default Behaviors

Unclassified traffic (traffic that does not meet the match criteria specified in the explicit traffic classes) is treated as belonging to the implicit default class. The default class is automatically part of any MQC configuration.

As shown earlier in Example 2-2, the default class cannot be specified in the class map part of the configuration because it contains by definition all traffic which is left unclassified by the explicit class maps in the configuration. However, default class traffic can be assigned QoS features (queuing, shaping, bandwidth, and so on) by specifying a policy map for **class-default** as in the CLI shown in Example 2-2. This is optional, and if not specified, default class traffic has no QoS features assigned, receives best-effort treatment, and can use all bandwidth not allotted or needed by the classes explicitly specified in the configuration.

The default treatment for unclassified traffic with no QoS features enabled is a first-in, first-out (FIFO) queue with tail drop (which treats all traffic equally and simply drops packets when the output queue is full).

Traffic Classification (Class Maps)

The three most common ways to classify traffic are as follows:

- **Markings:** Examine the Layer 2 (class of service [CoS]) or Layer 3 (IP precedence [IPP] or differentiated services code point [DSCP]) settings
- **Addressing:** Examine the source/destination interface, or the L2 destination address, or the L3 source/destination address, or source/L4 destination port. Using an IP address (for example an IP subnet) classifies traffic by a group of devices, whereas classifying by port number tends to classify by traffic type.
- **Application signatures:** Examine application content inside the packet payload, also known as deep packet inspection.

Marking- and addressed-based classifications use packet/frame/tag header information, whereas application-based classification examines the (L7) application layer packet payload:

A **class-map** is the CLI construct used to classify traffic for all three mechanisms (marking-based, address-based, and application-based classifications). A **class-map** contains one or more **match** statements to identify the traffic belonging to the class and can use one—or a combination of—the three basic mechanisms depending on the keywords following the **match** statement. You can match CoS or DSCP markings, or match specific source or destination addresses, or specific protocol or application information, as shown in Example 2-3.

Example 2-3 *Class Map Example*

```
Router# show run
class-map markings
  match dscp af41 af42 af43
!
class-map mac-address
  match destination-address mac 00:00:00:00:00:00
!
class-map ftp
  match protocol ftp
```

The **class-map** CLI also allows you to specify logical combinations of match criteria to identify a class. To satisfy the classification semantics for a class of traffic, you can

- Match all criteria in a set of statements (**match-all**)
- Match none of the criteria in a set of statements (**match not**)
- Match at least one of the criteria in a set of statements (**match-any**)

Note The default logical operator (if unspecified) is **match-all.**

Definition of Policies (Policy Maps)

Policy maps contain one or more actions or QoS treatments to be specified for each class of traffic identified by the class maps. Example 2-2 earlier showed a sample policy map named WAN-EDGE-4-CLASS, which specifies the QoS treatment policies for four classes of traffic. The treatments shown in the example include bandwidth allocation (the **priority** and **bandwidth** keywords), queuing algorithms (the **priority** and **fair-queue** keywords), and packet-dropping thresholds (the **random-detect** keyword).

The types of actions that can be included in a policy definition include the following:

- Bandwidth allocation
- Queuing directives (priority queuing, fair queuing, queue length limits)

- Traffic shaping (rate limiting by queuing or delaying packets)
- Packet dropping (policing or rate limiting by dropping packets, random dropping, tail dropping, or unconditional dropping)
- Marking packets (setting various header fields to certain values)
- Counting packets
- Compression of the packet header
- Admission decisions

Although the sequence in which class maps are defined within the configuration is unimportant, the sequence of classes within a policy map is significant. As with access control list (ACL) logic, policy maps apply a “first-true-match” rule, meaning that a packet is examined against each subsequent class within a policy map until a match is found. When found, the examination process terminates, and no further classes are checked. If no matches are found, the packet ends up in the default class.

To illustrate the significance of sequencing classes within a policy map, consider the policy shown in Example 2-4, which includes two classes of traffic: VOICE for voice traffic and FAX-RELAY for fax traffic. The sequence of class maps FAX-RELAY and VOICE within the global configuration does not matter; these can be entered in any order. However, the desire is to treat fax traffic differently from voice traffic (to control strictly the maximum bandwidth used by each class of traffic) despite the fact that they are marked the same way—all packets are marked to DiffServ Code Point Expedited Forwarding (DSCP EF) at their respective sources.

Example 2-4 *Policy Map Sequence Example*

```
Router# show run
class-map match-all FAX-RELAY
  match dscp ef
class-map match-all VOICE
  match protocol rtp audio
!
policy-map VOICE-AND-FAX
class VOICE
  priority 216
  police cir 216000
class FAX-RELAY
  priority 64
  police cir 64000
```

The policy map VOICE-AND-FAX shown in Example 2-4 provides this operation through careful ordering of the classes within it. Class VOICE is placed first and performs Network Based Application Recognition (NBAR) classification to identify whether the traffic is Real-time Transfer Protocol (RTP) audio (in other words, voice). Only traffic that fails this examination is checked against the second class under the policy map (class FAX-RELAY).

Class FAX-RELAY checks whether the packet's DSCP value is EF. Because only two types of traffic have DSCP values of EF (voice and fax relay) and voice has already been filtered out, any remaining traffic that matches these criteria must be fax relay. Fax-relay traffic then is administratively assigned the same queuing policy, but a different bandwidth allocation. If the sequence of the two **class** statements within the policy map were reversed, the policy would work differently: No traffic would ever show against the VOICE class because both voice and fax-relay traffic would match on DSCP EF and would therefore be assigned to the FAX-RELAY class. Figure 2-3 shows the decision hierarchy for each packet examined by the policy map VOICE-AND-FAX.

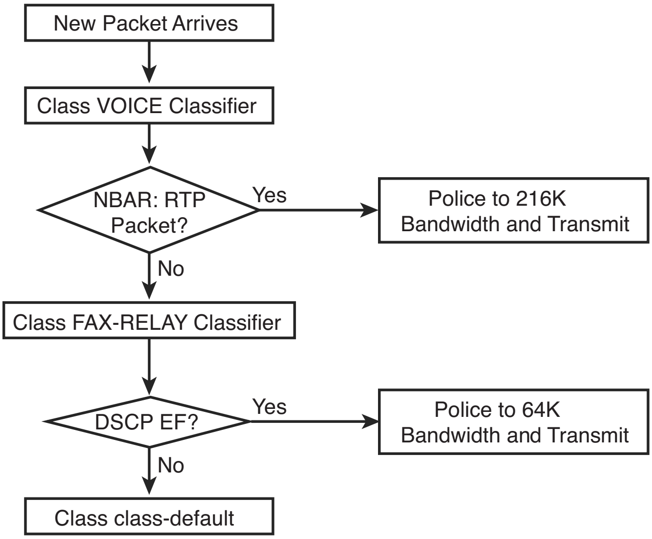

New Packet Arrives
Class VOICE Classifier
NBAR: RTP Packet?
Yes
Police to 216K Bandwidth and Transmit
No
Class FAX-RELAY Classifier
DSCP EF?
Yes
Police to 64K Bandwidth and Transmit
No
Class class-default

Figure 2-3 *Packet Action Decisions by Policy Map VOICE-AND-FAX*

Attaching Policies to Traffic Flows (Service Policy)

A policy map (and thereby the associated policies for each traffic class you defined) is attached to a logical or physical interface, including the following:

- Main interfaces
- Subinterfaces
- Multilink Point-to-Point Protocol (MLPPP) bundles
- Virtual templates
- Virtual local-area networks (VLANs)

- Asynchronous Transfer Mode (ATM) or Frame Relay (FR) virtual circuits (VCs)

The **service-policy** command also specifies whether the policies should be applied to ingress or egress traffic on this interface, as shown in Example 2-5. The targets of the **service-policy** command may vary across different platforms, and in the wireless space service set identifications (SSIDs) may also be used.

Example 2-5 *Ingress and Egress Policies*

```
Router# show run
policy-map POLICY-1
  ...
  bandwidth 20000
policy-map POLICY-2
  ...
  bandwidth 64000
!
interface Ethernet 1/0
  service-policy input POLICY-1
!
interface Ethernet 1/1
  service-policy output POLICY-2
```

Hierarchical QoS and HQF

Hierarchical or nested policies have long been allowed in the MQC framework and syntax and allow you to apply some policy actions per subinterface (for example, policing certain traffic classes on certain subinterfaces not to exceed a given rate) and overall policy actions at the interface level that apply to all traffic (such as an aggregate shaping rate to comply with a service provider contract).

Figure 2-4 shows hierarchical policies on subinterfaces 1.1 and 1.2 on main interface Gigabit Ethernet (GE) 1. In this example, the two subinterfaces are collectively allowing 22K of traffic onto the main interface, which in turn is shaped to 20K throughput. Therefore, if both subinterfaces use their full allotted bandwidth simultaneously, additional traffic shaping takes place on the main interface to maintain an aggregate throughput not to exceed 20K.

To achieve a hierarchical policy, you apply a service policy to a policy map instead of the interface configuration, as shown in Example 2-6. The example shows all traffic at the interface level being shaped overall to 20K (**policy-map AGGREGATE**), while voice traffic within that rate is guaranteed to get a minimum of 3K of bandwidth.

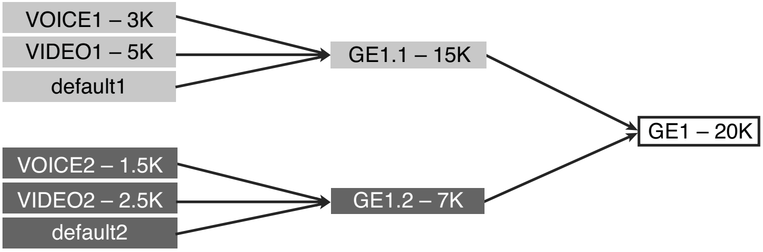


Figure 2-4 *Hierarchical Policies*

Example 2-6 *Hierarchical Policies*

```
Router# show run
! Definitions for sub-interface GE1.1
policy-map CHILD1
  class VOICE1
    priority 3000
  class VIDEO1
    bandwidth 5000
policy-map PARENT1
  class class-default
    shape average 15000
    service-policy CHILD1
!
! Definitions for sub-interface GE1.2
policy-map CHILD2
  class VOICE2
    priority 1500
  class VIDEO2
    bandwidth 2500
policy-map PARENT2
  class class-default
    shape average 7000
    service-policy CHILD2
!
! Definitions for the main interface
policy-map AGGREGATE
  class class-default
    shape average 20000
!
interface ge 1/1.1
  service-policy output PARENT1
interface ge 1/1.2
  service-policy output PARENT2
interface ge 1/1
  service-policy output AGGREGATE
```

In 2006, the QoS Hierarchical Queuing Framework (HQF) was introduced to enhance hierarchical policies to support hierarchical queuing. This means that queuing can be applied per subinterface (or at a logical level above the egress interface), while interface queuing works in concert with those logical levels of queuing. The interface level queuing algorithm provides backpressure to the higher levels to ensure proper packet scheduling across the hierarchy.

Legacy QoS CLI No Longer Used

The MQC CLI has existed in Cisco IOS Software for over a decade, but there are still QoS configuration commands that predate the existence of the MQC. These legacy CLI commands have long been maintained in parallel with the MQC syntax and cause confusion as to the interaction of the QoS features configured outside of the MQC with those configured with the MQC. To this end, several legacy pre-MQC QoS CLI commands were deprecated starting in July 2011; the 15.2M and 15.2T software releases issued a warning that in time they would likely be removed. If you have older platforms that use these commands, you should migrate these to the equivalent MQC commands at your earliest convenience.

In summary, the QoS features with legacy CLI outside the MQC that must be migrated to their MQC equivalents include those given in Table 2-1. For details of the equivalent MQC syntax, see the Cisco Product Bulletin (PB) *Legacy QoS CLI Commands Deprecation*, PB580832, April 2012.

Table 2-1 *Summary of Legacy QoS CLI for Migration to the MQC CLI*

QoS Feature	Legacy CLI
Weighted random early detection (WRED) or distributed WRED	**random-detect** **random-detect dscp** **random-detect** (*dscp-based keyword*) **random-detect exponential-weighting-constant** **random-detect** (*prec-based keyword*) **random-detect precedence**
Bandwidth allocation	**max-reserved-bandwidth**
Custom queuing	**custom-queue-list**
Priority queuing	**ip rtp priority**
Weighted fair queuing	**fair-queue**
Assigning a priority group to an interface	**priority-group**

QoS Feature	Legacy CLI
Frame Relay-specific commands	**frame-relay congestion threshold de** **frame-relay custom-queue-list** **frame-relay congestion threshold ecn** **frame-relay fair-queue** **frame-relay ip rtp priority** **frame-relay priority-group** **frame-relay adaptive-shaping** (*becn keyword*) **frame-relay adaptive-shaping** (*foresight keyword*) **frame-relay fecn-adapt** **frame-relay bc** **frame-relay be** **frame-relay cir**
show commands	**show queue** **show queueing** **show interfaces random-detect** **show random-detect-group** **show traffic-shape** **show traffic-shape queue** **show traffic-shape statistics** **show interfaces fair-queue**

AutoQoS

To simplify QoS deployment, Cisco has developed the Automatic QoS (AutoQoS) feature to provision automatically best-practice QoS designs. AutoQoS is an intelligent macro that allows an administrator to enter one or two simple commands to enable recommended QoS features and settings for applications on a specific interface on a specific platform. You can use the AutoQoS configurations as is, or you can leverage them as a starting point to deploy quickly the bulk of the configuration and then subsequently tailor and fine-tune them to your more specific requirements.

AutoQoS VoIP, the first release of the AutoQoS features, became available on Cisco IOS router platforms in Release 12.2(15)T. AutoQoS VoIP provides best-practice QoS designs for VoIP on Cisco Catalyst switches and Cisco IOS routers. By entering one global or one interface command, depending on the platform, the AutoQoS VoIP macro expands these commands into the recommended VoIP QoS configurations (with all the calculated parameters and settings) for the platform and interface on which the AutoQoS CLI is being applied.

For campus Catalyst switches, AutoQoS VoIP automatically performs the following:

- Enforces a trust boundary at Cisco IP phones
- Enforces a trust boundary on switch access ports and uplinks/downlinks
- Enables strict-priority queuing for voice and weighted round-robin queuing for data traffic
- Modifies queue admission criteria (CoS-to-queue mappings)
- Modifies queue sizes and queue weights where required
- Modifies CoS-to-DSCP and IP precedence-to-DSCP mappings

For Cisco IOS routers, AutoQoS VoIP is supported on FR, ATM, High-Level Data Link Control (HDLC), MLPPP, and FR-to-ATM links, and automatically performs the following:

- Classifies and marks VoIP bearer traffic (to DSCP EF) and call-signaling traffic (to DSCP CS3)
- Applies the appropriate scheduling configuration:
  - Low-latency queuing (LLQ) for voice
  - Class-based weighted fair queuing (CBWFQ) for call signaling
  - Fair queuing (FQ) for all other traffic
- Enables Frame Relay traffic shaping (FRTS) with optimal parameters, if required
- Enables link fragmentation and interleaving (LFI), either MLP LFI or FRF.12, on slow (768 Kbps) links, if required
- Enables IP RTP header compression (cRTP), if required
- Provides Remote Monitoring (RMON) alerts of dropped VoIP packets

AutoQoS Enterprise, the second release of the AutoQoS features, became available on Cisco IOS router platforms in 12.3(7)T. AutoQoS Enterprise, for Cisco IOS routers only, detects and provisions for up to 10 classes of traffic, as follows:

- Voice
- Interactive video

- Streaming video
- Call signaling
- Transactional data
- Bulk data
- Routing
- Network management
- Best effort
- Scavenger

The AutoQoS Enterprise feature consists of two configuration phases, completed in the following order:

- **Autodiscovery (data collection):** Uses NBAR-based protocol discovery to detect the applications on the network and performs statistical analysis on the network traffic.
- **AutoQoS template generation and installation:** Generates templates from the data collected during the autodiscovery phase and installs the templates on the interface. These templates are then used as the basis for creating the class maps and policy maps for your network. After the class maps and policy maps are created, they are then installed on the interface.

In 2010, an updated version of AutoQoS for video applications was released in 12.2(55) SE for the Cisco Catalyst 2960-G/S, 2975-GS, 3560-G/E/X, and 3750-G/E/X family of switches. These configurations deploy the recommended QoS designs for rich-media application support across this family of switches. This release of AutoQoS provides four main ingress QoS policy options in interface configuration mode:

- **auto qos voip** [**cisco-phone** | **cisco-softphone** | **trust**]: This option expands on the VoIP models to include provisioning for additional classes of rich-media applications and to include data plane policing/scavenger class QoS policy elements to protect and secure these applications.
- **auto qos trust** {**cos** | **dscp**}: This option configures the port to trust statically either CoS or DSCP. If neither CoS nor DSCP is explicitly specified, the **auto qos trust** command configures (by default) CoS trust on Layer 2 switch ports and DSCP trust on Layer 3 routed interfaces.
- **auto qos video** [**cts** | **ip-camera**]: This new option provides automatic configuration support for both Cisco TelePresence systems (via the **cts** keyword) and IP video surveillance cameras (via the **ip-camera** keyword).
- **auto qos classify** {**police**}: This option provides a generic template to classify and mark up to six classes of rich-media traffic and optionally provision data plane policing/scavenger class QoS policy elements for these traffic classes (via the optional **police** keyword).

Summary

This chapter briefly covered QoS architecture, syntax, and structure. Chapter content explained the architectural and behavioral model of the QoS features and the MQC framework and syntax structure of class maps, policy maps, and service policies. In addition, you learned about QoS deployment principles and hierarchical and AutoQoS capabilities.

This chapter provides the foundation necessary for the specific feature discussions following in the next six chapters.

Further Reading

General

Quality of Service Solutions Configuration Guide Library: http://www.cisco.com/en/US/docs/ios-xml/ios/qos/config_library/15-0m/qos-15-0m-library.html

Modular QoS (MQC) Command-Line Interface Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_mqc/configuration/15-1mt/Applying_QoS_Features_Using_the_MQC.html

Hierarchical Queuing Framework (HQF) Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_hrhqf/configuration/15-0m/qos-hrhqf-15-0m-book.html

Cisco Product Bulletin Legacy QoS CLI Commands Deprecation, PB580832, April 2012 (also available at http://www.cisco.com/go/qos > Data Sheets and Literature > Bulletins): http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6558/product_bulletin_c25-580832.html

AutoQoS

AutoQoS At-a-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/autoqosmediacampus.pdf

Overview: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html#wp61040

Tools: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoSIntro.html#wp46186

AutoQoS Enterprise: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/BranchQoS.html#wp100320

AutoQoS for switches: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html#wp1098289

Technical presentation: http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6558/ps6613/ps6656/prod_presentation0900aecd80313756.pdf

AutoQoS Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_auto/configuration/15-1mt/qos-auto-15-1mt-book.html

This page intentionally left blank

Chapter 3

Classification and Marking

This chapter covers the following topics:

- Terminology
- Packet markings in different technologies: packet and tunnel header fields used for classification and marking
- Classification tools and mechanisms
- Marking tools and mechanisms
- Recommendations

The first step in implementing a quality of service (QoS) policy is to analyze and identify the traffic that is to be treated by some set of actions. Once determined, the packets belonging to the traffic class are usually marked by writing a value into a header field so that the traffic classification analysis happens only once (usually at the ingress edge of the network, and as close to the source of the traffic as feasible).

Classification and Marking Topics

Before looking at each of the specific classification and marking tools in greater depth, let's review a few general classification and marking topics:

- Classification and marking terminology
- Security and QoS
- Challenges posed by video and wireless traffic
- Marking fields in different technologies
- Mapping of QoS markings

Classification and Marking Terminology

Although the terms *classification* and *marking* are often used interchangeably, the terms represent distinct and different actions:

- **Classification:** An action that sorts packets into different traffic types, to which different policies can then be applied. Classification of packets can happen without marking.
- **Marking (or re-marking):** Writes a value into the packet header. Marking usually establishes a trust boundary at the network edge or at the intersection between two different networks, where preexisting packet markings are accepted or rejected (and as such, re-marked). Marking also can be used in other locations in the network and is not always used solely for purposes of classification.

As with the general terms classification and marking, the actual tools apply different actions to traffic:

- **Classifier tool:** Inspects one or more fields in a packet to identify the type of traffic the packet is carrying. Once identified, the traffic is directed to a policy-enforcement mechanism for that traffic type, where it receives predefined treatment, including marking, queuing, policing, shaping, or any combination QoS actions.
- **Marker tool:** Writes a value in the header of the packet, frame, tag, cell, or label to preserve the classification decision reached at the network trust boundary. Nodes subsequent to the trust boundary do not have to repeat the in-depth classification and analysis (which can be computationally intensive tasks) to determine how to treat a packet.

Differentiated Services Code Point (DSCP), Class Selector (CS) code point, Class of Service (CoS), Type of Service (ToS), and Traffic Identifier (TID) are all different terms used to indicate a designated field in a Layer 2 frame or Layer 3 packet header. Field values have specific meanings about the treatment or service expected by the frame or packet.

All five terms have the same conceptual meaning and intent, and are in some cases used interchangeably, but the width of the fields and the precise meaning of individual values differ:

- CoS is usually used in conjunction with Ethernet Layer 2 frames (predominantly with IEEE 802.1Q/p) and contains 3 bits.
- ToS is generally used to indicate the Layer 3 IPv4 packet field and comprises 8 bits, 3 of which are designated the IP Precedence field, as defined in RFC 791. IPv6 RFC 2460 changed the terminology to Traffic Class for this same field in the packet header.
- DSCP is a set of values, based on a 6-bit width, that are used to encode the meaning of the Layer 3 IPv4 ToS field (the far left 6 of the 8 bits in that field, as given in

Figure 1-3 in Chapter 1, “Introduction and Brief History of QoS & QoE”) according to RFCs 2474, 2597, and 2598.

- CS is a term used to indicate a 3-bit subset of DSCP values; it designates the same 3 bits of the field as IP Precedence, but the interpretation of the field values maps to the per-hop behaviors as per the RFCs defining 6-bit DSCPs.
- TID is a term used to indicate a 3-bit field in the QoS Control field of the 802.11 WiFi MAC frame. The 8 values of this field correspond to eight user priorities (UPs) and map roughly, but not exactly, to the Ethernet CoS values and meanings. TID is typically used for wireless Ethernet connections, and CoS is used for wired Ethernet connections.

Security and QoS

Important intersections exist between security and QoS, including the following:

- Trust boundaries
- Recognizing, classifying, and treating traffic that represents a worm or other security attack against the network

Trust Boundaries

A trust boundary is a network location where packet markings are not accepted (and so may be rewritten). Conversely, trust domains are network locations where packet markings are accepted and acted on.

The untrusted domain may include network segments and devices with direct end user access, such as computers and printers, whereas the trusted part of the network includes the routers and switches only the network administrator has access to. A trust boundary may also be established between an enterprise network and a service provider network: The service provider may establish its own trust boundary regardless of how secure the enterprise network might consider itself. Trust boundaries are also common in governmental and educational networks between different departments, ministries, institutes, schools, or organizations. In an enterprise campus network, the trust boundary is almost always at the edge switch.

QoS markings on traffic arriving from an untrusted domain are usually ignored, the traffic re-inspected, reclassified, and re-marked before it is forwarded to the trusted network. This prevents rogue devices, or allowing end-user-controlled markings on their devices, from taking unfair or disastrous advantage of the network’s QoS treatments. For example, a user computer set up to mark all traffic at DSCP EF will be ignored by the access switch at the trust boundary, and the traffic is inspected and re-marked according to enterprise QoS policies implemented on the switch.

Network Attacks

Denial-of-service (DoS) and worm attacks are increasing in frequency, complexity, and scope of damage. QoS tools and designs can mitigate the effects of worms and keep critical applications available during DoS attacks.

For example, the QoS “Scavenger class,” based on RFC 3662, is intended to provide a “less-than-best-effort” service to traffic that has little or no contribution to organizational objectives, such as entertainment-oriented applications. Specifically, Scavenger class traffic is dropped the most aggressively when network congestion occurs.

DoS and worm attacks that flood the network with unwanted traffic can be mitigated by profiling applications to determine what constitutes legitimate normal versus abnormal flows and policing and re-marking traffic that exceeds this profile to Scavenger class service.

Classification Challenges of Video and Wireless Traffic

The newer types of network traffic including video traffic and traffic arriving via wireless edge technologies present additional challenges in classification compared to older types of traffic or wired technologies.

Video traffic comes in a wide array of different traffic types belonging to applications that may be extremely high priority and delay sensitive (such as immersive Cisco TelePresence traffic) to unwanted Scavenger class traffic (nonorganization entertainment videos, such as YouTube) that in many cases may be dropped outright.

Based on RFC 4594, there are generally four recognized types of video traffic (which can be grouped as interactive streaming):

- **Interactive video:**
    - **Real-time interactive:** High-definition interactive video applications (for example, Cisco TelePresence)
    - **Multimedia conferencing:** Desktop software multimedia collaboration applications (for example, Cisco Unified Personal Communicator)
- **Streaming video:**
    - **Broadcast video:** Broadcast IPTV, live events (for example IP video surveillance and digital signage applications)
    - **Multimedia streaming:** Video-on-demand (VoD) streaming flows, which can be buffered and are therefore neither delay not jitter sensitive (for example, e-learning videos and video meeting replays)

Traffic arriving over wireless access offers a different set of challenges in that wireless devices roam and the network must ideally provide the “same” class of service to the same end user regardless of the user’s current location. This QoS requirement interacts with the caching of wireless credentials (needed for fast roaming and delivery of real-time content to the device), and the IEEE 802.11 specification, which provides a means for wireless devices to request traffic in different access categories with different markings (which are usually on the untrusted side of the network trust boundary and so raises the question of whether the trust boundary for wireless devices could, or should, be extended to the wireless device under certain circumstances).

The WiFi Multimedia (WMM) and IEEE 802.11e standard classifications (the original 802.11e wireless standard has since been rolled into the broader 802.11 Ethernet standard) are different from the classifications recommended and used in the Cisco network, which are based on IETF recommendations. The wireless QoS profiles for these devices (and user devices and access points [APs]) must be mapped into the QoS classes and settings of the network's QoS design strategy. In Cisco's current wireless model, every traffic type from a wireless device is put into its own service set identifier (SSID), which is mapped to a VLAN. The SSID determines the handling of QoS. Wireless traffic is also usually encrypted and tunneled between the AP and he controller, making traditional QoS marking fields in the header unusable.

Marking Fields in Different Technologies

As introduced in the "Packet Headers" section in Chapter 1, there are various L2 and L3 frame and packet header fields used for marking traffic. In addition there are tunnel headers that encapsulate IP packets, including generic routing encapsulation (GRE), Multiprotocol Label Switching (MPLS), and Internet Protocol Security (IPsec) or other forms of virtual private network (VPN) tunnels. There is also the Control and Provisioning of Wireless Access Points (CAPWAP) tunneling protocol commonly used for wireless transmissions. Each of these fields and technologies is covered in this section.

L3 packet marking with IP precedence and DSCPs is the most widely deployed marking option because L3 packet markings have end-to-end network significance and easily can be translated to and from the L2 frame markings.

Field Values and Interpretation

Fields using a single bit value, such as Asynchronous Transfer Mode (ATM) and Frame Relay (FR), provide simply an indication of relative drop priority (yes or no) among the frames. Fields using a 3-bit width can encode up to 8 classes of service. The best granularity is offered by using a 6-bit field width, where up to 64 classes of service can be defined. Table 3-1 provides a summary of the technologies and field widths.

Table 3-1 *L2 and L3 Marking Field Summary*

Technology	Layer	Marking Field	Field Width (Bits)	Value Range
Ethernet (802.1 Q/p)	2	CoS	3	0 to 7
802.11 WiFi	2	TID	3	0 to 7

Technology	Layer	Marking Field	Field Width (Bits)	Value Range
Frame Relay (FR)	2	DE bit*	1	0 to 1
ATM	2	CLP bit*	1	0 to 1
MPLS	2	EXP*	3	0 to 7
IPv4 and IPv6	3	IP Precedence	3	0 to 7
IPv4 and IPv6	3	DSCP	6	0 to 63

* DE (Discard Eligible), CLP (Cell Loss Priority), EXP (Experimental)

The Ethernet CoS and L3 IP ToS IP Precedence fields are encoded as given in Table 3-2.

Table 3-2 *CoS/IP Precedence Values by Application Types*

CoS Value	Application
7	Reserved
6	Reserved
5	Voice
4	Video conferencing
3	Call signaling
2	High-priority data
1	Medium-priority data
0	Best-effort data

The 6-bit DSCP fields used in IPv4 and IPv6 headers are encoded as given in Figure 3-1. DSCP values can be expressed in numeric form or by special keyword names, called per-hop behaviors (PHBs). Three defined classes of DSCP PHBs exist: Best-Effort (BE or DSCP 0), Assured Forwarding (AFxy), and Expedited Forwarding (EF). In addition to these three defined PHBs, Class-Selector (CSx) code points have been defined to be backward compatible with IP precedence. (In other words, CS1 through CS7 are identical to IP precedence values 1 through 7.) The RFCs describing these PHBs are 2547, 2597, and 3246.

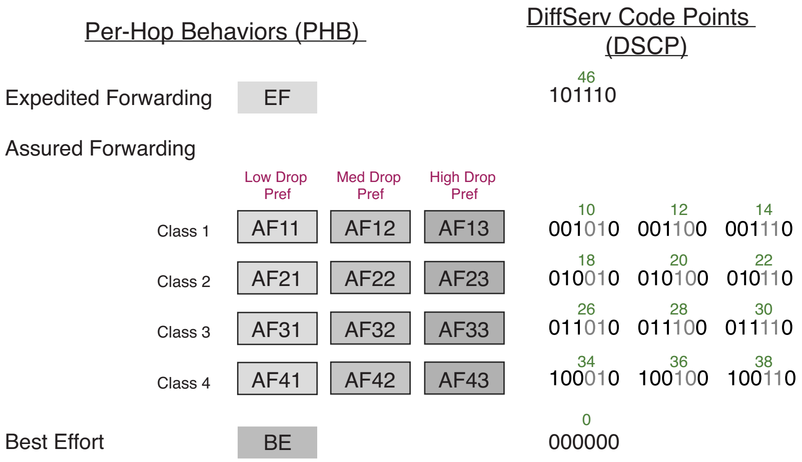


Figure 3-1 *DSCP Encoding Scheme*

RFC 2597 defines four Assured Forwarding classes, denoted by the letters AF followed by two digits. The first digit denotes the AF class and can range from 1 through 4. The second digit refers to the level of drop preference within each AF class and can range from 1 (lowest drop preference) to 3 (highest drop preference). For example, during periods of congestion (on an RFC 2597-compliant node), AF33 would statistically be dropped more often than AF32, which, in turn, would be dropped more often than AF31. Figure 3-2 shows the AF PHB encoding scheme.

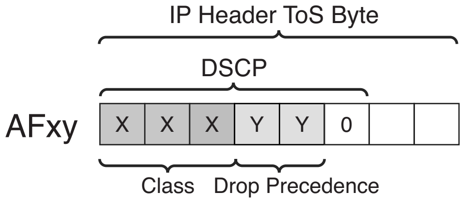


Figure 3-2 *DSCP Assured Forwarding Encoding Scheme*

Ethernet 802.1Q/p

Ethernet frames can be marked with their relative importance at Layer 2 by setting the 802.1p user priority bits (CoS) of the 802.1Q header, as shown in Figure 3-3.

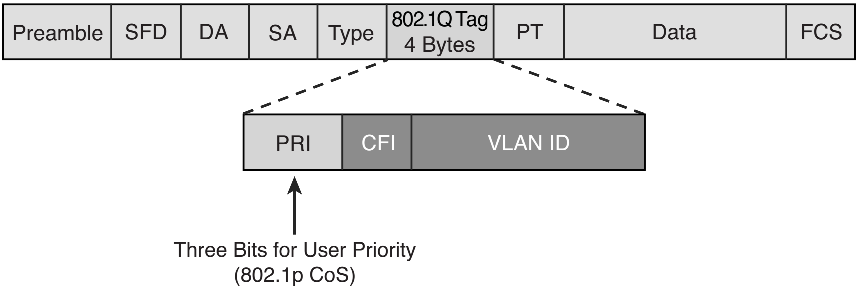


Figure 3-3 *Ethernet Frame: 802.1Q/p CoS Field*

Ethernet 802.11 WiFi

Wireless Ethernet frames can be marked at Layer 2 by setting the 802.11 WiFi Traffic Identifier (TID) field with in the QoS Control field of the 802.11 WiFi MAC header, as shown in Figure 3-4.

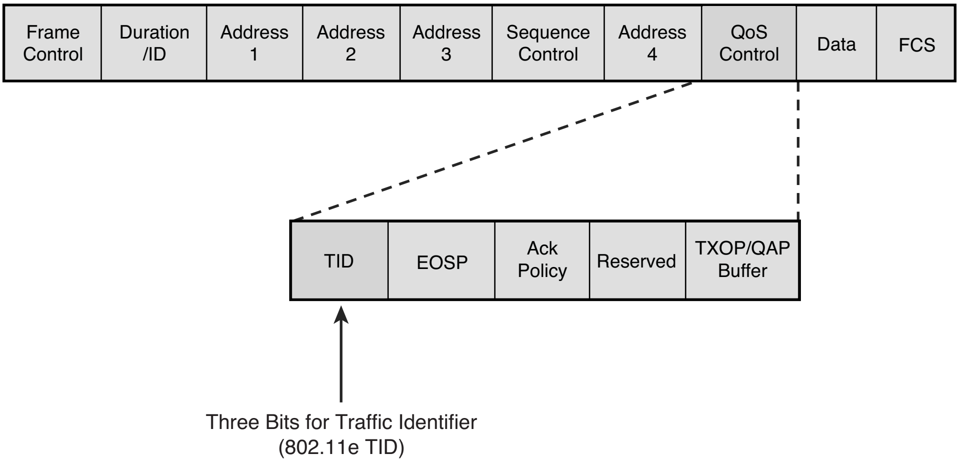


Figure 3-4 *802.11 WiFi MAC Frame QoS Control Field*

ATM and FR

ATM and FR are covered here purely for historical perspective. These are old technologies, and although they are widely deployed, they are seldom or never being installed in new networks.

The ATM CLP bit is a binary field with two values: 0 (the default), which indicates higher-priority traffic; and 1, for cells carrying lower-priority traffic eligible to be dropped if congestion is encountered. Example 3-1 shows how the ATM CLP bit can be set with class-based marking.

Example 3-1 *Setting the ATM CLP Bit*

```
Router# show run
policy-map SET-ATM-CLP
  class OUT-OF-SLA
    set atm-clp
```

The Frame Relay DE bit is defined and used exactly as the ATM CLP bit. Example 3-2 shows how the Frame Relay DE bit can be set inside a service policy.

Example 3-2 *Setting the Frame Relay DE Bit*

```
Router# show run
policy-map SET-FR-DE
  class OUT-OF-SLA
    set fr-de
```

IPv4 and IPv6

The IPv4 packet carries an 8-bit Type of Service (ToS) byte, while the IPv6 header carries an 8-bit Traffic Class field. The first 3 bits of both these fields are the IP Precedence (IPP) bits. These 3 three bits combined with the next three bits are known collectively as the DiffServ Code Point (DSCP) bits.

The IPv4 and IPv6 packet headers were given in Figure 1-3 in Chapter 1. These fields (CS or DSCP) can be set as shown in Example 3-3.

Example 3-3 *Setting the IP Precedence or DSCP Field in IPv4 or IPv6 Headers*

```
Router# show run
policy-map SET-DSCP
  class DSCP-AF31
    set dscp af31
```

L2 and L3 Tunnels

Cisco routers offer a variety of tunneling features, such as GRE, IPsec, and Layer 2 Tunneling Protocol (L2TP), which enable service providers to provide L3 VPN tunnels by enveloping one IP packet within another. Such encapsulation envelopes the original header information in an outer header to provide address preservation across one or more intervening networks. Additional features such as encryption may also be provided by the tunneling technology, and this provides privacy of the original packet in additional to address preservation.

Ethernet over IP tunnels (EoIP), described in RFC 3378, allows the tunneling of non-IP protocols over an IP network. EoIP tunnels are essentially a L2TP tunnel used in wireless deployments.

Figure 3-5 shows some example packet header layouts of GRE, IPsec, and EoIP tunnel packets.

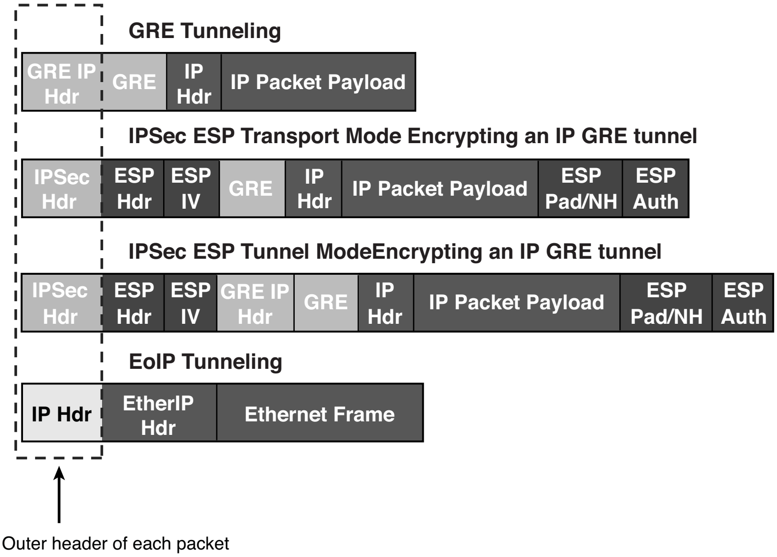


Figure 3-5 *L3 Tunnel Packet Layout Examples*

The marking field from the inner header might or might not be copied automatically to the outer header. If not, explicit CLI must be used to mark the outer header. Methods to achieve this include the **qos pre-classify** CLI on the tunnel interface (GRE and L2TP) or on the crypto map (for IPsec). The **l2tp tos reflect** CLI can also be used on L2TP tunnels. L2TPv3 is widely used to transport L2 frames over IP networks.

CAPWAP

CAPWAP is an IEEE protocol specified in RFC 5415 that enables a wireless controller to manage a group of wireless access points. Based on the Lightweight Access Point Protocol (LWAPP), it includes Datagram Transport Layer Security (DTLS) tunnel establishment and provides for configuration and device management of wireless APs. As a tunneling protocol, it includes an outer IP header with a DSCP field for QoS marking of the packet.

MPLS

MPLS is a tunneling technology that inserts a 4-byte label header onto an IP packet. An MPLS label contains a 3-bit marking field for CoS referred to as the MPLS EXP (Experimental) bits, as shown in Figure 3-6.

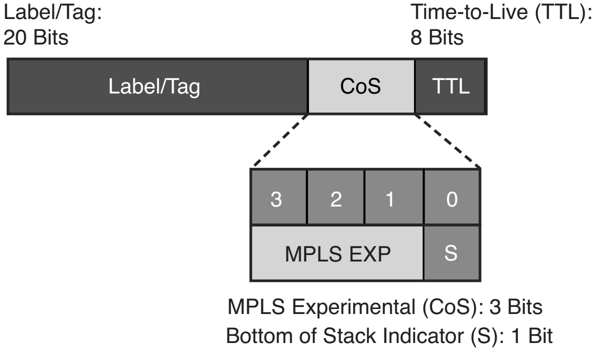


Figure 3-6 *MPLS EXP Bits Within an MPLS Label*

The possible values of the MPLS EXP bits are the same as those for 802.1Q/p CoS and IP precedence and can be mapped to 6-bit DSCP values in the same manner. MPLS EXP bits can be manipulated with the class map **match** and **set** commands.

In MPLS tunneling scenarios, there can be multiple MPLS headers on a packet. The **set mpls experimental imposition** command sets a value on all labels on the packet, and the **set mpls experimental topmost** command sets a specific value only on the outermost label.

Mapping QoS Markings

There are several places where re-marking of traffic is typically necessary, including the following:

- A L2 to L3 network boundary
- A Cisco to RFC 4594 network boundary
- A wireless-to-wired network boundary

Mapping L2 to L3 Markings

L2 CoS or L3 IPP values generally constitute the 3 most significant bits of the equivalent 6-bit DSCP value, therefore mapping directly to the Code Selector (CS) points defined by the DiffServ RFCs.

For example, CoS 5 (binary 101) maps to DSCP 40 (binary 101000). Using the layout given in Figure 3-2, the mapping is formed by replacing the XXX value in the figure with the CoS value, while the YY value remains 0. Table 3-3 shows the mappings between CoS, CS, and DSCP values. The same mapping in Table 3-3 can also be used to map L3 IPP to L3 DSCP.

Table 3-3 *L2 CoS to L3 Class Selector/DSCP Mappings*

802.1p CoS Value	CoS Binary Equivalent	Class Selector	CS Binary Equivalent	DSCP Value (Decimal)
0	000	CS0*/DF	000 000	0
1	001	CS1	001 000	8
2	010	CS2	010 000	16
3	011	CS3	011 000	24
4	100	CS4	100 000	32
5	101	CS5	101 000	40
6	110	CS6	110 000	48
7	111	CS7	111 000	56

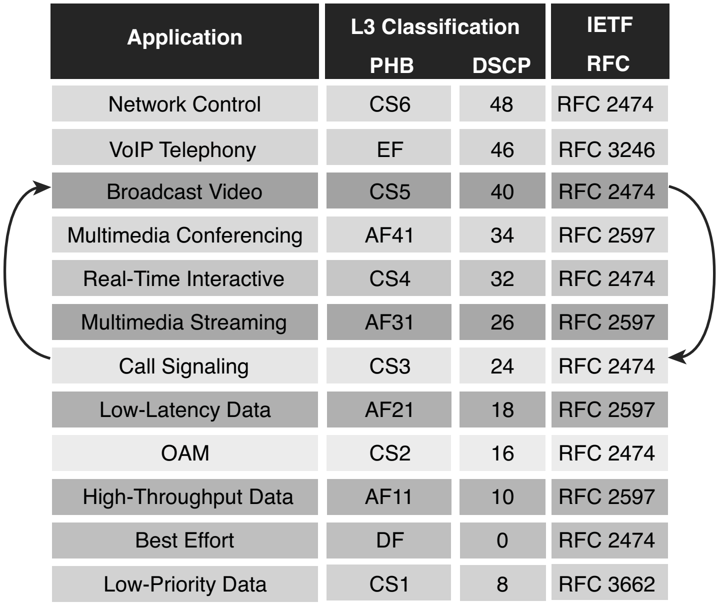
* Class Selector 0 is a special case; it represents the default marking value (defined in RFC 2474-Section 4.1). Therefore, it is not usually called Class Selector 0, but rather Default Forwarding or DF.

Mapping Cisco to RFC 4594 Markings

Cisco QoS marking recommendations follow RFC 4594, with the single exception that the marking values for Call Signaling (Cisco=CS3, RFC4594=CS5) and Broadcast Video (Cisco=CS5, RFC4594=CS3) are swapped, as shown in Figure 3-7.

Note The reason for this marking swap is explained in Chapter 9, "Application Visibility Control (AVC)."

If you are interfacing a Cisco network with a network where traffic is marked strictly in accordance with RFC 4594, you may have to re-mark the Broadcast Video and Call Signaling traffic classes at the network boundary.



Application	L3 Classification		IETF
	PHB	DSCP	RFC
Network Control	CS6	48	RFC 2474
VoIP Telephony	EF	46	RFC 3246
Broadcast Video	CS5	40	RFC 2474
Multimedia Conferencing	AF41	34	RFC 2597
Real-Time Interactive	CS4	32	RFC 2474
Multimedia Streaming	AF31	26	RFC 2597
Call Signaling	CS3	24	RFC 2474
Low-Latency Data	AF21	18	RFC 2597
OAM	CS2	16	RFC 2474
High-Throughput Data	AF11	10	RFC 2597
Best Effort	DF	0	RFC 2474
Low-Priority Data	CS1	8	RFC 3662

Figure 3-7 *Cisco-Modified RFC 4594-Based Marking Values*

Mapping Markings for Wireless Networks

Cisco Unified Wireless products support WiFi MultiMedia (WMM), a QoS system based on the IEEE 802.11e (an Institute of Electrical and Electronics Engineers standard) and published by the WiFi Alliance. The 802.11e amendment for wireless has since been rolled into the broader 802.11 Ethernet standard and is commonly referred to as 802.11 WiFi. The IEEE 802.11 WiFi classifications are different from how Cisco wireless technology deals with classification (based on IETF RFC 4594), as shown in Table 3-4. The primary difference in classification is the changing of voice and video traffic to CoS 5 and 4, respectively (from 6 and 5 used by the IEEE 802.11 WiFi).

Table 3-4 *IEEE 802.11 WiFi, WMM, and DSCP Mappings*

802.1 Traffic Type	Cisco DSCP	WMM	IEEE 802.11 WiFi
Network control	48	7	-
Internetwork control (CAPWAP control, 802.11 WiFi management)	48	6	7
Voice	46 (EF)	5	6
Video	34 (AF41)	4	5
Voice control	26 (AF31)	3	4

802.1 Traffic Type	Cisco DSCP	WMM	IEEE 802.11 WiFi
Background (gold)	18 (AF21)	2	2
Background (gold)	20 (AF22)	2	2
Background (gold)	22 (AF23)	2	2
Background (silver)	10 (AF11)	1	1
Background (silver)	12 (AF12)	1	1
Background (silver)	14 (AF13)	1	1
Best effort	0 (BE)	0	0, 3
Background	2	0	1
Background	4	0	1
Background	6	0	1

To comply with both standards, the Cisco Unified Wireless solution performs a conversion between packet markings when the traffic crosses the wireless-wired boundary. WMM voice traffic arrives with a CoS of 6 at the wireless controller or AP, and the controller or AP automatically performs a CoS-to-DSCP mapping. With the Unified Wireless Network, you should always think in terms of IEEE 802.11 WiFi classification and allow the Cisco Unified Wireless Network Solution to convert between IEEE classification and the Cisco QoS recommendations.

Classification Tools

Classification of traffic determines what class of traffic packets or frames in a flow belong to. Only after traffic is positively identified can policies be applied to it. Therefore, best-practice design recommendations are to identify and mark traffic at the trust boundary, and this boundary should be as close to the source of the traffic as possible, usually in the wiring closet or within the trusted devices (such as IP phones or wireless controllers) themselves. If markings and trusts are set correctly, the intermediate hops do not have to repeat the same in-depth classification. Instead, they can administer QoS policies (such as scheduling) based on the previously set markings, which is lightweight compared to the processor-intensive deep packet inspection to determine the type of traffic from the flow.

As covered in Chapter 2, "IOS-Based QoS Architectural Framework and Syntax Structure," there are three general ways to classify traffic, the first two using header information, the third using the packet payload:

- **Markings:** Examine the L2 (CoS) or L3 (IPP or DSCP) settings
- **Addressing:** Examine the source/destination port, interface, or address (L2 or L3 addressing)

- **Application signatures:** Examine application content inside the packet payload, also known as deep packet inspection

Class-Based Classification (Class Maps)

Within the MQC structure, packet classification is done using the class map construct already introduced in Chapter 2. Specifically, the **match** CLI within a class map.

Match statements of various combinations (logical AND, OR, and NOT) can be used to achieve very specific and fine-tuned classification criteria. In summary, the following types of criteria can be specified in **match** statements to build up the specifications of traffic classification:

- **Packet header markings:**
  - **match atm-clp:** Matches ATM Cell-Loss Priority (CLP) bit
  - **match cos:** Matches 802.1Q/p CoS values
  - **match discard-class:** Matches a discard class setting
  - **match dscp:** Matches IPv4 and IPv6 DSCP values
  - **match fr-de:** Matches FR Discard Eligible (DE) bit
  - **match ip dscp:** Matches IPv4 DSCP values
  - **match ip precedence:** Matches IPv4 IP precedence values
  - **match mpls experimental:** Matches MPLS Experimental (EXP) bits
  - **match precedence:** Matches IPv4 and IPv6 IP precedence values
- **Packet attributes, characteristics, or field values:**
  - **match access-group:** Matches an access control list (ACL)
  - **match class-map:** Matches nested "match" statements in another class map
  - **match field:** Matches a field in a protocol header description file (PHDF)
  - **match flow pdp:** Matches a Packet Data Protocol (PDP) flow
  - **match group-object:** Matches a user in the source and destination security group
  - **match packet length:** Matches L3 packet length in the IP header
  - **match qos-group:** Matches a QoS group value
  - **match start:** Matches a value starting at a specific place in! the header
  - **match tag:** Matches a tag type in the header
- **Protocols:**
  - **match application:** Matches metadata application values

  - **match atm:** Matches ATM control traffic
  - **match protocol:** Matches the Network Based Application Recognition (NBAR) protocol

- **Addressing information:**
  - **match destination-address:** Matches destination MAC address
  - **match source-address:** Matches source MAC address
- **Logical or Physical interface:**
  - **match atm-vci:** Matches an ATM virtual circuit
  - **match fr-dlci:** Matches a FR virtual circuit
  - **match input vlan:** Matches incoming packets on a specific Ethernet virtual LAN
  - **match input-interface:** Matches incoming packets on a specific interface
  - **match vlan:** Matches all traffic on an Ethernet virtual LAN
- **Ports:**
  - **match ip rtp:** Matches traffic on an RTP port
  - **match port-type:** Matches routed or switched ports

Logical combinations of match criteria can be specified in the Modular QoS command-line interface (MQC) syntax to define a class as the logical OR or AND of several individual match criteria. as shown in Example 3-4.

Example 3-4 *match-any and match-all Example*

```
Router# show run
class-map match-any TRAFFICTYPE1
  match <criteria1>
  match <criteria2>
class-map match-all TRAFFICTYPE2
  match <criteria3>
  match <criteria4>
class-map TRAFFICTYPE3
  match not <criteria5>
class-map DETAILS
  match <criteria6>
class-map HIGHER-LEVEL
  match class-map DETAILS
  match <criteria7>
```

In the example, **class TRAFFICTYPE1** defines a class of packets matching any one, or more, of criteria1 or criteria2. The CLI **class TRAFFICTYPE2** defines a class of packets that match each, and all, of criteria3 and criteria4. If a class map is defined without an explicit **match-any** or **match-all** keyword, the default treatment is **match-all.**

For a **match** statement that contains multiple values in the statement itself, such as the **match dscp af21 af22 af23** statement shown in Example 2-2 in Chapter 2 (in the "MQC Syntax" section), a packet has to match only a single value in the list to satisfy the statement as a whole and therefore to qualify for inclusion in the class.

Further, it is possible to define a class as matching none of one, or a set of, criteria. The CLI **class TRAFFICTYPE3** in Example 3-4 defines a class of packets that all packets satisfy except packets that match criteria5.

It is also possible to nest class definitions by including previously defined classes as the match criteria for another class. Example 3-4 shows the class **HIGHER-LEVEL** including traffic that matched by the nested class called **DETAILS.**

Network-Based Application Recognition

NBAR is a L4–L7 deep-packet inspection classifier triggered by the **match protocol** command within a **class-map** definition. It is a more CPU-intensive than classifiers that match traffic by markings (DSCPs), addresses, or ACLs.

The majority of data applications can be identified using L3 or L4 criteria (such as discrete IP addresses or well-known TCP/UDP ports), but not all applications can be identified by these criteria alone. For example, some peer-to-peer media-sharing applications deliberately negotiate ports dynamically with the objective of penetrating firewalls.

When L3 or L4 parameters are insufficient to identify positively an application, NBAR can recognize packets by examining the data payload of stateless protocols and identifying application layer protocols by matching them against a Protocol Description Language Module (PDLM), which is essentially an application signature. PDLM definitions are modular, and new ones can be added to a system without requiring a Cisco IOS upgrade.

Example 3-5 shows the two modes of operation that NBAR offers:

- **Passive mode:** Discovers and provides real-time statistics on applications per interface or protocol and gives bidirectional statistics such as bit rate (bps), packet, and byte counts
- **Active mode:** Classifies applications for the purpose of marking the traffic so that QoS policies can be applied.

Example 3-5 *NBAR Example*

```
! NBAR used in application discovery mode
Router# show run
interface fastethernet 0/0
  ip nbar protocol-discovery

! NBAR used as a classifier
Router# show run
class-map match-any MY-VIDEO
  match protocol cuseeme
  match protocol h323
  match protocol rtp video
```

NBAR2 (the most recent version of the NBAR tool) can classify a very large set of stateless and stateful protocols, including non-TCP and non-UDP IP protocols, protocols using statically or dynamically assigned TCP and UDP port numbers, protocols using dynamically assigned TCP, and UDP port numbers and subport classification.

NBAR Protocols

Cisco Product Bulletin 627831, *NBAR2 Protocol Library*, provides a comprehensive list of protocols (http://cisco.com/go/qos > Network Based Application Recognition (NBAR) > Data Sheets and Literature > Bulletins).

In summary, the categories of protocols that NBAR can inspect include the following:

- Browsing protocols such as HTTP and Gopher
- Business and productivity protocols such as Citrix and ClearCase
- Email protocols such as Internet Message Access Protocol (IMAP) and Messaging Application Programming Interface (MAPI)
- File-sharing protocols such as Common Internet File System (CIFS) and File Transfer Protocol (FTP)
- Gaming protocols such as Doom
- Industrial protocols such as DICOM and OPC-Job-Start
- Instant messaging protocols such as gTalk and ICQ
- Internet privacy protocols such as IPsec and Secure Sockets Layer (SSL)
- L3-over-IP protocols such as Argus and AX25
- Network administration protocols such as sshell and SunRPC
- Voice and video protocols such as appleqtc and CoolTalk

Example 3-6 shows the CLI of some NBAR classification configurations.

Example 3-6 *NBAR Example*

```
Router# show run
class-map match-any ERP
   match protocol sqlnet
   match protocol ftp
   match protocol telnet
class-map match-any AUDIO-VIDEO
   match protocol http mime "*/audio/*"
   match protocol http mime "*/video/*"
class-map match-any WEB-IMAGES
   match protocol http url "*.gif"
   match protocol http url "*.jpg|*.jpeg"
```

Example 3-6 defines three different class maps. The first one, the class map ERP, instructs the classifier (NBAR) to pick traffic of any of the protocols listed in the subsequent statements. In the class map AUDIO-VIDEO, NBAR is looking for MIME traffic of particular types—audio and video in this case. The WEB-IMAGES class map is filtering out HTTP traffic for picture (GIF or JPEG) content.

RTP Traffic

Voice and video traffic can usually be classified by easier means than deep packet protocol inspection, and NBAR can also be used to classify this traffic if required. For example, **match protocol h323** identifies all H.323 voice traffic. The syntax **match protocol rtp** [**audio** | **video** | **payload-type payload-string**] can be used to classify traffic based on Real-time Transfer Protocol (RTP_ payload values if the desire is to classify based on codec types. The **audio** keyword matches by RTP payload-type values 0–23, the **video** keyword matches values 24–33, and the **payload-type** keyword matches by specific binary, decimal, or hexadecimal value for more granular matching.

Performance Routing

The Performance Routing (PfR) feature can also be used in conjunction with NBAR to classify and profile an application for PfR, as shown in Example 3-7. The resulting traffic classes are added to the PfR application database to be passively and actively monitored.

Example 3-7 *PfR and NBAR Example*

```
Router# show run
ip prefix-list LIST1 permit 10.2.1.0/24
ip prefix-list LIST1 permit 10.2.2.0/24
ip prefix-list LIST1 permit 172.17.1.0/24
```

```
pfr-map APP_NBAR_MAP 10
  match traffic-class application nbar rtp-audio prefix-list LIST1
```

Metadata Classification

A recent addition to the deep packet inspection capabilities is to classify traffic by metadata inserted into the packet by certain applications, in particular video applications. Metadata is "data about data" and is, in the context of QoS, fields added to the packet with descriptive or type information about the content of the rest of the packet. The metadata in a packet can (if trusted) be used for classification of the traffic flow. Metadata is discussed further in Chapter 8, "Medianet."

Marking Tools

Marking of traffic entails writing a value in the frame or packet header to document the class of traffic the packet has been determined (by classification) to belong to. Marking is done in conjunction with classification to ensure that a packet does not have to be reclassified at every node. Once a packet is classified and marked at the trust boundary of the network, subsequent nodes can treat the packet purely based on the packet marking (provided the intermediate nodes are configured to trust the markings, if not, they re-mark the packet to DSCP 0).

The main marking tools are class-based marking and marking as part of class-based policing. Some legacy marking techniques include committed access rate (CAR) and policy-based routing (PBR).

In Cisco voice and video products, marking is often done in the end device (for example, by the IP phones, voice gateways, and TelePresence devices). End-device markings are often not trusted because the end user has access to these devices and traffic from these devices is re-marked at the network trust boundary. However, a network administrator-owned device such as a voice gateway is often deployed within the trust boundary of the network, and end-device markings from gateways are usually trusted. The end device may also be considered a "trusted" device (and therefore technically exist within the trust boundary) if the markings applied by the device are centrally controlled and downloaded by a server or management application under control of the administrator.

Class-Based Marking (Class Maps)

Within the MQC structure, packet marking is done using the *class map* construct already introduced in Chapter 2. Specifically, the **set** CLI within a class map is used to write a marking into a packet. Example 3-8 shows the general syntax.

Example 3-8 *set Command Example*

```
Router(config)# policy-map CB-MARKING
Router(config-pmap)# class FOO
Router(config-pmap-c)# set ?
  atm-clp      Set ATM CLP bit to 1
  cos          Set IEEE 802.1Q/ISL class of service
  dscp         Set DSCP in IP(v4)and IPv6 packets
  mpls         Set MPLS specific values
  precedence   Set precedence in IP(v4) and IPv6 packets
  qos-group    Set QoS Group
```

In summary, the following types of markings can be controlled with **set** statements to write various values in frame and packet header fields:

- ToS values:
  - set cos
  - set dscp
  - set dscp cos
  - set dscp qos-group
  - set precedence
  - set qos-group
- Packet discard eligibility:
  - set atm-clp
  - set fr-de
- Tunnel ToS values:
  - set cos inner
  - set discard-class
  - set dscp tunnel
  - set precedence tunnel
  - set ip dscp tunnel
  - set ip precedence tunnel
  - set mpls experimental
  - set vlan-inner

Note The **set dscp** command applies to IPv4 and IPv6 packets, whereas the **set ip dscp** command applies only to IPv4 packets. In most cases, you should use the **set dscp** and **set precedence** forms of the commands.

Effects of Feature Sequence

Class-based marking occurs *after* classification of the packet (in other words, **set** happens after the match criteria). Therefore, if used on an output policy, the packet marking applied can be used by the next-hop node to classify the packet but cannot be used on *this* node for classification purposes. However, if class-based marking is used on an ingress interface as an input policy, the marking applied to the packet can be used on the same device on its egress interface for classification purposes.

On output policies both classification and marking can happen before or after tunnel encapsulation, depending on where the service policy is attached. Therefore, if a policy is attached to a GRE or IPsec tunnel interface, the marking is applied to the original inner packet header. However, if the policy is attached to the physical interface, only the tunnel header (the outer header) is marked, and the inner packet header is left unchanged.

Mapping Markings with the Table Map Feature

The **set** CLI can also be used to translate various QoS markings, including the following:

- CoS
- IPP
- DSCP
- QoS group
- MPLS EXP

You can build a conversion table with the **table-map** CLI and then reference the table in a **set** command to do the translation, as shown in Example 3-9. In this example, the DSCP value will be set according to the CoS value defined in the table map called map1. Any values not explicitly defined in a "to-from" relationship are set to the default value. If the default value is omitted from the table, the content of the packet header is left unchanged.

Example 3-9 *Mapping QoS Values*

```
Router# show run
table-map MAP1
  map from 0 to 0
  map from 2 to 1
  default 3
```

```
!
policy-map POLICY1
  class traffic1
    set dscp cos table MAP1
```

Example 3-10 shows several **set** command examples using the table map feature to translate from one type of packet marking to another.

Example 3-10 *Examples of the Table Map Feature*

```
set precedence cos table TABLE-MAP-NAME
set dscp cos table TABLE-MAP-NAME
set cos precedence table TABLE-MAP-NAME
set cos dscp table TABLE-MAP-NAME
set qos-group precedence table TABLE-MAP-NAME
set qos-group dscp table TABLE-MAP-NAME
set mpls experimental topmost qos-group table TABLE-MAP-NAME
set mpls experimental topmost precedence table TABLE-MAP-NAME
set mpls experimental imposition dscp table TABLE-MAP-NAME
set qos-group mpls exp topmost table TABLE-MAP-NAME
set precedence qos-group table TABLE-MAP-NAME
set dscp qos-group table TABLE-MAP-NAME
```

Marking (or Re-Marking) with Policing

Policing and other rate-limiting tools (covered in more detail in Chapter 4, "Policing, Shaping, and Markdown Tools") offer another way to mark packets. Instead of marking every packet of a certain type with a particular value, a policer is generally used to re-mark (or drop) packets that violate a particular rate or service level agreement (SLA).

Class-based policing can set various marking fields (including IP Precedence, DSCP, and MPLS EXP) of a packet based on rate-limiting measurements, as shown in Example 3-11.

Example 3-11 *Re-Marking Options for the Class-Based Policer*

```
Router(config)# policy-map CB-POLICING
Router(config-pmap)# class FOO
Router(config-pmap-c)# police 8000 conform-action ?
  drop                 drop packet
  exceed-action        action when rate is within conform and
                       conform + exceed burst
  set-clp-transmit             set atm clp and send it
  set-discard-class-transmit   set discard-class and send it
  set-dscp-transmit            set dscp and send it
```

```
set-frde-transmit            set FR DE and send it
set-mpls-exp-imposition-transmit  set exp at tag imposition
                                  and send it
set-mpls-exp-topmost-transmit     set exp on topmost label
                                  and send it
set-prec-transmit            rewrite packet precedence
                             and send it
set-qos-transmit             set qos-group and send it
transmit                     transmit packet
```

AutoQoS Marking

AutoQoS Enterprise (discussed previously in the “AutoQoS” section of Chapter 2) configurations do automatic marking of traffic classes. You can override these if you want, but the default AutoQoS markings are given in Table 3-5.

Table 3-5 *AutoQoS for the Enterprise Feature Class Definitions*

AutoQoS Traffic Class	Traffic Type	DSCP Value
IP Routing	Network control traffic, such as routing protocols	CS6
Interactive Voice	Inactive voice-bearer traffic	EF
Interactive Video	Interactive video data traffic	AF41
Streaming Video	Streaming media traffic	CS4
Telephony Signaling	Telephony signaling and control traffic	CS3
Transactional/Interactive	Database applications transactional in nature	AF21
Network Management	Network management traffic	CS2
Bulk Data	Bulk data transfers, web traffic, general data service	AF11
Scavenger	Casual entertainment, rogue traffic (Traffic in this category is given less-than-best-effort treatment.)	CS1
Best Effort	Default class, all noncritical traffic, HTTP, all miscellaneous traffic	0

Recommendations and Guidelines

Implement the following recommendations and guidelines when classifying and marking traffic in your network:

- Classify and mark traffic as close to its source as technically and administratively feasible.
- Classification and marking can be done on ingress or egress, whereas other QoS actions such queuing and shaping are usually done on egress.
- Use an end-to-end DiffServ PHB model for packet marking in your network.
- Map less-granular marking fields (such as 3-bit Ethernet CoS and MPLS fields) to 6-bit DSCP values as close to the traffic source as technically and administratively feasible.
- Set a trust boundary and mark or re-mark traffic that enters the network from beyond the trust boundary (generally distrust devices and endpoints that have direct end user access).
- Follow standards-based DiffServ PHB markings to ensure interoperability with other service provider networks, enterprise networks, or when merging networks together.
- Use the **set dscp** and **set precedence** commands (marks all IP traffic) rather than **set ip dscp** and **set ip precedence** commands (marks IPv4 traffic only).
- Take feature sequencing into consideration when using tunnel interfaces to make sure that the inner or outer packet headers (or both) are marked as you intended.

Summary

This chapter examined classification and marking features and tools. Classification is the action of inspecting a packet (certain fields within the packet) to determine what type of packet or traffic it is. This determination is used to guide the treatment that the packet (and other packets of the same traffic type or stream) receives from the network.

Marking is the action of changing a field within the packet header to note the determination reached by the classifier. The various ways of doing packet marking at L2 and L3 up to L7 were illustrated, and ways to translate one type of marking to another were discussed.

The treatment of the packet, which is based on the classification and marking results, includes capabilities such as policing, shaping, and queuing. Policing and shaping are discussed in Chapter 4, “Policing, Shaping, and Markdown Tools,” and queuing is discussed in Chapter 5, “Congestion Management and Avoidance Tools.”

Further Reading

Classification and Marking

Enterprise QoS Solution Reference Network Design Guide (3.0) Overview: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoSIntro.html#wp46124

Enterprise Medianet Quality of Service Design 4.0 Overview: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html#wp60933

Classification Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_classn/configuration/15-0m/qos-classn-15-0m-book.html

Classification of Locally Sourced Packets: http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12sclocp.html

Marking: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_classn/configuration/15-0m/qos-classn-mrkg-ntwk-trfc.html

Class-Based Marking: http://www.cisco.com/en/US/docs/ios/12_1t/12_1t5/feature/guide/cbpmark2.html

Implementing Quality of Service Policies with DSCP: http://www.cisco.com/en/US/tech/tk543/tk757/technologies_tech_note09186a00800949f2.shtml

Classification, Policing, and Marking on LAC (L2TP Access Concentrator): http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/qos_on_lac_ps6350_TSD_Products_Configuration_Guide_Chapter.html

NBAR

General NBAR information: http://www.cisco.com/en/US/products/ps6616/products_ios_protocol_group_home.html

NBAR white papers: http://www.cisco.com/en/US/products/ps6616/prod_white_papers_list.html

NBAR Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_nbar/configuration/15-0m/qos-nbar-15-0m-book.html

NBAR2 Frequently Asked Questions (FAQ): http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6558/ps6616/qa_c67-697963.html

NBAR2 Protocol Library (Cisco Product Bulletin 627831): http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6558/ps6616/product_bulletin_c25-627831.html

Video QoS

QoS Requirements of Video: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoSIntro.html#wp46626

Wireless QoS

Cisco Unified Wireless QoS: http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob41dg/ch5_QoS.html

RFCs

RFC 3378: *EtherIP: Tunneling Ethernet Frames in IP Datagrams*

RFC 5415: *Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification*

RFC 5416: *Control and Provisioning of Wireless Access Points (CAPWAP) Protocol Binding for IEEE 802.11*

This page intentionally left blank

Chapter 4

Policing, Shaping, and Markdown Tools

This chapter covers the following topics:

- Terminology, recommendations, and differences (related to policing, shaping, and markdown tools)
- Policing tools and mechanisms
- Shaping tools and mechanisms
- Advanced policing topics (such as the single-rate and two-rate, three-color policers; and hierarchical, multi-action, color-aware, and percentage-based policing)

An important early step in implementing a quality of service (QoS) policy is to analyze and identify the traffic to be treated by some set of actions. Chapter 3, “Classification and Marking,” covered how traffic is identified into classes and then marked to document the classification. The next QoS task is to assign actions or policy treatments to these classes of traffic, including bandwidth assignments, policing, shaping, queuing, and dropping decisions.

This chapter explores policing and shaping tools, and to some extent dropping mechanisms. Chapter 5, “Congestion Management and Avoidance Tools,” provides additional discussion on dropping mechanisms.

Policing and Shaping Topics

Before looking at each of the specific policing and shaping tools in greater depth, an understanding of the following few general policing and shaping topics is necessary:

- Policing and shaping terminology
- Tail drop and random drop
- Security and QoS

Policing and Shaping Terminology

Policers and shapers are tools to identify and respond to traffic violations; they usually identify traffic violations in an identical manner, but differ in how they respond to the violations:

- **Policers** perform checks for traffic violations against a configured rate and take immediate prescribed actions, such as dropping or re-marking the excess traffic. Policers do not delay traffic; they merely check the traffic and take an action.
- **Shapers** are traffic-smoothing tools that work in conjunction with buffering mechanisms. The objective of a shaper is not to drop traffic, but to smooth out the peaks of traffic arrival so that it never exceeds the configured rate. Shapers usually are employed to meet service level agreements (SLAs). If the offered traffic momentarily spikes above the contracted rate, the excess traffic is buffered and delayed until the offered traffic once again dips below the defined rate. If the offered traffic is below the contracted rate, it is sent immediately.

Figure 4-1 illustrates the difference between policing and shaping. Both mechanisms measure traffic against a given traffic rate and are therefore classed as rate limiters, but they differ in how they treat the traffic in excess of the defined rate. Policers drop excess traffic, shapers delay excess traffic.

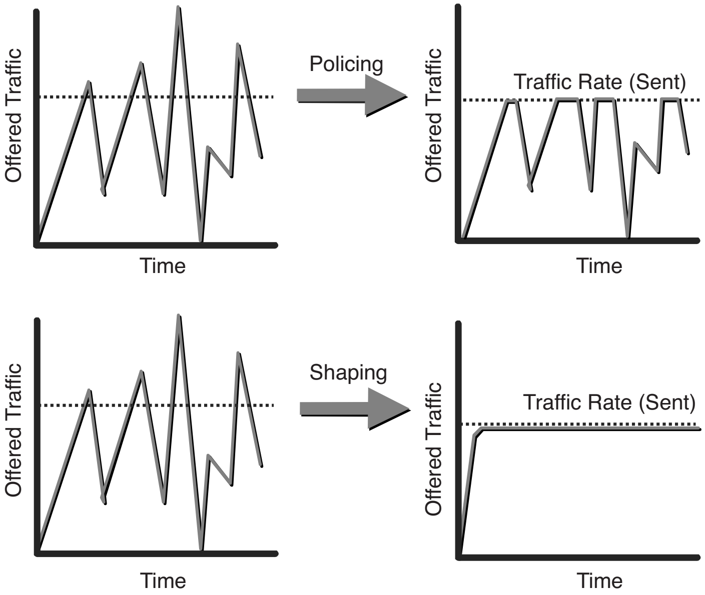


Figure 4-1 *Generic Policing Versus Shaping*

Table 4-1 compares the characteristics of policing and shaping tools.

Table 4-1 *Comparisons Between Policers and Shapers*

Policer	Shaper
Causes TCP resends when traffic is dropped	Typically delays (rather than drops) traffic; involves fewer TCP resends
Inflexible and inadaptable; makes instantaneous packet drop decisions	Can adapt to network congestion by queuing excess traffic
An ingress or egress interface tool	Typically an egress interface tool
Introduces no delay or jitter in a traffic stream	Introduces delay and jitter if arriving traffic exceeds the contracted (shaped) rate
Rate limiting without buffering	Rate-limiting with buffering

Placing Policers and Shapers in the Network

Because policers make instantaneous send/drop decisions, these are optimally deployed as ingress tools: if a packet is going to be dropped, you might as well drop it before spending valuable processing cycles on routing and handling. However, policers are also often deployed at egress to control bandwidth used (or allocated) to a particular class of traffic, because such a decision often cannot be made until the packets reach the egress interface.

Shapers, because of their dependency on buffers to smooth out spiking traffic flows for later transmission, are only deployed as egress tools. Shapers are commonly used on enterprise-to-service-provider links (on the enterprise egress side) to ensure that traffic destined for the service provider (SP) does not exceeds a contracted rate, because if it does, the SP policer (on ingress) is likely to drop this traffic to enforce the contracted rate on the link to the enterprise. In this case, policers are deployed for both ingress and egress traffic because it is easiest for the SP to check a contracted rate on the direct link between the SP's equipment (the provider edge) and the customer's equipment (enterprise edge).

Tail Drop and Random Drop

There are two main ways of dropping packets: tail drop or random drop.

A policer does tail drop, which describes an action that drops every packet that exceeds the given rate, until the traffic drops below the rate. Shapers strive not to drop packets at all, but do so if the traffic consistently exceeds the given rate and buffers overflow, in which case, these also employ tail drop.

Tail drop can have adverse effects on TCP retransmission methods and cause waves of traffic spikes in the network. Another mechanism of dropping packets is random dropping, which is employed by tools such as random early detection (RED) and weighted RED (WRED). These methods work more effectively with TCP retransmission logic, but they are not policing/shaping tools. RED and WRED are discussed in more detail in Chapter 5.

Re-Mark/Markdown

When a traffic rate is exceeded, a policer can take one of two actions:

- Drop the traffic
- Re-mark it to another class of service, usually a class that has a higher drop probability

Whenever feasible, re-marking (or markdown) should be done according to standards-based rules such as Request For Comments (RFC) 2597, *Assured Forwarding (AF) Per-Hop-Behavior (PHB) Group*. For example, excess traffic arriving as AFx1 should be marked down to AFx2 (or AFx3, whenever dual-rate policing—such as defined in RFC 2698—is supported). Following such markdowns, congestion management policies, such as differentiated services code point (DSCP)-based WRED, should be configured to drop AFx3 (statistically) more aggressively than AFx2, which in turn should be dropped (statistically) more aggressively than AFx1.

Traffic Types to Police and Shape

Regulating real-time traffic such as voice and video with policing (dropping) and shaping (delaying) is generally counterproductive. Call admission control (CAC) strategies should be used to prevent real-time traffic from exceeding the capacity of the network. Policing and shaping tools are best employed to regulate TCP-based data traffic.

Token Bucket Algorithms

Cisco IOS policers and shapers are modeled after token bucket algorithms, although they do not credit tokens in exactly the same way. A general and simplified explanation follows and does not necessarily strictly represent how each algorithm on each Cisco platform operates. There are many variations in the implementation details and across products and software releases.

Token bucket algorithms are metering engines that keep track of how much traffic can be sent to conform to a specified traffic rate. A token permits a single unit (usually a bit, but can be a byte) of traffic to be sent. Tokens are granted at the beginning of a specific time increment, usually every second, according to the specified rate called the committed information rate (CIR). The CIR is the access bit rate contracted in the SLA with a service provider.

For example, if the CIR is set to 8000 bps, 8000 tokens are placed in a bucket at the beginning of the time period. Each time a bit of traffic is offered to the policer, the bucket is checked for tokens. If there are tokens in the bucket, the traffic is viewed as *conforming* to the rate and the typical action is to send the traffic. One token is removed from the bucket for each bit of traffic passed. When the bucket runs out of tokens, any additional offered traffic is viewed to *exceed* the rate, and the exceed action is taken, which is typically either to re-mark or drop the traffic.

Note At the end of the second, there might be unused tokens. The handling of unused tokens is a key differentiator among policers. This is discussed in the "Types of Policers" section later in this chapter.

Because the interface clock rate cannot change to enforce CIR policy, the only way to impose a rate limit on an interface is to use time-division multiplexing (TDM). With TDM, when a rate limit (or CIR) is imposed on an interface, the traffic is allocated a subsecond time slice during which it can be sent. This subsecond time slice is referred to as the interval (or Tc). For example, if an 8-Kbps CIR is imposed on a 64-Kbps link, traffic can be sent for an interval of 125 ms (64,000 bps / 8000 bits).

The entire amount allowed by the CIR (8000 bits) could theoretically be sent at once, but then the algorithm would have to wait 875 ms before it could send any more data (to comply with the rate limit), causing excessive interpacket delays. Therefore, to smooth out the allowed flow over each second, the CIR is divided into smaller units, referred to as the committed burst (Bc), which is the sustained number of bits that can be sent per Tc interval. These smaller units are sent over multiple instances during a single second. Continuing with the previous example, if the Bc is set to 1000, each committed burst can take only 15.6 ms (1000 bits / 64,000 bps) to send traffic out the interface at the clock rate. The algorithm waits 109.4 ms (125 ms – 15.6 ms) and sends another 15.6 ms of data (1000 bits). This process is repeated a total of eight times during each second. The data transmission "bursts" comprising a Bc of 1000 (and therefore a Tc of 125ms) is illustrated in the bottom of Figure 4-2.

A packet or frame of 1500 bytes constitutes (1500 * 8) 12000 bits. At 1000 bits (the Bc value) for each time slice (the value of Tc, in this case 125ms), it takes 12 time slices to send the entire packet. Each time slice is 125 ms, so the entire packet takes 1.5 seconds to be sent in entirety (in small bursts of 100 bits each at line rate transmission speed).

Therefore, the token bucket algorithm is as follows: Bc = CIR * Tc (Bits = Rate * Time)

Cisco IOS Software does not allow the explicit definition of the interval (Tc). Instead, it takes the CIR and Bc values as arguments and derives the interval and the number of bursts per second. For example, if the CIR is 8000 and the Bc is set to 4000, two bursts occur per second (Tc = 500 ms). The resulting transmission of a Bc value of 4000 is illustrated in the middle part of Figure 4-2. If the Bc is set to 2000, four bursts occur per second (Tc = 250 ms). If the Bc is set to 1000, eight bursts occur per second (Tc = 125 ms). The resulting transmission of a Bc value of 8000 is illustrated in the top part of Figure 4-2.

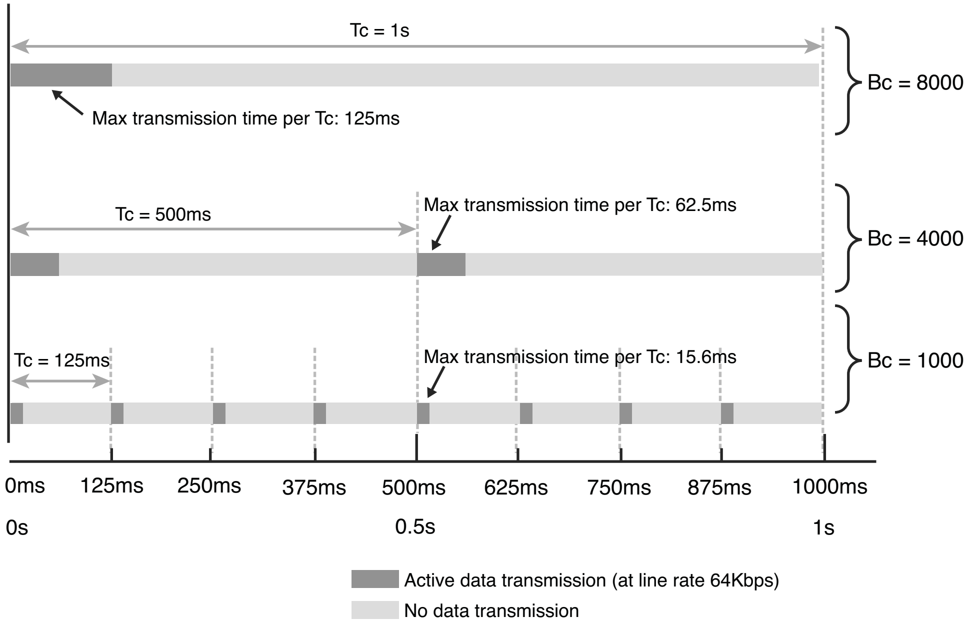

Tc = 1s
Bc = 8000
Max transmission time per Tc: 125ms
Tc = 500ms
Max transmission time per Tc: 62.5ms
Bc = 4000
Tc = 125ms
Max transmission time per Tc: 15.6ms
Bc = 1000
0ms
125ms
250ms
375ms
500ms
625ms
750ms
875ms
1000ms
0s
0.5s
1s
Active data transmission (at line rate 64Kbps)
No data transmission

Figure 4-2 *Token Bucket Algorithm*

The preceding examples illustrate the operation of the feature from a theoretical perspective. From a practical perspective, when implementing networks, Tc should not exceed 125 ms. Shorter intervals can be configured and are necessary to limit jitter in real-time traffic, but longer intervals are not practical for most networks because the interpacket delay becomes too large.

Types of Policers

There are different variations of policing algorithms, including the following:

- Single-rate two-color
- Single-rate three-color
- Dual-rate three-color

Single-Rate Two-Color Policers

The original policers implemented use a single-rate, two-color model with

- A single rate and single token bucket algorithm.
- Traffic identified as one of two states (or colors): *conforming* to or *exceeding* the CIR. Marking or dropping actions are performed on each of the two states of traffic.

This type of policer is fairly simple and is illustrated in the top portion of Figure 4-1, earlier in this chapter.

RFC 2697 Single-Rate Three-Color Policers

An improvement to the single-rate two-color policer algorithm is based on RFC 2697, which details the logic of a single-rate three-color model with

- A single rate and two token buckets algorithm
- Traffic identified as one of three states (or colors): *conforming* to, *exceeding* or *violating* the CIR. Marking or dropping actions are performed on each of the three states of traffic.

The first token bucket operates just like the single-rate two-color system. But if there are any tokens left over in the bucket after each time period, these are placed in the second bucket to be used as credits later for temporary bursts that might exceed the CIR. Tokens placed in this second bucket are called the excess burst (Be), and this number of tokens is placed in the bucket when Bc is not full. Be is the maximum number of bits that can exceed the Bc burst size.

With this two token-bucket mechanism, traffic can be identified in three states (or three colors) as follows:

- **Conform:** Traffic within the CIR—usually sent (optionally re-marked)
- **Exceed:** Traffic within the excess burst allowance above CIR—can be dropped, or re-marked and sent
- **Violate:** Traffic beyond the excess burst—usually dropped (optionally re-marked and transmitted)

The single-rate three-color marker/policer uses the following definitions parameters to meter the traffic stream:

- **CIR:** Committed information rate, the policed rate
- **CBS:** Committed burst size, the maximum size of the first token bucket
- **EBS:** Excess burst size, the maximum size of the second token bucket
- **Tc:** Token count of CBS, the instantaneous number of tokens left in the CBS bucket (Do not confuse the term Tc here with the earlier use of Tc in the context of time elapsed for policing or shaping traffic intervals.)
- **Te:** Token count of EBS, the instantaneous number of tokens left in the EBS bucket
- **B:** Byte size of offered packet

Figure 4-3 illustrates the logical flow of the single-rate three-color marker/policer two-token-bucket algorithm.

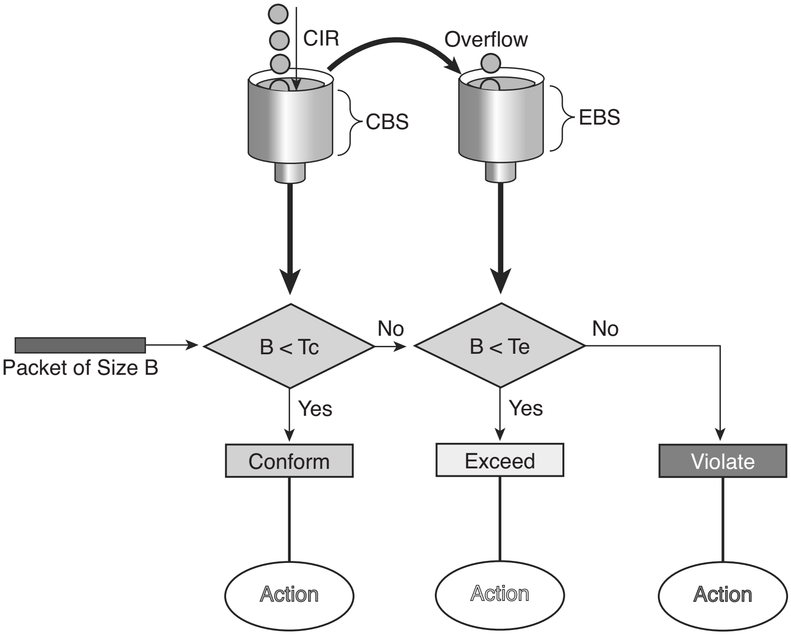


Figure 4-3 *RFC 2697 Single-Rate Three-Color Policer Logic*

The single-rate three-color policer's tolerance of temporary bursts results in fewer TCP retransmissions and is therefore more efficient for bandwidth utilization. It is a highly suitable tool for marking according to RFC 2597 AF classes, which have three "colors" (or drop preferences) defined per class (AFx1, AFx2, and AFx3). Using a three-color policer generally makes sense only if the actions taken for each color differ. If the actions for two or more colors are the same, a simpler policer (and therefore a simpler QoS policy) is more suitable to implement, making the network easier to maintain.

RFC 2698 Dual-Rate Three-Color Policers

The single-rate three-color marker/policer was a significant improvement for policers—it made allowance for temporary traffic bursts as long as the overall average transmitted rate was equal to or below the CIR. However, the variation in the number of accumulated excess burst credits could cause a degree of unpredictability in traffic flows. To improve on this, a two-rate three-color marker/policer was defined in RFC 2698. This policer addresses the peak information rate (PIR), which is unpredictable in the RFC 2697 model. Furthermore, the two-rate three-color marker/policer allows for a sustainable excess burst (negating the need to accumulate credits to accommodate temporary bursts) and allows for different actions for the traffic exceeding the different burst values.

The dual-rate three-color marker/policer uses the following definitions parameters to meter the traffic stream:

- **PIR:** Peak information rate, the maximum rate that traffic ever is allowed
- **PBS:** Peak burst size, the maximum size of the first token bucket
- **CIR:** Committed information rate, the policed rate
- **CBS:** Committed burst size, the maximum size of the second token bucket
- **Tp:** Token count of PBS, the instantaneous number of tokens left in the PBS bucket
- **Tc:** Token count of CBS, the instantaneous number of tokens left in the CBS bucket
- **B:** Byte size of offered packet

The two-rate three-color policer also uses an algorithm with two token buckets, but the logic varies slightly. Instead of transferring unused tokens from one bucket to another, this policer has two separate buckets that are filled each second with two separate token rates. The first bucket is filled with the PIR tokens and the second bucket is filled with the CIR tokens. In this model, the Be works the same as the Bc, except for the PBS bucket (not the CBS bucket). This means that Be represents the peak limit of traffic that can be sent during a subsecond interval.

The logic varies further in that the initial check is to see whether the traffic is within the PIR. Only then is the traffic compared against the CIR. In other words, a violate condition is checked for first, then an exceed condition, and finally a conform condition, which is the reverse of the logic of the previous model. Figure 4-4 illustrates this logic.

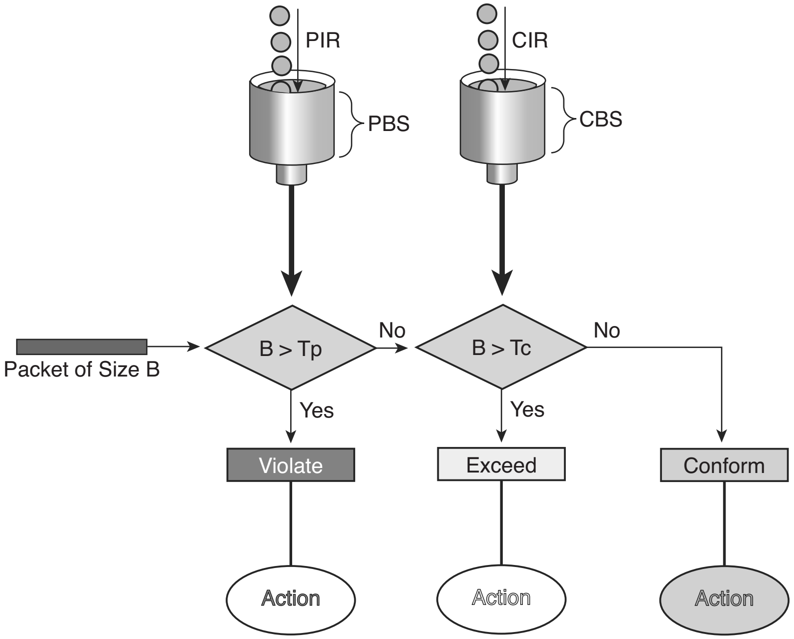


Figure 4-4 *RFC 2698 Dual-Rate Three-Color Policer Logic*

Security and QoS

QoS policers can serve purposes beyond traditional QoS policies. For instance, these can help improve the security posture of the network. One way they can do this by helping to mitigate denial-of-service (DoS) and worm network attacks. DoS attacks tend to flood the network with unwanted traffic. Worm attacks usually do not target the network infrastructure itself, but cause residual excess traffic as the worm exponentially self-propagates.

Therefore, a QoS strategy to improve network infrastructure security is to profile applications to determine what constitutes normal as opposed to abnormal flows. Threshold demarking of normal/abnormal flows vary and legitimate traffic flows often temporarily exceed traffic thresholds. However, sustained abnormal streams generated simultaneously by multiple hosts (highly indicative of DoS/worm attacks) should be subject to aggressive dropping after legitimate traffic has been serviced. Then, after these flows have been profiled, network edge policers can meter all flows and re-mark abnormal traffic to Scavenger (DSCP CS1) class. And finally, to complement these re-marking policies, it would be necessary to enforce end-to-end "less-than-best-effort" Scavenger-class queuing policies within the entire network.

Another use of policers relating to security is found in control plane policing (CoPP). CoPP is a feature that allows the configuration of QoS policers to rate-limit the traffic destined to the main CPU of the switch/router. Such CoPP policers serve to protect the control plane of the switch/router from DoS attacks and reconnaissance activity.

The use of QoS policers to harden the security of the network infrastructure is described in more detail in Chapter 10, "Business and Application QoS Requirements."

Policing Tools

Different variations of policing algorithms exist, as discussed earlier in this chapter in the "Types of Policers" section, but they all meter traffic into the following categories, each of which can be assigned a separate action. Possible actions include sending, re-marking and sending, or dropping the traffic:

- **Conforming:** Traffic falls within the configured rate.
- **Exceeding:** Traffic above the configured rate but within the burst parameters.
- **Violating:** Traffic above both the configured rate and the burst parameters.

Policers as Markers

If none of the configured actions for a policer includes a drop function, the operation of the policer becomes that of a marker. Re-marking packets with a policer should be done with care, keeping in mind the overall policies of the network. Packets are usually marked as close to the source as technically and administratively feasible (either at the source itself, if trusted, or at a network trust boundary). In these locations, traffic is usually marked by application (for example, voice, call signaling, video, high-priority data), and so on. This can be thought of as a *vertical separation of traffic*.

A policer, however, has no direct knowledge of applications and marks traffic (if configured to do so) purely based on the traffic rate. This can be thought of as *a horizontal separation of traffic*. A policer could have indirect knowledge of an application by virtue of the class where the policer is applied. The policer itself, however, does not look at the class; it looks only at the rate of the traffic flowing through it.

Class-Based Policing (Policy Maps)

Class-based policing is the recommended tool to use for policing. The **police** command within the Modular QoS command-line interface (MQC) syntax is used to trigger a policing action on a class of traffic within a policy map. The single-rate policer is used if the **pir** keyword is omitted, and the dual-rate policer if the **pir** keyword is present.

The generic syntax structure for the single-rate policer is as follows:

```
police cir bps [ [bc] normal-burst-bytes [maximum-burst-bytes |
[be] [burst-bytes] ]] [conform-action action [exceed-action action
[violate-action action]]]
```

The generic syntax structure for the dual-rate policer is this:

```
police cir cir [bc conform-burst] [pir pir] [be peak-burst]
[conform-action action [exceed-action action [violate-action
action]]]
```

Traffic CIR and PIR rates can also be expressed as percentages instead of absolute rates. Example 4-1 shows all three variations of the class-based policing **police** command.

Example 4-1 *Class-Based Policing*

```
Router# show run
! Single-rate policer
policy-map POLICY1
  class C1
    police cir 1000000 conform-action transmit exceed-action drop
!
! Dual-rate policer
policy-map POLICY2
  class C2
    police cir 500000 bc 10000 pir 1000000 be 10000 conform-action
    transmit exceed-action set-prec-transmit 2 violate-action drop
!
! Percentage-based policing
```

```
policy-map POLICY3
   class C3
     police cir percent 20 bc 300 ms be 400 ms pir percent 40
     conform-action set-cos-inner-transmit 3
```

Traffic in classes that are not policed contends for bandwidth availability along with all other traffic on the interface. When no congestion exists, all traffic is transmitted. When congestion is experienced, packets can be dropped out of nonpoliced classes and policed classes. It is therefore better to configure explicit policers so that traffic is dropped based on your preferences.

A general summary of actions that can be taken on exceed or violate (some keywords are platform specific) is as follows.

- **drop**
- **set-clp-transmit**
- **set-cos-inner-transmit**
- **set-cos-transmit**
- **set-discard-class-transmit**
- **set-dscp-transmit**
- **set-dscp-tunnel-transmit**
- **set-frde-transmit**
- **set-mpls-experimental-imposition-transmit**
- **set-mpls-experimental-topmost**
- **set-mpls-experimental-topmost-transmit**
- **set-prec-transmit**
- **set-prec-tunnel-transmit**
- **set-qos-transmit**
- **transmit**

Drop and transmit are the basic actions, and the remainder are re-marking (in addition to transmit) actions which are, in general, the same as the marking actions that can be specified with the **set** command covered in Chapter 3, "Classification and Marking."

Multi-Action Policing

At times, multiple actions are required on conforming (or exceeding or violating) traffic (for example, when packets need to be re-marked at both Layer 2 and Layer 3). Example 4-2 illustrates how to mark exceeding and violating traffic with both DSCP (at Layer 3) and the Frame Relay (FR) Discard Eligible (DE) bit (at Layer 2).

Example 4-2 *Multi-Action Policing*

```
Router# show run
policy-map MULTIACTION-POLICER
  class class-default
    police cir 10000 pir 2000000
    conform-action transmit
    exceed-action set-dscp-transmit af32
    exceed-action set-frde-transmit
    violate-action set-dscp-transmit af33
    violate-action set-frde-transmit
```

Hierarchical Policing

It is advantageous to police some applications at multiple levels. For example, it might be desirable to limit all TCP traffic to 10 Mbps, while at the same time limiting FTP traffic (a subset of TCP traffic) to no more than 1.5 Mbps. To achieve this nested policing requirement, hierarchical policing can be used with up to three levels.

The policer at the second level in the hierarchy acts on packets transmitted or re-marked by the policer at the first level. Example 4-3 shows the configuration for the nested, two-level, hierarchical policing of TCP and FTP traffic.

Example 4-3 *Hierarchical Policing*

```
Router# show run
policy-map FTP-POLICER
  class FTP
   police cir 1500000
     conform-action transmit
     exceed-action  drop
!
 policy-map TCP-POLICER
  class TCP
   police cir 10000000
     conform-action transmit
     exceed-action  drop
     service-policy FTP-POLICER
!
interface ge 1/1
  service-policy output TCP-POLICER
```

Percentage-Based Policing

Most networks contain a wide array of interfaces with different bandwidths. If it is desirable to have an overall network policy in which, for example, FTP traffic is not to exceed 10 percent of the bandwidth on any interface (regardless of absolute speed), percentage-based policing can be used.

The CIR and PIR values can be specified with percent, but not the burst sizes; the burst sizes are configured in units of milliseconds. If the CIR is configured in percent, the PIR also must be. When the service policy is attached to an interface, the CIR (and PIR, if configured) is determined as a percentage of the interface bandwidth. If the interface bandwidth is changed, the CIR and PIR values and burst sizes are automatically recalculated using the new interface bandwidth value. For subinterfaces, the bandwidth of the main interface is used for the calculation.

If the percent feature is used in a second- or third-level policy, the bandwidth of the lower-level policy statement is determined by the configuration of the higher or parent level. Table 4-3 summarizes this decision process.

Table 4-3 *CIR/PIR Policing Behavior Based on Percentages*

Configuration	Decision
CIR and PIR configured	If the conform action is drop, the bandwidth is 0. (This is most likely not a practically useful configuration, but it is nevertheless the result of the police statement if this should be configured.) If the exceed action is drop, the CIR specification is used as the bandwidth. If the violate action is drop, the PIR specification is used as the bandwidth.
CIR configured	If the conform action is drop, the bandwidth is 0. (This is most likely not a practically useful configuration, but it is nevertheless the result of the police statement if this should be configured.) If the exceed action is drop, the CIR specification is used as the bandwidth. If the violate action is drop, the CIR specification is used as the bandwidth.
Neither CIR nor PIR configured	If this is a second- or third-level policy, the rate is determined by the parent class. If this is a first level policy, the interface of the permanent virtual circuit (PVC) bandwidth is used.

Color-Aware Policing

RFC 2697 and RFC 2698 describe three-color policers, meaning that the packets can be colored to three states: conform, exceed, or violate. These RFCs define two modes that these policers can operate in

- A color-blind mode, where the policer assumes that the packet stream was previously uncolored,
- A color-aware mode, where the policer assumes that some preceding entity has already colored the packet stream

Table 4-4 shows the two-rate policer decisions for the color-blind versus color-aware modes. The number of bytes in the packet under consideration is indicated by pkt-length.

Table 4-4 *Color-Blind Versus Color-Aware Policing Modes for the Dual-Rate Policer*

Color-Blind	Color-Aware	Explanation
If (Pkt-length > PIR + PBS), mark the packet as violating.	If the packet already is marked as violating OR (Pkt-length > PIR + PBS), mark the packet as violating.	The packet length exceeds the sum of the peak information rate and peak burst size.
If (Pkt-length < PIR + PBS) *and* (Pkt-length > CIR + CBS), mark the packet as exceeding.	If the packet already is marked as exceeding *or* ((Pkt-length < PIR + PBS) *and* (Pkt-length > CIR + CBS)), mark the packet as exceeding.	The packet length exceeds the sum of the committed information rate and committed burst size, but is less than the sum of the peak information rate and peak burst size.
Otherwise, mark the packet as conforming.	Otherwise, mark the packet as conforming.	The packet length is less than the sum of the committed information rate and committed burst size.

Policing as Part of Low-Latency Queuing

Queuing tools are discussed in Chapter 5, but it is worth noting here that the LLQ mechanism (triggered by the **priority** keyword within the MQC) contains an implicit policer. To contain latency, LLQ gives strict transmission priority to real-time traffic, and by doing so it introduces the possibility of starving lower-priority traffic. To prevent this situation, the LLQ mechanism polices traffic to the bandwidth specified in the **priority** statement by indiscriminately tail-dropping traffic exceeding the configured rate, specified as 512K in Example 4-4.

Example 4-4 *LLQ Policing*

```
Router# show run
policy-map LLQ
  class VOICE
    priority 512
```

The **priority** statement shown in Example 4-4 can be specified with an absolute bandwidth or by using a percentage.

Control Plane Policing

CoPP allows the configuration of QoS policers to rate-limit the traffic handled by the main CPU of the switch. These policers serve to protect the control plane of the switch/router from DoS attacks and reconnaissance activity. With CoPP, QoS policies are configured to permit, block, or rate-limit packets destined to the main CPU. For example, if a large amount of multicast traffic is introduced into the network with a Time To Live (TTL) of 1, this traffic would force the switch to decrement the TTL, and thereby force the control plane to send an ICMP (Internet Control Message Protocol) error message. If enough of these events happened, the CPU would not be able to process them all, and the node would be effectively taken out of service. CoPP can protect a node against this type of attack. Example 4-5 shows a policy that limits Border Gateway Protocol (BGP) routing traffic to 500 packets per second (pps), Interior Gateway Protocol (IGP) routing traffic to 50 pps, file management traffic to 400 pps, and all unclassified (default) traffic to 100 pps. The policy is attached to the control plane (which is presented as a logical interface within the configuration) in both the input and output directions.

Example 4-5 *CoPP Policy*

```
Router# show run
! Define the policy CPP-POLICY
policy-map CPP-POLICY
  class CPP-ACL-BGP
    police rate 500 pps
    conform-action transmit
    exceed-action drop
     ! Polices BGP control-plane traffic to 500 pps
  class CPP-ACL-IGP
    police rate 50 pps
    conform-action transmit
    exceed-action drop
   ! Polices IGP control-plane traffic to 50 pps
 class CPP-ACL-FILE-MANAGEMENT
   police rate 400 pps
```

```
     conform-action transmit
     exceed-action drop
     ! Polices File-Mgmt control-plane traffic to 400 pps
 class class-default
   police rate 100 pps
     conform-action transmit
     exceed-action drop
     ! Polices all other control-plane traffic to 100 pps
! Apply CPP policy to the control plane in both directions
control-plane
  service-policy input CPP-POLICY
  service-policy output CPP-POLICY
```

Unconditional Packet Drop

Packets belonging to a certain class can be dropped unconditionally, and although this is not strictly speaking a policing tool (because it does not meter traffic based on a certain rate as policers do), it is nevertheless another tool for regulating traffic in a network. Example 4-6 shows how traffic, usually undesirable traffic on the network, can be dropped outright in the MQC.

Example 4-6 *Unconditional Packet Drop*

```
Router# show run
policy-map EXAMPLE
  class UNDESIRABLE
    drop
```

Traffic Shaping Tools

Similar to policers, shapers meter the transmission rate of packets through a network; however, unlike policers, shapers delay (instead of drop or re-mark) packets that exceed the committed rate. Such delaying smooths out bursts in traffic flows and allows for conformance to SLAs, as shown in Figure 4-5.

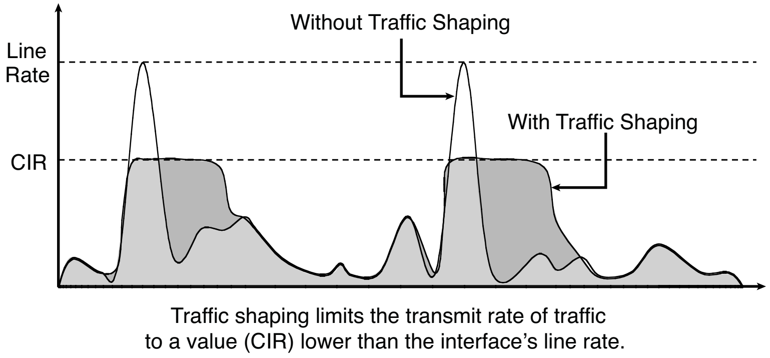


Figure 4-5 *Traffic Shaping Effect on Traffic Flow*

Shapers prove particularly useful when traffic must conform to a specific rate to meet a SLA or to guarantee that traffic offered to a service provider is within a contracted rate. Traditionally, shapers have been associated with nonbroadcast multiple-access L2 WAN topologies, like Asynchronous Transfer Mode (ATM) and Frame Relay (FR), where potential speed mismatches exist. However, shapers are increasingly deployed on L3 WAN access circuits, such as Ethernet-based handoffs, to conform to subline access rates. Shapers are also a key element in Hierarchical Queuing Framework (HQF), discussed in Chapter 2, "IOS-Based QoS Architectural Framework and Syntax Structure," where per-site shapers help to create backpressure to activate QoS policies applied to tunnels carrying traffic to each site. This technique is widely used in WAN aggregation routers and in virtual private network (VPN) deployments.

Similar to policers, shapers use token bucket algorithms, and use the CIR, Bc, and Be to smooth a traffic stream to a specified rate. By default, some shapers set the Bc to equal CIR / 8, which yields an interval (Tc) of 125 ms. This interval value might be adequate for data, but it is not for real-time traffic, which cannot tolerate 125-ms end-to-end delays.

Therefore, it is recommended that shaped interfaces carrying real-time traffic be shaped to 10-ms intervals (Tc). Tc cannot be administered directly by Cisco IOS commands. Instead, use Bc tuning to affect Tc indirectly. A 10-ms Tc interval value can be achieved (Tc = Bc / CIR) by setting the Bc equal to CIR / 100.

Class-Based Shaping (Policy Maps)

Class-based shaping is the recommended tool to use for traffic shaping. The **shape** command within the MQC is used to trigger a shaping action on a class of traffic within a policy map which details the CIR, Bc, Be, and other shaping parameters.

The generic syntax structure for class-based traffic shaping is as follows:

```
shape [average | peak] [cir] [Bc burst_size [Be excess_burst_size]]
```

Example 4-7 shows class-based shaping on a T1 interface to a 768-Kbps CIR with the Bc set to the recommended value for real-time networks (CIR / 100) and Be set to 0.

Example 4-7 *Class-Based Traffic Shaping*

```
Router# show run
policy-map CBS-768
  class class-default
    shape average 768000 7680 0
```

Hierarchical Class-Based Shaping

As with hierarchical policing, shaping policies sometimes need to shape aggregate levels and provide additional QoS functions on sublevels of traffic. For example, a service provider wants to shape a customer's aggregate traffic to 384K and, within that, provide a bandwidth guarantee (via a queuing policy) of 200K for transactional data and 100K for bulk data. This can be achieved with a nested, or hierarchical, shaping policy, as shown in Example 4-8. The PARENT-POLICY is attached to the interface with the purpose to create backpressure, as QoS policies become active only when congestion occurs

Example 4-8 *Hierarchical Class-Based Shaping*

```
Router# show run
policy-map PARENT-POLICY
  class CUSTOMER1-TRAFFIC
    shape average 384000
    service-policy CHILD-POLICY
!
policy-map CHILD-POLICY
  class CUSTOMER1-TRANSACTIONAL-DATA
    bandwidth 200
  class CUSTOMER1-BULK-DATA
    bandwidth 100
```

Percentage-Based Shaping

As with policing, shaping can be stated as a percentage of bandwidth instead of absolute bandwidth. Burst sizes cannot be configured using percentages, but they can be specified in terms of milliseconds. When a percentage configuration is used for average or peak shaping rates, default burst sizes are automatically assigned, or can be stated explicitly as shown in Example 4-9. When the service policy is attached to an interface, the CIR value is determined from the interface bandwidth in conjunction with the configured CIR percent value.

Example 4-9 *Percentage-Based Shaping*

```
Router# show run
policy-map POLICY1
  class-map class1
    shape average percent 25 300 ms 400 ms
interface serial 3/1
  service-policy output POLICY1
```

Legacy Shaping Tools

Several shaping tools, such as ATM and Frame Relay Traffic Shaping (FRTS) and generic traffic shaping (GTS), predate the MQC QoS syntax framework and are covered here merely for historical completeness. Class-based shaping is the preferred and recommended Cisco IOS shaping tool.

ATM Traffic Shaping

Shaping is an integral part of an ATM PVC L2 specification. ATM technology achieves different L2 QoS levels by configuring different PVC traffic contracts, such as real-time variable bit rate (VBR-RT), non-real-time variable bit rate (VBR-NRT), available bit rate (ABR), and unspecified bit rate (UBR). L3 shaping mechanisms like class-based shaping are not typically used with ATM (including digital subscriber line [DSL]) technologies. The parameters specified on the ATM PVC configuration determine the shaped average or peak rates, and the burst parameters, as shown in Example 4-10.

Example 4-10 *ATM PVC Configuration*

```
Router# show run
interface ATM3/0.1 point-to-point
  ip address 10.2.12.1 255.255.255.252
  pvc 0/12
    vbr-nrt 149760 149760
```

Frame Relay Traffic Shaping

The initial shaping mechanism for Frame Relay interfaces was FRTS, triggered by specifying the **frame-relay traffic-shaping** command on the interface, as shown in Example 4-11. This legacy tool should no longer be used. Instead, class-based traffic shaping should be used on FR interfaces.

Example 4-11 *Legacy Frame Relay Traffic Shaping*

```
Router# show run
interface Serial0/1
  no ip address
  encapsulation frame-relay
  frame-relay traffic-shaping
!
interface Serial0/1.50 point-to-point
  bandwidth 1536
  ip address 10.200.50.1 255.255.255.252
  frame-relay interface-dlci 150
    class FRTS-1536
!
map-class frame-relay FRTS-1536
  frame-relay cir 1536000
  frame-relay bc 15360
  frame-relay be 0
  frame-relay mincir 1536000
```

Recommendations and Guidelines

General guidelines for traffic policing and shaping include the following:

- MQC class-based policing and shaping are the tools of choice and should be used wherever possible.
- Policing should be done as close to the source of the traffic as possible, and on ingress rather than egress, especially if packets are going to be dropped.
- The single-rate three-color policer's tolerance of temporary bursts results in fewer TCP retransmissions than the two-color policer algorithm and is therefore more efficient for bandwidth utilization.
- Interfaces with speed mismatches (either between the physical transmission speed and the SLA or between a remote-end access link and the aggregated head-end link) should be shaped to smooth out traffic flows.
- If shaping is done on an interface carrying real-time traffic, the Tc value should be 10 ms.

Summary

This chapter examined policing and shaping algorithms, features, and tools. Both policing and shaping are methods of metering traffic and making treatment decisions about traffic that exceeds the metered rate. Policing makes an instantaneous drop, transmit, or re-mark and transmit decision on each packet, whereas shaping delays and smooths out traffic (with the help of buffering) over time to meet the configured rate.

Scheduling tools such as queuing and additional methods of making drop decisions about excess traffic are discussed in Chapter 5.

Further Reading

General

QoS Policing and Shaping Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_plcshp/configuration/15-1mt/qos-plcshp-15-1mt-book.html

DiffServ Policing Standards

RFC 2697, A Single Rate Three Color Marker: http://www.ietf.org/rfc/rfc2697

RFC 2698, A Two Rate Three Color Marker: http://www.ietf.org/rfc/rfc2698

Policing

QoS SRND: policing and markdown tools: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html#wp60957

Class-based policing: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_plcshp/configuration/15-1mt/qos-plcshp-class-plc.html

Control plane policing: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_plcshp/configuration/15-1mt/qos-plcshp-ctrl-pln-plc.html

Control plane policing: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html#wp131394

QoS: Classification Policing and Marking on a LAC (L2TP Access Concentrator) Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_lac/configuration/15-1mt/qos-lac-15-1mt-book.html

Committed access rate: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_plcshp/configuration/15-1mt/qos-plcshp-oview.html#GUID-B9D42CA8-EE64-4EE4-B82C-5931CFAFF335

ATM policing: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_plcshp/configuration/15-1mt/qos-plcshp-atm-plc.html

Shaping

QoS SRND: shaping tools: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html#wp60968

Class-based shaping: http://www.cisco.com/en/US/docs/ios/12_1t/12_1t2/feature/guide/clsbsshp.html

Class-based shaping for FR: http://www.cisco.com/en/US/docs/ios/12_2t/12_2t13/feature/guide/frqosmqc.html

Configuring ATM traffic parameters: http://www.cisco.com/en/US/docs/ios/12_2/wan/configuration/guide/wcfatm.html#wp1001126

Configuring Frame Relay Traffic Shaping for VoIP and VoFR: http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a00800d6788.shtml.

This page intentionally left blank

Chapter 5

Congestion Management and Avoidance Tools

This chapter covers the following topics:

- Terminology and recommendations
- Cisco IOS Layer 2 and Layer 3 queuing tools
- Cisco IOS congestion avoidance tools such as random early detection (RED) and weighted RED (WRED)

After traffic has been analyzed and identified to be treated by some set of actions, the next quality of service (QoS) task is to assign actions or policy treatments to these classes of traffic, including bandwidth assignments, policing, shaping, queuing, and dropping decisions.

Chapter 4, "Policing, Shaping, and Markdown Tools," explored various ways of limiting traffic rates. In this chapter, congestion management (queuing) and congestion avoidance (early-dropping) tools are explored which help to manage excess traffic until it can be sent.

Note The queuing and scheduling mechanisms discussed in this chapter are for the Cisco IOS-based router implementations, for two reasons: First, the IOS routers are the primary featured platform throughout this book; and second, the purpose of this chapter is to introduce general concepts relating to queuing and scheduling. Other place in the network (PIN)-specific queuing and scheduling mechanisms—such as the campus switch hardware queuing, the Nexus Priority Flow Control (PFC) and Enhanced Transmission Selection (ETS) methods in the data center, and the Wireless Multimedia (WMM) access categories for WLANs are discussed in the respective design parts covering those technologies, including platform-specific internal queuing mechanisms.

Congestion Management and Avoidance Topics

Congestion management tools, including *queuing tools*, apply to interfaces that may experience congestion. Whenever packets enter a device faster than they can exit, the potential for congestion exists and queuing mechanisms apply—it is important to note that queuing tools are activated *only* when congestion exists. In the absence of congestion, packets are sent as soon as they arrive. However, when congestion occurs, packets must be *buffered*, or queued—the temporary storage and subsequent scheduling of these backed-up packets—to mitigate dropping.

Packet markings at either Layer 2 (L2) or Layer 3 (L3) generally influence queuing policies so that the device can reorder packets before transmission. Therefore, queuing policies are usually complementary and depend on classification and marking policies.

Congestion Management and Avoidance Terminology

Congestion management encompasses both queuing and scheduling. Conceptually, queuing and scheduling are complementary but intertwined processes. These terms are quite often incorrectly used interchangeably:

- **Queuing** (which is also referred to as buffering) is the logic of ordering packets in linked output buffers. Queuing processes are engaged when an interface is experiencing congestion and are deactivated when congestion clears. As queues fill, packets can be reordered so that higher priority packets exit the device sooner than lower priority ones.
- **Scheduling** is the process of deciding which packet to send next. Scheduling (unlike queuing) occurs regardless of whether the interface is experiencing congestion.

Figure 5-1 shows a general example of queuing and scheduling.

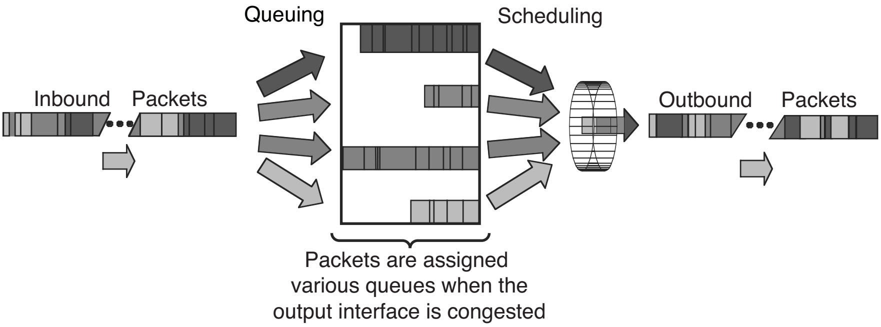

Queuing
Scheduling
Inbound Packets
Outbound Packets
Packets are assigned various queues when the output interface is congested

Figure 5-1 *Generic Queuing Example*

Congestion Management and Congestion Avoidance

Congestion management includes queuing and scheduling methods where excess traffic is buffered or queued (and sometimes dropped) while it waits to be sent on an egress interface.

Congestion avoidance tools are simpler; they monitor network traffic loads in an effort to anticipate and avoid congestion and by selectively dropping traffic before congestion occurs.

Scheduling Algorithms

Under congestion conditions, queuing algorithms are engaged and the scheduler makes decisions on which queue to service next. This decision can use different scheduling algorithms, including the following Cisco IOS router mechanisms:

- **Strict priority:** The lower-priority queues are served only when the higher-priority queues are empty. This method can potentially starve traffic in lower-priority queues.
- **Round-robin:** Queues are served in a set sequence. While this algorithm never starves traffic, it can potentially cause unpredictable delays in queues that hold real-time, delay-sensitive traffic.
- **Weighted fair:** Packets in the queues are weighted, usually by IP precedence, so that some queues are served more frequently than others. Although this method solves the downsides of both the strict-priority and round-robin algorithms, it does not provide the bandwidth or rate guarantee that real-time flows require. The resulting bandwidth per flow instantaneously varies based on the number of flows present and the weights of each of the other flows.

The scheduling tools used for QoS deployments therefore offer a combination of these algorithms and various ways to mitigate their downsides to allow you to best tune your network for the actual traffic flows present.

The Cisco switch platforms employ other queuing variations, including shaped round-robin (SRR) and deficit weighted round robin (DWRR), which are addressed in the chapters covering switch design.

Levels of Queuing

Queuing (and scheduling) happens at various layers for any particular traffic flow. The most sophisticated queuing algorithms exist at L3 and are independent of the interface type. Layer 3 queuing methods typically consider only the IP packet overhead in its bandwidth provisioning. On the Cisco IOS router platforms, L3 queuing can also be configured in a hierarchical manner so that a cascade of L3 queues feed traffic to lower-layer queues.

There is also queuing at L2, for certain interface types, to accommodate media-specific requirements and idiosyncrasies, such as the older technologies for Asynchronous Transfer Mode (ATM) and Frame Relay (FR) circuits. When the L2 queues fill up, they, in turn, push back packets into the L3 queues.

A final queue, usually referred to as a *transmit ring* (or Tx-Ring), is located within the L1 device driver. Tx-Rings are media and hardware dependent and can operate quite differently on different routers, cards, and modules. When the Tx-Ring queue fills up, the higher-level queues are pressed into service and this is essentially when QoS becomes active on the device.

Note Not all levels of queuing are always present.

Queuing and Scheduling Tools

This section examines the different L1, L2, and L3 queuing tools, with emphasis on the L3 tools. Cisco IOS class-based weighted fair queuing (CBWFQ) and low-latency queuing (LLQ) are examined in detail because these are the principal recommended router-based queuing mechanisms for rich-media networks where different traffic types share the same transmission media. Legacy queuing methods are covered only briefly for historical perspective.

Whether to use the default settings for the queuing methods or to tune them is difficult to answer as a general statement. Instead, it depends on the platform and the specific design being put in place. To this end, default settings, or changes to them, are discussed in the relevant design chapters for various places in the network (PINs) and specific platforms.

Class-Based Queuing (Policy Maps)

There is a long history of queuing algorithms in Cisco IOS Software. The older methods are insufficient for modern rich-media networks as they predate these traffic types, but a quick review is instructive. The key legacy queuing methods (all of which predate the Modular QoS command-line interface [MQC] architecture) include the following:

- **First-in, first-out queuing (FIFO):** A single queue with packets sent in the exact order they arrived.
- **Priority queuing (PQ):** A set of four queues served in strict-priority order. This method's major drawback is the potential for starvation of lower-priority traffic.
- **Custom queuing (CQ):** A set of 16 queues with a round-robin scheduler. It provides bandwidth guarantees and prevents starvation, but does not provide the strict priority required by delay-sensitive, real-time flows.
- **Weighted fair queuing (WFQ):** The algorithm divides the interface's bandwidth

by the number of flows (weighted by IP precedence, or IPP), ensuring an equitable distribution of bandwidth for all applications. This method provides better service for high-priority real-time flows, but lacks a bandwidth guarantee for any particular flow.

- **IP RTP priority queuing (PQ-WFQ):** This was a transient method providing a single strict-priority queue for real-time traffic in addition to a WFQ complex for other traffic. It was soon superseded by low-latency queuing (LLQ).

The current, and much newer, queuing mechanisms recommended and suitable for rich-media networks (and present in the MQC) sought to combine the best features of the legacy algorithms and, at the same time, to minimize their drawbacks. Real-time, delay-sensitive traffic requires two attributes of a queuing algorithm: an absolute bandwidth guarantee and a delay guarantee. Other traffic requires not to be starved in the presence of real-time traffic. The current recommended queuing algorithms are as follows:

- **Class-based weighted fair queuing (CBWFQ):** A hybrid queuing algorithm combining a bandwidth guarantee (from CQ) with dynamic fairness to other flows within a class of traffic (from WFQ). It does not provide a latency guarantee and as such is suitable only for data traffic management.
- **Low-latency queuing (LLQ):** This method adds a strict-priority capability to CBWFQ and is therefore suitable for mixes of real-time and non-real-time traffic. It provides both latency and bandwidth guarantees.

Table 5-1 compares the attributes of the recommended algorithms.

Table 5-1 *Queuing Algorithm Comparison*

Attribute	CBWFQ	LLQ
Classification	Class based	Class-based
Number of queues	Up to 256 classes	1 PQ* + CBWFQ
Bandwidth Guarantee	Yes	Yes
Latency Guarantee	No	Yes
Recommended for real-time traffic	Generally no, but under certain conditions video traffic can be handled	Yes

* LLQ functionally supports only a single strict-priority queue, but multiple “LLQs” may be configured, as discussed later in this chapter.

Class-Based Weighted Fair Queuing

CBWFQ enables the creation of up to 256 queues, serving up to 256 classes of traffic. Each queue is serviced based on the bandwidth assigned to that class. CBWFQ is configured using the **bandwidth** keyword in a policy map. With CBWFQ, a minimum bandwidth is explicitly defined and enforced. The bandwidth can be specified in absolute or percentage terms (**percent** keyword).

Example 5-1 shows two applications managed with CBWFQ. First, a minimum bandwidth guarantee is given to the TRANSACTIONAL-DATA class. Second, all other traffic (which falls into class-default) is fair queued.

Example 5-1 *CBWFQ Policy Example*

```
Router# show run
policy-map WAN-EDGE
 class TRANSACTIONAL-DATA
   bandwidth percent 20   ! Bandwidth for this class to 20%
   fair-queue             ! fair-queuing pre-sorter
 class class-default
   fair-queue             ! Fair-queuing for unclassified traffic
```

In the event of congestion, the L1 Tx-Ring for the interface fills up and pushes packets back into the L3 CBWFQ queues (if configured). Each CBWFQ class is assigned its own queue. CBWFQ queues may also have a fair-queuing presorter applied (using the **fair-queue** keyword within a policy map) to manage fairly multiple flows contending for a single queue. In addition, each CBWFQ queue is serviced in a weighted round-robin (WRR) fashion based on the bandwidth assigned to each class. The CBWFQ scheduler then forwards packets to the Tx-Ring.

Figure 5-2 illustrates the operation of CBWFQ.

Low-Latency Queuing

LLQ is essentially CBWFQ combined with a single, strict PQ. Traffic assigned to the strict-priority queue, using the **priority** keyword, is serviced up to its assigned bandwidth before other CBWFQ queues are serviced. All real-time traffic should be configured to be serviced by the priority queue. Multiple classes of real-time traffic can be defined, and separate bandwidth guarantees given to each, but a single priority queue schedules all that combined traffic.

As with CBWFQ, you can configure LLQ with absolute or percentage-based bandwidth allocations. In Example 5-2, 33 percent of the interface bandwidth is guaranteed for real-time traffic; other traffic classes receive CBWFQ bandwidth guarantees.

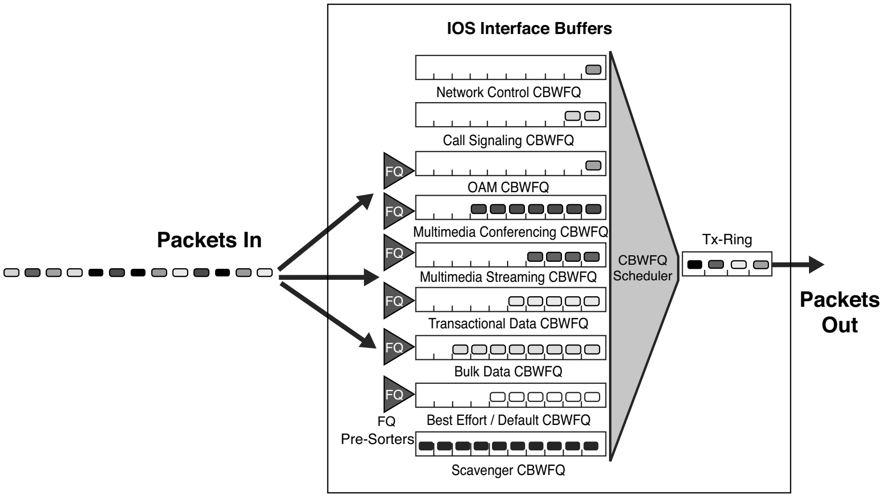


Figure 5-2 *CBWFQ Operation*

Example 5-2 *LLQ Policy Example*

```
policy-map WAN-EDGE-4-CLASS
  class REALTIME
    priority percent 33   ! 33% LLQ for REALTIME traffic
  class CONTROL
    bandwidth percent 7   ! 7% CBWFQ for CONTROL traffic
  class TRANSACTIONAL-DATA
    bandwidth percent 35  ! 35% CBWFQ for TRANSACTIONAL-DATA traffic
    fair-queue            ! fair-queuing pre-sorter
  class class-default
    bandwidth percent 25  ! 25% CBWFQ for Best-Effort traffic
    fair-queue            ! fair-queuing pre-sorter
```

Note Extensive testing in Cisco labs has shown that for links carrying a mix of voice, video, and data traffic, the amount of strict-priority queuing traffic (the sum of all traffic classes using the **priority** command) should not exceed 33 percent of link capacity. For links dedicated to a particular traffic type, such as a TelePresence service, this guideline does not apply.

As mentioned in Chapter 4, in the "Policing as Part of Low-Latency Queuing" section, LLQ includes an implicit policer that limits the bandwidth that can be consumed by traffic in the real-time queue and thus prevents bandwidth starvation of the non-real-time flows serviced by the CBWFQ scheduler. The policing rate for this implicit policer is the bandwidth allocated to the class, and traffic exceeding this rate is tail dropped.

Figure 5-3 illustrates the operation of LLQ. In the illustration, three real-time classes of traffic are all funneling into the priority queue of LLQ while other classes of traffic use the CBWFQ algorithm. Example 5-3 provides a configuration that matches this traffic flow with an aggregate of 10M of real-time traffic funneling into the single priority queue, but each class individually policed to 1M VoIP, 4M of broadcast video, and 5M of interactive video.

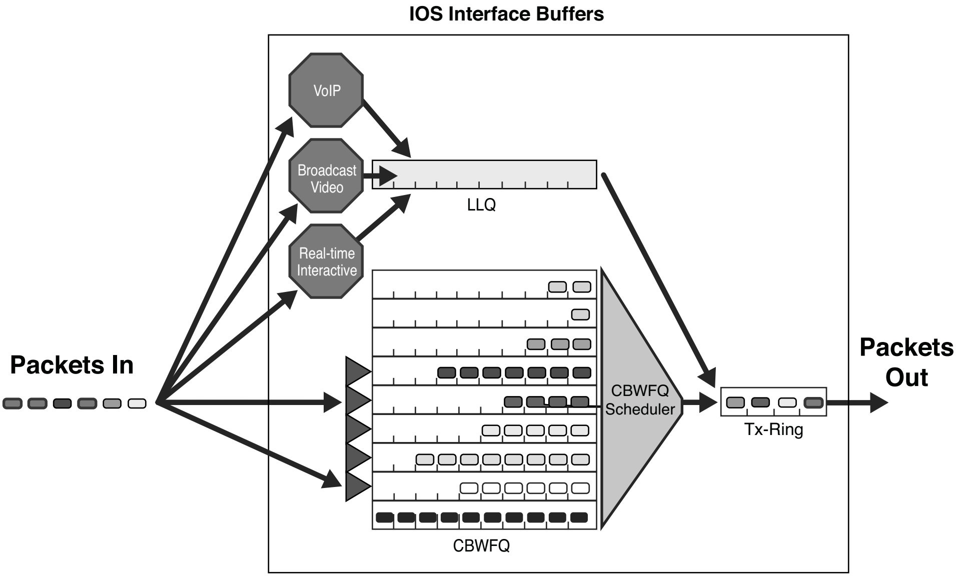


Figure 5-3 *LLQ Operation*

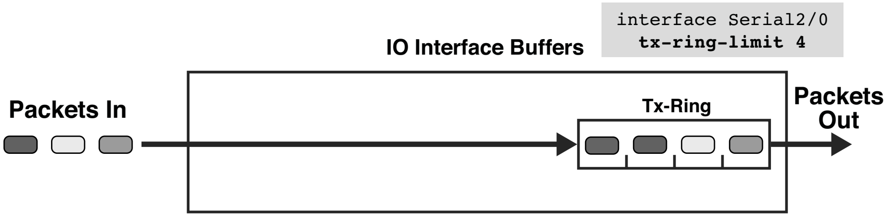
Example 5-3 *LLQ Configuration for Multiple Class LLQ Traffic*

```
Router# show run
class-map match-all VOIP
  match dscp ef
class-map match-all BROADCAST-VIDEO
  match dscp cs5
class-map match-all REALTIME-INTERACTIVE
  match dscp cs4
policy-map WAN-EDGE
  class VOIP
    priority 1000  ! 1M of VoIP traffic
  class BROADCAST-VIDEO
    priority 4000  ! 4M of Broadcast video traffic
  class REALTIME-INTERACTIVE
    priority 5000  ! 5M of Interactive video traffic
```

Queuing Below Layer 3: Tx-Ring Operation

Device driver level queuing at L1 is done as the last egress point before transmission. Specific link technologies such as FR and ATM also have queuing at L2, but these are very old technologies and are not discussed in any further detail.

The Tx-Ring, a L1 queuing mechanism, is a relatively small FIFO queue and is the final output buffer for a WAN interface. Its purpose is to maximize the physical link bandwidth utilization by matching the outbound transmission rate with the physical interface rate. Figure 5-4 illustrates the Tx-Ring operation.



```
interface Serial2/0
 tx-ring-limit 4
```

Figure 5-4 *Tx-Ring Operation*

The placement of the Tx-Ring compared to L3 queues is shown on the right sides of Figures 5-2 and 5-3. If the Tx-Ring fills to capacity, the interface is considered congested and the software activates any LLQ/CBWFQ policies that have been applied to the interface.

The size of the Tx-Ring depends on the hardware, software, L2 media, and queuing algorithm configured on the interface. It is a general best practice to set the Tx-Ring to a value of 3 on WAN interfaces of up to 768K.

The Tx-Ring is especially important on ATM links (including digital subscriber line [DSL]), in which each permanent virtual circuit (PVC) has a dedicated driver-level Tx-Ring to ensure adherence to the ATM class of service (CoS) of the PVC. Default ATM Tx-Rings are usually deep, containing 64 or more particles (each particle is usually 576 bytes) to ensure that enough particles exist in the buffer to drive the PVC to its full bandwidth utilization. A shallow Tx-Ring increases the number of interrupts and wait states between the driver and the main CPU and is suboptimal for higher-speed circuits. A deep Tx-Ring, however, can impact voice quality because the Tx-Ring is a FIFO and packets can no longer be reorganized based on priority to reduce latency.

Therefore, the tuning of the Tx-Ring is a compromise between optimizing real-time packet latency and achieving maximum CPU and bandwidth efficiency. The higher the bandwidth of the link or circuit, the deeper the Tx-Ring can be set without adversely affecting voice or video quality.

Congestion Avoidance Tools

Buffering memory is a limited resource on any interface, and when queuing buffers fill up packets might be dropped either as they arrive (tail drop) or selectively, before all buffers are filled. Selective dropping of packets when queues are filling up is referred to as *congestion avoidance*. Queuing algorithms manage the *front* of a queue, and congestion avoidance mechanisms manage the *tail* of a queue.

Congestion avoidance mechanisms work best with TCP-based applications because selective dropping of packets causes the TCP windowing mechanisms to throttle back and adjust the rate of flows to manageable rates. TCP has built-in flow-control mechanisms that operate by increasing the transmission rates of traffic flows (even though bounded by buffers and window sizes) until packet loss occurs. At this point, TCP abruptly squelches the transmission rate and gradually begins to ramp the transmission rates higher again. Figure 5-5 shows the suboptimal bandwidth utilization that tail drop has on TCP traffic. For this reason, random dropping congestion avoidance mechanisms are much more effective in managing TCP traffic. Although this figure may be somewhat theoretical, it does illustrate visually the effects of managing TCP traffic queuing—in practice, these "waves" may happen to a larger or smaller extent depending on the actual traffic characteristics and flow patterns present.

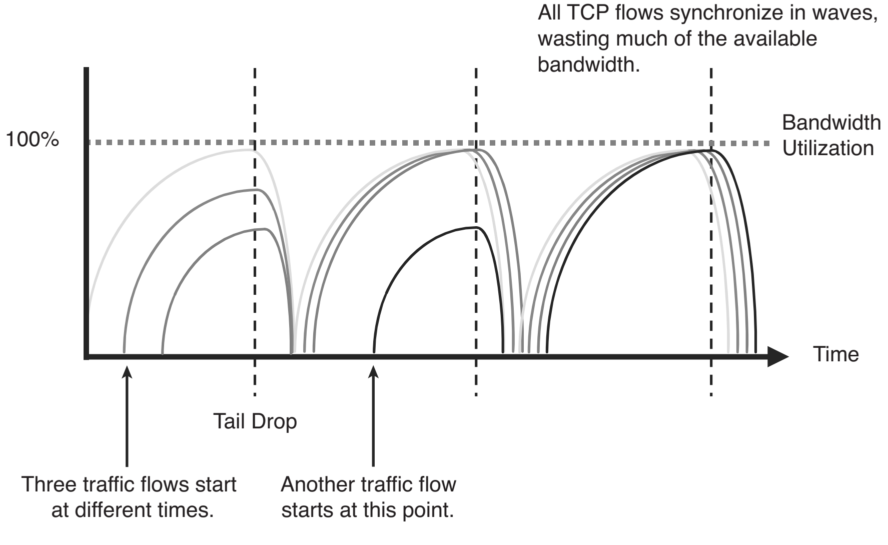

All TCP flows synchronize in waves, wasting much of the available bandwidth.
100%
Bandwidth Utilization
Time
Tail Drop
Three traffic flows start at different times.
Another traffic flow starts at this point.

Figure 5-5 *TCP Global Synchronization*

Random Early Detection

RED counters the effects of TCP global synchronization (TCP sessions do a periodic synchronization exchange, and these exchanges cause waves of extra traffic on the network; if they all hit at once, a global synchronization event results) by randomly dropping packets before the queues fill to capacity. Randomly dropping packets instead of dropping them all at once, as is done in a tail drop, avoids global synchronization of TCP streams. RED monitors the buffer depth and performs early discards (drops) on random packets when the minimum defined queue threshold is exceeded.

Note The User Datagram Protocol (UDP) does not contain inherent retry logic, with the result that—in a very narrow, theoretical sense—congestion avoidance techniques do not have any significant effect on managing UDP-based traffic. In practice, however, the situation is more complicated, and the effect of congestion avoidance techniques on various types of UDP-based traffic should be carefully evaluated. Some high-layer protocols (for example, Trivial File Transfer Protocol [TFTP]) or applications that use UDP as transport may have retry logic of their own. In addition, RTP Control Protocol (RTCP) is sometimes used as an acknowledgment method that causes UDP-based applications to react to dropped packets. Video traffic may react in a variety of ways to dropped packets depending on what type of packet was dropped, a reference frame or a regular frame.

Cisco IOS Software does not support pure RED; it supports weighted RED. However, if all the packets assigned to an interface or class have the same differentiated services code point (DSCP) markings, the effective resulting policy is simply RED.

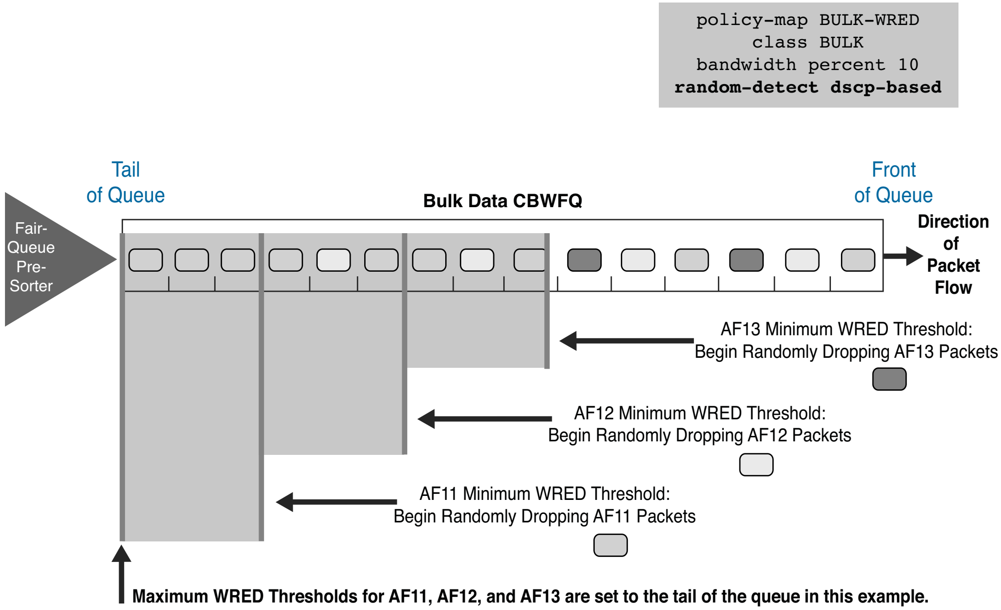
Weighted Random Early Detection

The principle Cisco IOS congestion avoidance mechanism is WRED. The randomness of packet-drop selection can be skewed by traffic weights denoted either by IPP or DSCP. Packets with lower IPP values are dropped more aggressively than higher values. (For example, IPP 1 would be dropped more aggressively than IPP 6.)

DSCP-based WRED weights are denoted by the Assured Forwarding (AF) Drop Preference values and the higher AF Drop Preference packets are dropped more aggressively. (For example, AF23 is statistically dropped more aggressively than AF22, which in turn is statistically dropped more aggressively than AF21.)

WRED can also be used to set the IP Explicit Congestion Notification (ECN) bits to indicate that congestion was experienced in transit.

Figure 5-6 illustrates DSCP-based WRED operation.



```
policy-map BULK-WRED
 class BULK
 bandwidth percent 10
random-detect dscp-based
```

Tail of Queue
Bulk Data CBWFQ
Front of Queue
Fair-Queue Pre-Sorter
Direction of Packet Flow
AF13 Minimum WRED Threshold: Begin Randomly Dropping AF13 Packets
AF12 Minimum WRED Threshold: Begin Randomly Dropping AF12 Packets
AF11 Minimum WRED Threshold: Begin Randomly Dropping AF11 Packets
Maximum WRED Thresholds for AF11, AF12, and AF13 are set to the tail of the queue in this example.

Figure 5-6 *DSCP-Based WRED Operation*

WRED, triggered with the **random-detect** Cisco IOS Software CLI command, enables WRED. DSCP-based WRED results if the optional keyword **dscp-based** is used; this should generally be used whenever the feature is available. Example 5-4 shows a WRED configuration.

Example 5-4 *WRED Configuration*

```
Router# show run
policy-map WAN-EDGE
  class REALTIME
    priority percent 33
  class TRANSACTIONAL-DATA
    bandwidth percent 35
    fair-queue
    random-detect dscp-based
  class class-default
    bandwidth percent 25
    fair-queue
    random-detect dscp-based
```

WRED and DSCP-based WRED thresholds and mark probability denominators are tunable on a per-code-point basis. In Example 5-5, the minimum threshold is set to begin dropping AF11-marked packets at 50. (That is, as soon as the queue depth reaches 50 packets, WRED randomly begins dropping AF11 packets.) The maximum threshold for AF11 (at which all AF11-marked packets are dropped) is set to 60 packets. The mark probability denominator is set to 8 (meaning that, between these two thresholds, up to 1 in 8 packets marked with AF11 are dropped, and at the maximum threshold of 20, exactly 1 in 8 packets are dropped—above the maximum threshold of 20, all packets marked with AF11 are dropped).

Example 5-5 *WRED with Thresholds Configuration*

```
Router# show run
policy-map DSCP-WRED
  class class-default
   random-detect dscp-based af11 50 60 8
```

WRED can also be used to set the RFC 3168 IP ECN bits to indicate that congestion was experienced in transit. ECN functionality is enabled with the **ecn** keyword in conjunction with the **random-detect** policy map class configuration command, as shown in Example 5-6.

Example 5-6 *WRED with ECN Configuration*

```
Router# show run
policy-map pol1
   class class-default
     bandwidth per 70
    random-detect ecn
```

Recommendations and Guidelines

General guidelines for queuing and congestion avoidance include the following:

- Critical applications such as VoIP require service guarantees regardless of network conditions—the only way to provide service guarantees is to enable queuing at any node that has the potential for congestion.
- Because a large number of unclassified applications always end up in the default class, it is recommended that you reserve at least 25 percent of link bandwidth for the default Best Effort class.
- If a link carries mixed voice, video, and data traffic, limit the amount of strict-priority queuing to 33 percent of link capacity; this 33 percent includes the aggregate of all traffic with the **priority** keyword in their policy maps.
- Enable LLQ whenever real-time, latency-sensitive traffic is present.

- Use WRED as a congestion avoidance method for TCP traffic (and carefully evaluate what its effect may be on different UDP classes of traffic).
- Use DSCP-based WRED whenever available.

Summary

Congestion management tools were examined in this chapter, with an emphasis on the Cisco IOS-based CBWFQ and LLQ algorithms. These are the principal recommended Cisco IOS queuing tools for rich-media networks.

Although most queuing policies occur at L3, it is important to consider any impacts that lower levels of queuing, such as L2 and L1 Tx-Ring queuing, might contribute to overall network design.

The primary congestion avoidance tool, WRED, was covered in this chapter. WRED is used to regulate TCP data traffic in a bandwidth-efficient manner before tail drops caused by queue overflows occur.

Platform-specific congestion management tools, variations, defaults, and tuning are discussed in greater detail in the design chapters covering those platforms.

Chapter 6, "Bandwidth Reservation Tools," focuses on more detailed QoS tools available and necessary on specific link types.

Further Reading

Queuing

Congestion Management Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_conmgt/configuration/15-1mt/qos-conmgt-15-1mt-book.html

CBWFQ operation: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html#wp129419

LLQ operation: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html#wp129441

Tx-Ring operation: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html#wp129403

ATM Tx-Ring tuning: http://www.cisco.com/en/US/tech/tk39/tk824/technologies_tech_note09186a00800fbafc.shtml

Congestion Avoidance

Congestion Avoidance Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_conavd/configuration/15-1mt/qos-conavd-15-1mt-book.html

RED: http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfconav_ps1835_TSD_Products_Configuration_Guide_Chapter.html#wp1000899

WRED: http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfconav_ps1835_TSD_Products_Configuration_Guide_Chapter.html#wp1000959

WRED Solution Reference Network Design (SRND): http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html#wp129476

This page intentionally left blank

Chapter 6

Bandwidth Reservation Tools

This chapter covers the following topics:

- Admission control mechanisms
- RSVP IntServ and DiffServ designs and deployment models
- RSVP local policies and application IDs
- Scalability, endpoints, and proxies

The quality of service (QoS) tools discussed so far in this book, including marking, queuing, policing, and shaping mechanisms, have primarily been differentiated services (DiffServ) tools. DiffServ mechanisms provide bandwidth *guarantees* (at various levels of rigidity), but none of them provides bandwidth *reservations*. Guaranteed implies that the bandwidth is there when needed, but it is not set aside (or reserved) for a specific application or flow. Reserved, however, implies that a flow of packets can be recognized and that a certain amount of bandwidth has been agreed to be set aside for that flow.

The QoS tools discussed thus far all deal with managing traffic already present on the network. Admission control (AC) differs fundamentally in that these tools deal with traffic not yet admitted to the network and aid in making the decision on whether to admit the new traffic stream. Traffic that reacts well to delayed and dropped packets (primarily data traffic) benefits little from being subjected to AC. But real-time traffic such as voice and video benefits greatly because these traffic types are delay sensitive and dropping packets quickly impairs the stream to the point of being useless to session end users. To this end, AC is often referred to as call admission control (CAC), but its conceptual foundation has broader applicability to all delay- and drop-sensitive traffic sessions, and not merely to voice "calls."

There are two general categories of CAC methods:

- **Measurement based:** These methods are counting mechanisms that allow only a limited number of calls (sessions). The limits are usually statically configured and based on the intellectual knowledge of the administrator—independent of the current status of the network. These methods are not further discussed in this chapter because they live in the applications and are transparent to network operation.

- **Resource based:** These methods are based on the availability of network resources, usually bandwidth, and use the current status of the network to aid in the admission decision. These methods are discussed further in this chapter.

The Resource Reservation Protocol (RSVP) is one of the oldest Cisco IOS QoS tools, dating from the mid-1990s. The development/implementation of RSVP precedes the era of rich media networks, yet the purpose of RSVP was always in line with these later trends: to provide predictable latency and reserved bandwidth for time-sensitive applications. RFC 2205 defines RSVP as follows:

> RSVP is used by a host to request specific qualities of service from the network for particular application data streams or flows.

RSVP has been most widely deployed as the "guaranteed-bandwidth" tunnel building mechanism for Multiprotocol Label Switching Traffic Engineering (MPLS-TE) networks.

RSVP differs from other QoS tools in the following ways:

- It is a signaling protocol.
- It reserves resources (bandwidth).
- A full implementation (which is rare) requires all nodes in the network to do the following:
  - Understand RSVP
  - Implement a way to reserve resources on that node

Admission Control Tools

Interactive applications such as voice and video require real-time services from the network. Because these resources are finite, the number of flows contending for such priority resources must be limited; otherwise, the quality of all real-time flows would degrade, eventually to the point of becoming unusable. Real-time flows cannot be delayed or buffered when network resources are oversubscribed (as data applications can be), and therefore new incoming real-time flows must be admitted based on available network resources to carry them at the time of session initiation. This action constitutes admission control.

AC functionality is most effectively achieved at an application level, such as with Cisco Unified Communications Manager, which controls VoIP and IP video and TelePresence flows. Therefore, AC is not covered in detail in this book, but only to the extent that AC functions interact with the allocation of network resources providing QoS.

Most AC mechanisms are fairly simple "counting" algorithms that are unaware of the network architecture and paths that flows can take—they are also referred to as "off-path" AC tools. When most flows are of a predictable bandwidth (as G.711 and G.729 voice calls are), and the network is of limited complexity (typically hub-and-spoke topologies), these AC mechanisms work perfectly well and are widely deployed. They do not perform well, however, when the flows are of widely varying bandwidth requirements (voice and several different video formats) or when the network topology is more complex and there are multiple paths the flows could take between points A and B in the network. Router-based Cisco IOS has a session counting mechanism (the **call threshold** CLI command) often associated with SIP (Session Initial Protocol), but not inherently part of the protocol. These CAC tools are used on Cisco IOS routers when they are deployed as voice or video gateways, or as session border controllers.

Wireless networks also have a limited capability to do AC (when configured to do so), based on the traffic specification (TSpec) information an 802.11e client can insert to signal its traffic requirements to the access point (AP).

Resource Reservation Protocol

Resource Reservation Protocol (RSVP) is a network-aware technology (also called an "on-path" AC tool) that allocates bandwidth along the actual path a flow will take before the flow is admitted to the network. It thus has dual functions: access control and bandwidth reservation (or reserving "tunnels" of guaranteed bandwidth for an individual flow). The first function (AC) is highly scalable, whereas the second function (per-flow bandwidth reservation) is not (at least not in the aggregated backbone of a network where multiple-thousands or millions of flows can exist simultaneously). This fact leads to the network design use of RSVP as an AC tool (integrated services [IntServ]) at the edge of the network, together with the use of the more scalable differentiated services (DiffServ) tools for classification, queuing, and bandwidth management for aggregate flows in nodes deeper in the network.

RSVP Overview

RSVP is a per-flow protocol that requests a bandwidth reservation from every node in the path of the flow. The endpoints, or other network devices on behalf of the endpoints, send unicast signaling messages to establish the reservation before the flow is allowed. Figure 6-1 illustrates basic RSVP operation.

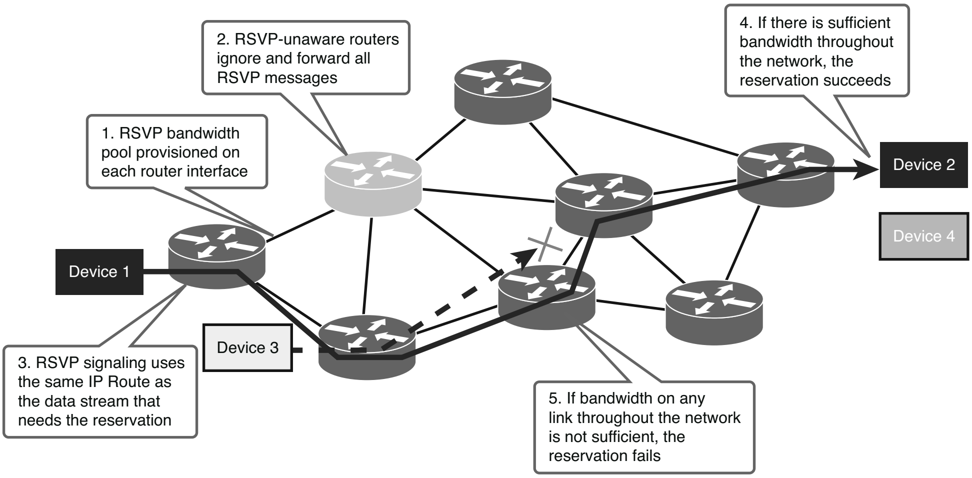

2. RSVP-unaware routers ignore and forward all RSVP messages
4. If there is sufficient bandwidth throughout the network, the reservation succeeds
1. RSVP bandwidth pool provisioned on each router interface
Device 2
Device 4
Device 1
Device 3
3. RSVP signaling uses the same IP Route as the data stream that needs the reservation
5. If bandwidth on any link throughout the network is not sufficient, the reservation fails

Figure 6-1 *Basic RSVP Operation*

Looking at Figure 6-1, you can see that

- In an ideal situation, every RSVP-enabled router in the path of the flow sees the messages and allocates the appropriate bandwidth for the given flow based on its interface configuration.
- In a more practical scenario, RSVP-unaware nodes in the network backbone merely forward the messages for the edge nodes to interpret.
- When Device 1 initiates a session to Device 2, it (or the router closest to the device acting as a proxy) initiates the RSVP reservation along the same route the actual flow will eventually follow.
- If there is sufficient bandwidth everywhere, the reservation succeeds and the session is admitted to the network.
- If Device 3 initiates a session and there isn't enough bandwidth somewhere along the path, the reservation fails and the session is not admitted.

RSVP and detailed methods of operation are covered in numerous Internet Engineering Task Force (IETF) Request For Comments (RFCs), including RFC 2205-2212, 2747, 2961, 2998, 3175, 3209, 3936, 4420, 4874 4874, 5151, 420, 5711, all of which you can review at http://www.ietf.org/rfc.html.

RSVP Proxy

Most end devices, such as phones and video endpoints (standalone devices and software applications on mobile devices, tablets, and computers), do not support the RSVP stack. To use RSVP as an AC mechanism for sessions initiated by these devices, the router closest to the device generally acts as a proxy, as illustrated in Figure 6-2.

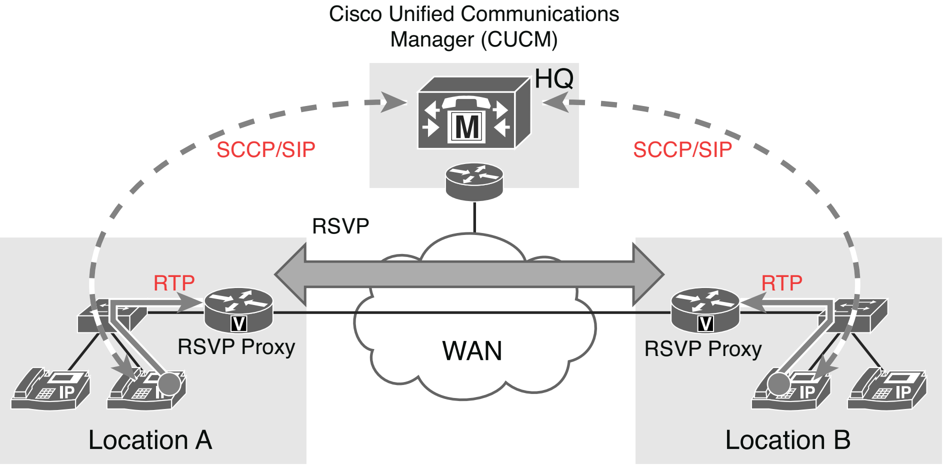


Figure 6-2 *RSVP Proxy Operation*

The on-location router co-resident with the endpoint uses an RSVP proxy (or RSVP agent) configuration that, in conjunction with Cisco Unified Communications Manager (CUCM), establishes AC before the session is allowed on the network. CUCM, at an application level, is ultimately in control of the admission decision, but it uses the network-aware knowledge of the RSVP-enabled routers to aid it in this decision, combining the strengths of centralized policy-oriented application-level AC tools with the network topology-aware benefits of a protocol like RSVP.

RSVP Deployment Models

RSVP can be deployed in two operational models, as shown in Figure 6-3:

- **IntServ model:** The legacy RSVP operational model, which has been largely abandoned due to inherent scalability limitations.
- **IntServ/DiffServ model:** Separates control plane operations from data plane operations. RSVP operation is limited to AC only, with DiffServ mechanisms handling classification, marking, policing, and scheduling operations. Therefore, the IntServ/DiffServ model is highly scalable and flexible.

As shown in Figure 6-3, under the IntServ/DiffServ model, RSVP is only used to perform AC, a control plane function; all other QoS functions—including classification, marking, policing, shaping, queuing, and dropping—are handled by DiffServ policies. These are all data plane functions. This combination allows for efficient scaling of policies along with network-aware AC.

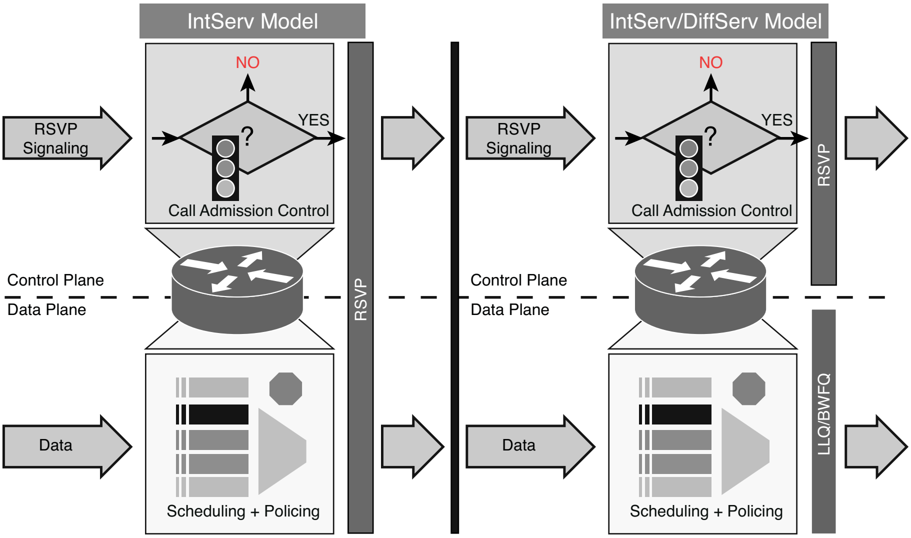


Figure 6-3 *RSVP Operational Models: IntServ and IntServ/DiffServ Models*

The right side of Figure 6-3, the IntServ/DiffServ model, is the one recommended for scalable deployment of RSVP as an AC mechanism for QoS. This model is now further divided into two designs, which are presented in the next sections:

- A basic RSVP design
- An advanced RSVP design that features application ID functionality

Basic RSVP Design (IntServ/DiffServ Model)

After the IOS RSVP agent has been configured, basic RSVP functionality is enabled by the **ip rsvp bandwidth interface** configuration command, which specifies how much bandwidth may be explicitly reserved by RSVP requests. (The default is 75 percent of a link's bandwidth.)

If all the priority traffic is RSVP enabled, the value specified in **the ip rsvp bandwidth** command and the **priority** low-latency queue (LLQ) command should match once the Layer 2 overhead of the priority queue bandwidth has been taken into account. However, if some priority traffic is not RSVP enabled, the sum of the values specified in the **ip rsvp bandwidth** command and in the complementary out-of-band call AC mechanism must not exceed the bandwidth value specified in the **priority** LLQ command.

RSVP should be enabled at the WAN edge of the network, including the router WAN interfaces on both sides of the WAN link, and at all WAN congestion points, including redundant links of different speeds. RSVP is not usually needed or used inside high-bandwidth campus networks, so the campus edge of the network (where end devices are attached to switches and not to routers) is not discussed here.

Configuring the IntServ/DiffServ RSVP model requires two additional interface configuration commands: **ip rsvp resource provider none** and **ip rsvp data-packet classification none.** These two additional commands instruct RSVP not to perform QoS operations that are handled within the data plane by DiffServ policies.

Example 6-1 shows a basic IntServ/DiffServ RSVP configuration.

Example 6-1 *RSVP IntServ/DiffServ Model Configuration*

```
 ! Configures basic IntServ/DiffServ RSVP on a WAN interface
Router(config)# interface Serial2/0
Router(config-if)# description CAMPUS-TO-BRANCH-SERIAL-T3-WITH-RSVP
Router(config-if)# bandwidth 44210
Router(config-if)# service-policy output WAN-EDGE-DIFFSERV-POLICY
 ! Attaches the DiffServ MQC policy to the interface
Router(config-if)# ip rsvp bandwidth 15000
 ! Specifies the amount of reservable BW (should match LLQ BW)
Router(config-if)# ip rsvp signalling dscp 24
 ! Marks RSVP signaling traffic to DSCP 24
Router(config-if)# ip rsvp data-packet classification none
 ! Enables IntServ/DiffServ by disabling RSVP for classification
Router(config-if)# ip rsvp resource-provider none
 ! Enables IntServ/DiffServ by disabling RSVP for scheduling
```

Advanced RSVP Design (IntServ/DiffServ Model)

An RSVP local policy provides a mechanism for controlling a reservation based on an application ID. Application IDs are mapped to RSVP local policies through the **ip rsvp policy identity** command. RSVP local policy identities are defined globally and are available to each interface for policy enforcement. Each identity can have one policy locator defined to match an application ID.

To give the user as much flexibility as possible in matching application policy locators to local policies, the RSVP local policy command accepts application ID match criteria in the form of UNIX-style regular expressions for the policy locator.

CUCM can set RFC 2872 application IDs for both voice and video flows as clusterwide service parameters. By default, the application IDs used by CUCM are AudioStream for voice flows and VideoStream for video flows (for both the audio and video components of a video stream). Therefore, the globally defined RSVP policy identities that match the CUCM application IDs for voice and video are (respectively) the following commands:

- **ip rsvp policy identity rsvp-voice policy-locator** .*AudioStream.*
- **ip rsvp policy identity rsvp-video policy-locator** .*VideoStream.*

In turn, local policies based on application IDs are applied to an interface using the **ip rsvp policy local identity** command. Example 6-2 shows an RSVP design featuring local policies for voice and video application IDs.

Example 6-2 *RSVP Local Policy with Application ID Configuration*

```
Router(config)# interface Serial2/0
Router(config-if)# description CAMPUS-TO-BRANCH-SERIAL-T3-WITH-RSVP
Router(config-if)# bandwidth 44210
Router(config-if)# service-policy output WAN-EDGE-DIFFSERV-POLICY
 ! Attaches the DiffServ MQC policy to the interface
Router(config-if)# ip rsvp policy local identity RSVP-VIDEO
Router(config-rsvp-local-if-policy)# maximum bandwidth group 12500
Router(config-rsvp-local-if-policy)# forward all
  ! Local RSVP policy to admit <12.5 Mbps of video flows
Router(config-if)# ip rsvp policy local identity RSVP-VOICE
Router(config-rsvp-local-if-policy)# maximum bandwidth group 2500
Router(config-rsvp-local-if-policy)# forward all
  ! Local RSVP policy to admit up to 2.5 Mbps of voice flows
 Router(config-if)# ip rsvp bandwidth 15000
  ! Specifies the amount of reservable BW (should match LLQ BW)
 Router(config-if)# ip rsvp signalling dscp 24
  ! Marks RSVP signaling traffic to DSCP 24
 Router(config-if)# ip rsvp data-packet classification none
  ! Enables IntServ/DiffServ by disabling RSVP for classification
 Router(config-if)# ip rsvp resource-provider none
  ! Enables IntServ/DiffServ by disabling RSVP for scheduling

! This section defines regular expressions to match RSVP AppIDs
Router(config)# ip rsvp policy identity RSVP-VIDEO policy-locator .*VideoStream.*
 ! RSVP AppIDs with the string "VideoStream" are
 ! associated with the RSVP-VIDEO local RSVP policy
Router(config)# ip rsvp policy identity RSVP-VOICE policy-locator .*AudioStream.*
 ! RSVP AppIDs with the string "AudioStream" are
 ! associated with the RSVP-VIDEO local RSVP policy
```

RSVP and LLQ

In a design using the RSVP IntServ/Diffserv model, RSVP is used as AC only (with the **ip rsvp bandwidth** command), and LLQ with its class-based rules (usually based on DiffServ code point [DSCP] markings) controls the queuing logic. Figure 6-4 shows how this operates.

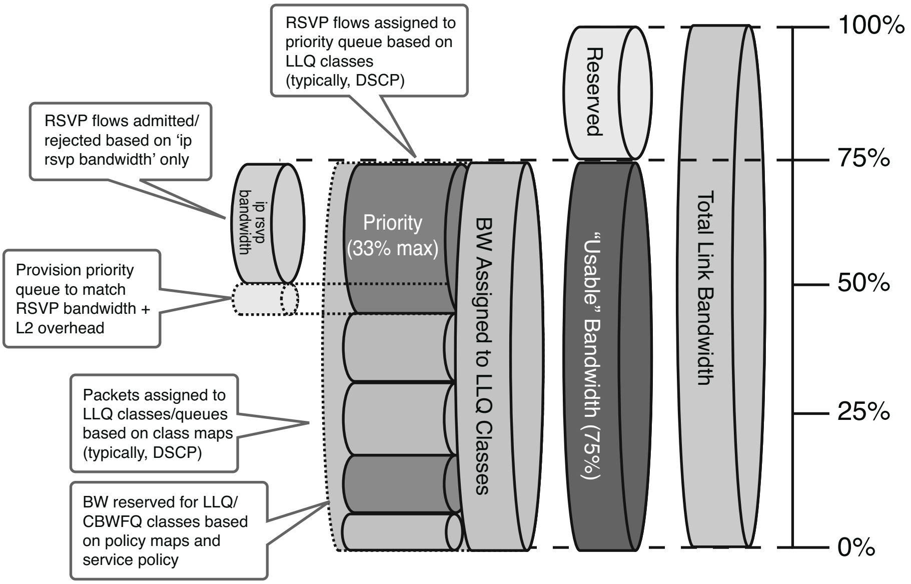

RSVP flows assigned to priority queue based on LLQ classes (typically, DSCP)
RSVP flows admitted/ rejected based on 'ip rsvp bandwidth' only
Provision priority queue to match RSVP bandwidth + L2 overhead
Packets assigned to LLQ classes/queues based on class maps (typically, DSCP)
BW reserved for LLQ/ CBWFQ classes based on policy maps and service policy
ip rsvp bandwidth
Priority (33% max)
BW Assigned to LLQ Classes
Reserved
"Usable" Bandwidth (75%)
Total Link Bandwidth
100%
75%
50%
25%
0%

Figure 6-4 *RSVP IntServ/DiffServ and LLQ*

The device where RSVP and LLQ are serviced does not control what bandwidth value is put into the original RSVP request; it merely acts on implementing the request. What value the endpoint, application, or proxy puts into the RSVP request (especially for video, where bandwidth requirements fluctuate during a session) may vary across different types of endpoints and applications.

LLQ and class-based queuing (CBWFQ) classes are configured as usual, and bandwidth is allocated to them on the interface. No bandwidth is reserved with the **ip rsvp bandwidth** command; this command acts as the AC logic that determines which flows are admitted or rejected. RSVP traffic assigned to queues is based on LLQ rules. If non-RSVP real-time applications are present, provision the **priority** command such that it can accommodate the RSVP and non-RSVP flows, and ensure that the non-RSVP flows are subject to some other form of AC to avoid oversubscription.

When a design using RSVP and application IDs is used for more granular bandwidth control over specific applications, RSVP and LLQ interwork as shown in Figure 6-5.

In general, LLQ and RSVP still work together as explained in Figure 6-4, but with the added proviso that RSVP flows are *not only* admitted against the value of the **ip rsvp bandwidth** command, but additionally against the local policy matching the flow's application ID (that is, the bandwidth stated in the **ip rsvp policy local identity** command block).

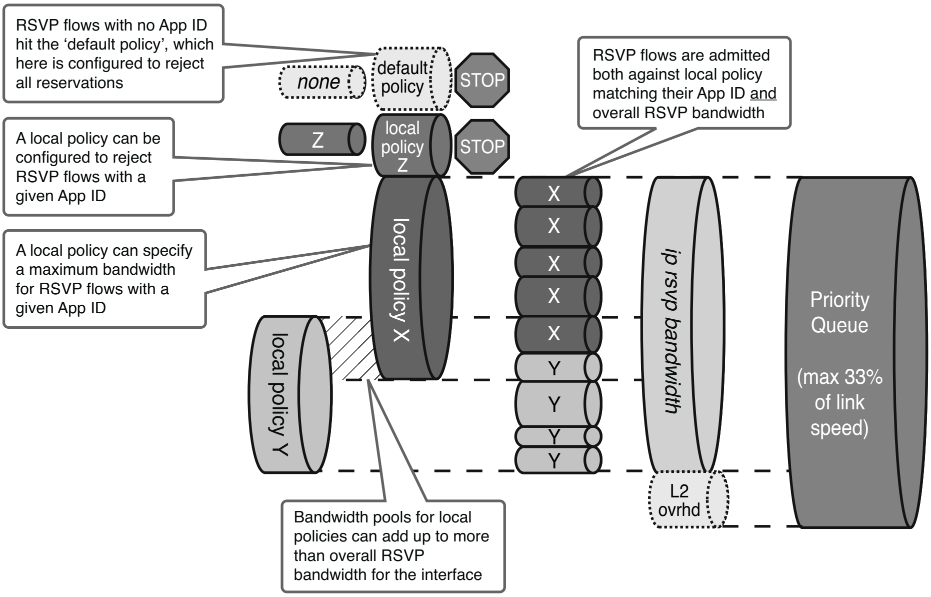

RSVP flows with no App ID hit the 'default policy', which here is configured to reject all reservations
none
default policy
STOP
A local policy can be configured to reject RSVP flows with a given App ID
Z
local policy Z
STOP
A local policy can specify a maximum bandwidth for RSVP flows with a given App ID
local policy X
local policy Y
RSVP flows are admitted both against local policy matching their App ID and overall RSVP bandwidth
X
X
X
X
X
Y
Y
Y
Y
ip rsvp bandwidth
L2 ovrhd
Priority Queue
(max 33% of link speed)
Bandwidth pools for local policies can add up to more than overall RSVP bandwidth for the interface

Figure 6-5 *RSVP IntServ/DiffServ with Application ID and LLQ*

Recommendations and Guidelines

With the explosion of rich media applications and traffic streams, Cisco recommends using the RSVP IntServ/DiffServ model with a router-based proxy device. This deployment model allows for the most efficient scaling of QoS policies along with dynamic network-aware AC for flows of varying and unpredictable bandwidth needs.

Summary

As part of the Medianet initiative, Cisco is continuing to expand RSVP functionality and making it more adaptable and efficient. RSVP is a signaling protocol, unlike the other QoS tools covered thus far in this book. It is a key network-aware technology to arbitrate bandwidth allocation between competing real-time traffic flows of widely varying bandwidth requirements (from a G.729 call to a high-definition multiscreen video stream).

"Call counting" AC methods work passably well when all real-time traffic streams have similar bandwidth requirements (such as G.729 and G.711 voice calls), but do not work efficiently when bandwidth requirements vary widely (as with voice and the many types of video).

RSVP operation is briefly reviewed in this chapter. Significant focus is given to the designs utilizing RSVP solely as an AC mechanism (rather than on the all the capabilities of the entire protocol stack) in a combined IntServ/DiffServ deployment model. This design allows for the most efficient and scalable use of RSVP to provide AC for rich media traffic flows.

RSVP is also used by the Medianet features Mediatrace and media monitoring. However, these features do not use the AC and bandwidth reservation aspects of RSVP, only the media monitoring information. This is discussed further in Chapter 8, "Medianet."

Further Reading

RSVP for Medianet

Medianet WAN/VPN QoS Design At-a-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoswanvpnaag.html

Enterprise Medianet Quality of Service Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html

Medianet WAN Aggregation QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html#wp129507

RSVP Technology

Configuring RSVP Support for LLQ: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_rsvp/configuration/15-2mt/rsvp_sprt_llq.html

Configuring RSVP Agent: http://www.cisco.com/en/US/docs/ios-xml/ios/voice/cminterop/configuration/15-2mt/vc-rsvp-agent.html

Configuring RSVP: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_rsvp/configuration/15-2mt/config-rsvp.html

RSVP Scalability Enhancements: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_rsvp/configuration/12-4/rsvp-scalability.html

RSVP Refresh Reduction and Reliable Messaging: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_rsvp/configuration/15-2mt/rsvp-messaging.html

RSVP Local Policy Support: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_rsvp/configuration/15-2mt/rsvp_local_plcy_sprt.html

This page intentionally left blank

Chapter 7

QoS in IPv6 Networks

This chapter covers the following topics:

- IPv6 addressing and classification
- How QoS tools function, and should be configured, in the presence of IPv6 traffic

All the quality of service (QoS) discussions in the first seven chapters of this book pertain to both IPv4 and IPv6 networks. When IPv6 traffic is present, it is often mixed with IPv4 traffic (in what will likely be a lengthy transition period in the industry). Therefore, there are a few considerations to ensure that the QoS tools are appropriately configured for both IPv4 and IPv6 traffic on your network.

IPv6 and QoS Overview

IPv6 was developed by the Internet Engineering Task Force (IETF) to solve the challenges of limited address space of IPv4. IETF Request For Comments (RFC) 2460 (*Internet Protocol Version 6, IPv6, Specification*) and RFC 3697 (*IPv6 Flow Label Specification*) are the key standards specifying IPv6.

IPv6 is an extension of IPv4 capabilities, but the two protocols are not interoperable and have to be tunneled or interworked to allow IPv4 and IPv6 devices to communicate on the same network. The most common method of interworking is with dual protocol-stack capable devices.

IPv6 support for QoS features is largely seamless, and there are only a small number of considerations to be aware of, including the following:

- Class-based QoS features are supported for IPv6, but many older features are not.
- IPv6 headers are much larger than IPv4 headers, and therefore the bandwidth consumption for small packet streams is higher.

- There are minor syntactical changes in some configuration commands (for example, the **ip** keyword [or rather, the omission of this keyword] in the **set** and **match** commands).

QoS Tools for IPv6

All the same QoS features and tools discussed in the book so far apply equally to IPv4 and IPv6 traffic, including the following:

- Packet classification and marking
- Policing and shaping
- Congestion management and avoidance
- Bandwidth reservation

QoS Feature Support for IPv6

QoS features supported for IPv6 environments include class-based packet marking and classification, queuing (low-latency queuing, or LLQ), class-based traffic shaping, weighted random early detection (WRED), and class-based policing. All QoS features supported for IPv6 are configured using the Modular QoS command line (MQC). Older, pre-MQC features such as Compressed RTP (cRTP), Committed Access Rate (CAR) and Priority Queuing (PQ) are not supported by IPv6.

It is recommended to check IPv6 support for a specific feature to confirm that it is ready for deployment on your network. The Cisco document *Cisco IOS IPv6 Feature Mapping* referenced in the "Further Reading" section of this chapter provides details.

Packet Headers, Classification, and Marking

IPv4 and IPv6 headers differ, as shown in Figure 7-1. The significant differences include the following:

- A much larger Address field (128 bits or 16 bytes)
- A 20-bit Flow Label field

The larger Address field impacts QoS deployment in that the IPv6 header (40 bytes) is double the size of the typical IPv4 header (generally 20 bytes, but could be larger if some seldom-used options are turned on) and therefore has a significant bandwidth impact on flows comprising very small packets, such as a G.729 voice stream, where the packet payload is only 20 bytes. You should review and enlarge the bandwidth allocation of classes with small packet flows using IPv6 as the transport protocol.

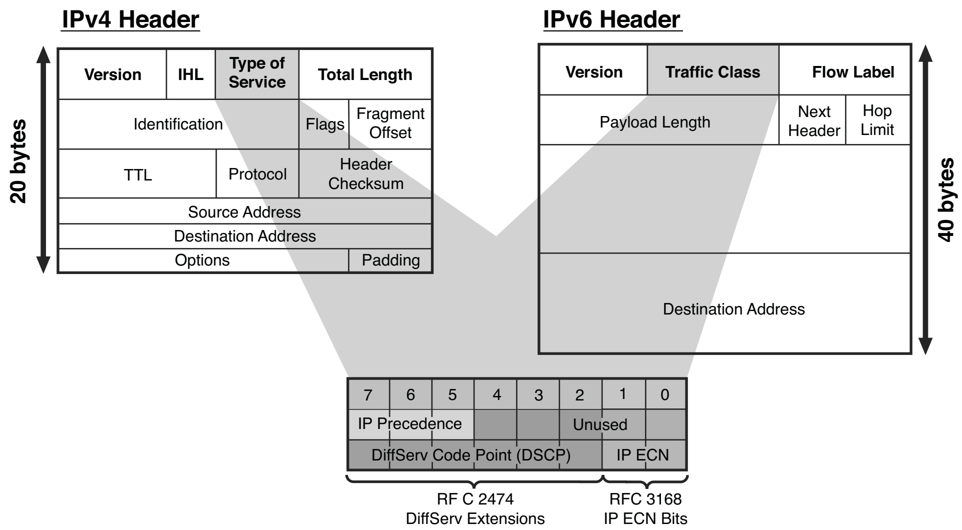


Figure 7-1 *IPv4 and IPv6 Packet Headers*

Packet Classification

The **match** class-based command to classify packets has an optional **ip** keyword, as shown in Example 7-1. When the **ip** keyword is used, the classifying action is applied *only* to IPv4 packets. If the **ip** keyword is omitted, the classifying action is applied to both IPv4 and IPv6 packets. Cisco recommends that you configure your network to omit the **ip** keyword from these commands.

Example 7-1 *Classifying IPv4 and IPv6 Packets*

```
! This pair of class-maps matches on IPv4 and/or IPv6 DSCP values
Router(config)# class-map MULTIMEDIA-CONFERENCING-IPV4
Router(config-cmap)# match ip dscp af21
! Classifies only IPv4 traffic marked with DSCP AF21

Router(config)# class-map MULTIMEDIA-CONFERENCING-ALL
Router(config-cmap)# match dscp af21
 ! Classifies both IPv4 and IPv6 traffic marked with DSCP AF21


! This pair of class-maps matches on IPv4 and/or IPv6 IPP values
Router(config)# class-map MULTIMEDIA-CONFERENCING-ALL
Router(config-cmap)# match ip precedence 5
 ! Classifies only IPv4 traffic marked with IPP 5

Router(config)# class-map MULTIMEDIA-CONFERENCING-ALL
Router(config-cmap)# match precedence 5
 ! Classifies both IPv4 and IPv6 traffic marked with IPP 5
```

Packet Marking

Currently, IPv6 provides support for QoS marking via the Traffic Class field in the IPv6 header. This field is shown in Figure 7-1 and was also briefly discussed in the section, “Packet Headers” in Chapter 1. The layout and interpretation of the IPv6 Traffic Class field is the same as the Type of Service (ToS) field in the IPv4 header so that IPv6 packets can be marked with class of service (CS1 to CS7) and differentiated services code point (DSCP) value exactly as IPv4 packets are marked.

In the **set** class-based command to mark packets, there is an optional **ip** keyword, as shown in Example 7-2. When the **ip** keyword is used, the marking action is applied *only* to IPv4 packets. If the **ip** keyword is omitted, the marking action is applied to both IPv4 and IPv6 packets. Cisco recommends that you configure your network to omit the **ip** keyword from these commands.

Example 7-2 *Marking IPv4 and IPv6 Packets*

```
Router(config)# policy-map IPv4-IPv6-MARKING
Router(config-pmap)# class MULTIMEDIA-CONFERENCING-IPv4
Router(config-pmap-c)# set ip dscp af21
 ! Marks only IPv4 traffic to DSCP AF21
Router(config-pmap)# class MULTIMEDIA-CONFERENCING-ALL
Router(config-pmap-c)# set dscp af21
 ! Marks both IPv4 and IPv6 packets to DSCP AF21
Router(config-pmap)# class-map MULTIMEDIA-CONFERENCING-IPv4
Router(config-pmap-c)# set ip precedence 5
 ! Mark only IPv4 traffic to IPP 5
Router(config-pmap)# class-map MULTIMEDIA-CONFERENCING-ALL
Router(config-pmap-c)# set precedence 5
 ! Marks both IPv4 and IPv6 traffic to IPP 5
```

IPv4 and IPv6 interworking can be accommodated via various tunneling methods. When using these tunnels, discovering and classifying IPv6 flows appropriately is not as straightforward as when IPv4 and IPv6 packet flows exist natively side by side in a dual-stack environment.

When flows exist natively, IPv6 traffic can be singled out (and therefore classified and marked) by using the **match** command shown in Example 7-3.

Example 7-3 *Classifying IPv6 Flows*

```
Router(config)# class-map IPV6-TRAFFIC
Router(config-cmap)# match protocol ipv6
```

IPv6 traffic tunneled or encapsulated inside IPv4 can be singled out using Network Based Application Recognition (NBAR2) commands, as shown in Example 7-4.

Example 7-4 *Classifying Tunneled IPv6 Flows*

```
Router(config)# class-map IPV6-TEREDO
Router(config-cmap)# match protocol teredo-ipv6-tunneled
 ! Matches IPv6 traffic tunneled with the Teredo method

Router(config)# class-map IPV6-ISATAP
Router(config-cmap)# match protocol isatap-ipv6-tunneled
 ! Matches IPv6 traffic tunneled with the ISATAP method

Router(config)# class-map IPV6-6TO4
Router(config-cmap)# match protocol sixtofour-ipv6-tunneled
 ! Matches IPv6 traffic tunneled with the 6to4 method
```

Policing and Shaping

The class-based policing and shaping configurations discussed in Chapter 4, “Policing, Shaping, and Markdown Tools,” can be used equally for IPv4 and IPv6 traffic. If IPv6 traffic is classified into a separate class, the IPv6 traffic can be policed or shaped using different parameters than IPv4 traffic.

The class-based queuing and packet-dropping configurations discussed in Chapter 5, “Congestion Management and Avoidance Tools,” can be used equally well for IPv4 and IPv6 traffic. If IPv6 traffic is classified into a separate class, then the IPv6 traffic can be queued or dropped using different parameters than IPv4 traffic.

Recommendations and Guidelines

Cisco recommends that you omit the **ip** keyword from the **set** and **match** commands to provide generic QoS treatment to all packets regardless of whether the transport protocol is IPv4 or IPv6.

In general, the QoS commands do not differentiate between IPv4 and IPv6 traffic, and the policing, shaping, queuing, and dropping configurations in your network apply equally, and without syntax changes, to both types of traffic. This is the most flexible and all-inclusive way to configure your network.

If you have a specific need to single out IPv6 traffic because it requires different treatment from the otherwise equivalent IPv4 traffic, you can use the NBAR2 **match protocol** commands to classify IPv6 traffic into a separate class and apply specific policies to this traffic separate from the IPv4 traffic policies. This is transient solution, though, because the mix of IPv4 and IPv6 traffic will likely continue on your network, and bandwidth allocations and treatments for separate classes of traffic may need to be reviewed frequently.

IPv6 has a significantly larger header overhead than IPv4, so be sure to review bandwidth allocation for classes of traffic containing small packets and using either IPv6 natively or IPv6 tunneled inside IPv4.

Summary

Many networks are in a lengthy transition period from IPv4 to IPv6 protocol transport. The mix of traffic using both protocols on the average network is expected to be present into the foreseeable future. Specific network deployment actions must be taken to accommodate both types of traffic on the same network because they do not natively interoperate. The nature of many of these actions—such as routing, tunneling, and dual-stack devices—is beyond the scope of this book, and the content of this chapter focuses more narrowly on the impact of IPv6 on the QoS features deployed in your network, including the following:

- The transport protocol is largely managed seamlessly by the syntax of the QoS features with only a few minor syntax changes, such as the **ip** keyword in the **set** and **match** commands.
- The review of adequate bandwidth allocation for small packet IPv6 streams.
- The impact of tunneling technologies on IPv6 traffic classification.
- Both IPv4 and IPv6 include an 8-bit ToS field so that CS and DSCP packet markings automatically interwork.

Further Reading

IPv6 QoS At-a-Glance: http://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8026004d.pdf

Implementing QoS for IPv6: http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-qos.html

Application Visibility and Control for IPv6: http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6553/aag_c45-665915.pdf

IPv6 Configuration Guide, Cisco IOS Release 15.2MT: http://www.cisco.com/en/US/docs/ios-xml/ios/ipv6/configuration/15-2mt/ip6-qos.html

Cisco IOS IPv6 Command Reference: http://www.cisco.com/en/US/docs/ios/ipv6/command/reference/ipv6_book.html

Cisco IOS IPv6 Feature Mapping: http://www.cisco.com/en/US/docs/ios-xml/ios/ipv6/configuration/15-2mt/ip6-roadmap.html

Chapter 8

Medianet

This chapter covers the following topics:

- Medianet framework
- Medianet features such as Performance Monitor, Mediatrace, and flow metadata

The discussions in the first seven chapters covered specific quality of service (QoS) topics and tools in the by-now-familiar categories of classification, marking, policing, shaping, congestion management, congestion avoidance, and bandwidth reservation for IPv4 and IPv6 networks. Medianet is a framework of a larger scope, encompassing both QoS and non-QoS topics and considerations. This chapter examines the Medianet architecture, with particular focus on the aspects of Medianet specific to QoS configuration, monitoring, and control in a network.

An Introduction to Medianet

Medianet is an end-to-end architectural network framework comprising advanced tools, technologies, and devices to deliver pervasive rich-media experiences with the best quality of experience (QoE). It is a critical component in a modern network where deployment of rich-media applications containing multiple traffic flows are accelerating and users expect optimal experiences from their network and devices.

The trends in voice, video, and media applications necessitating Medianet technologies are shown in Figure 8-1. Salient trends affecting network design and administration include the following:

- Applications that no longer fit into simple voice, video, or data buckets, but that often consist of various combinations of these depending on what aspect of the application the user is busy with at that moment
- The emergence of unmanaged applications, brought in by a technically enabled workforce that may or may not have approval from corporate network administrators

- Ubiquitous mobile devices of mixed personal and corporate use, and the users bringing them into the corporate environment expect them to work at all times

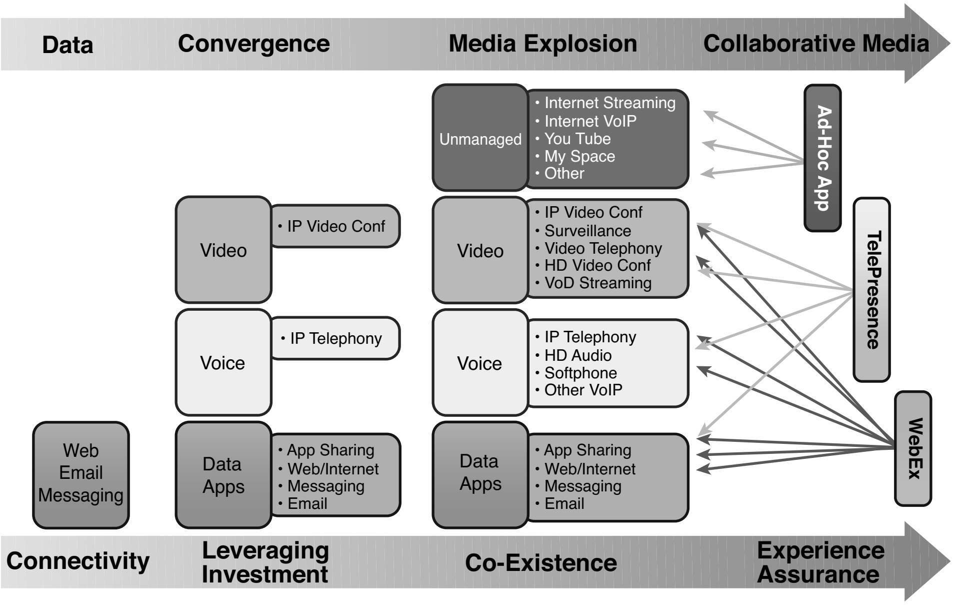


Figure 8-1 *Medianet Application Convergence Evolution*

Cisco WebEx is a prime example of an application with mixed traffic types, providing text, audio, instant messaging, application sharing, and desktop video conferencing services to all meeting participates, regardless of their location. Instead of a cumbersome setup of a video conference call, applications such as WebEx greatly simplify the process, and video capability is added to the conference just as easily as any other type of media, such as audio. The complexity these applications present to the network administrator relates to application classification: Because media applications include voice, video, and data subcomponents, the question of how to mark and provision a given media application becomes more difficult and blurry. The Medianet architecture and tools address this complexity for configuration, monitoring, and troubleshooting.

The Medianet architecture has the following characteristics:

- **Endpoint aware:** The network automatically detects and configures media endpoints.
- **Media aware:** The network detects and optimizes different traffic, media and application types (for example, TelePresence, video surveillance, desktop collaboration, and streaming media) to deliver an optimal user experience.
- **Network aware:** The network detects and responds to changes in device, connection, traffic, and service availability.

Note You can find more information about the overall Medianet benefits and solution overviews at http://www.cisco.com/go/medianet.

Medianet Architecture and Framework

The Medianet framework enables the *network* to become application and media aware so that the network can intelligently apply critical network services. The Medianet framework also enables the rich-media *applications* to become network aware, enabling them to adapt dynamically to network conditions. Integration of network capabilities with applications' capabilities provides much better end-user services than applications that run blindly on *pipes*, or much better than a network that has no knowledge of what the applications are trying to accomplish for their end users.

The Medianet framework encompasses the endpoints, access, distribution, and core layers of the network. The boundary of the network extends to the endpoints, and tight integration is achieved between network services and the rich-media applications. Figure 8-2 shows the Medianet architectural framework.

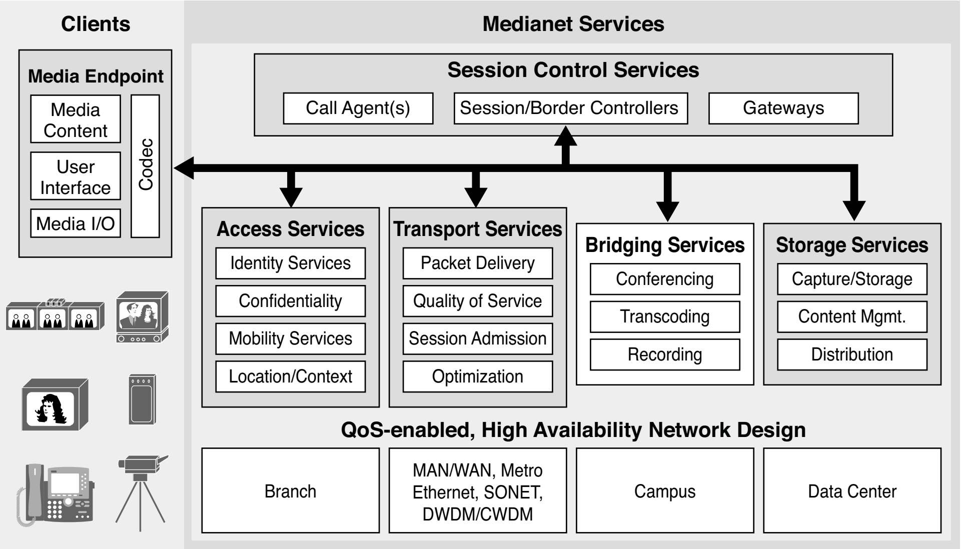


Figure 8-2 *Medianet Architecture and Framework*

The Medianet architecture includes the following elements:

- **Endpoints:** A wide variety of endpoint devices must be accommodated, including video surveillance cameras, desktop client applications, video and voice endpoints, and immersive TelePresence sessions

- **Network services:** Capabilities embedded in endpoints, proxies, routers, and switches that automate configurations, monitor network behavior, process media streams according to policies, and propagate information throughout the network to provide shared services and consistent treatment, including the following:
  - Access services
  - Transport services
  - Bridging services
  - Storage services
- **Cloud Services:** Extended Medianet capabilities beyond the enterprise to enable optimal service provider-to-business, business-to-business, and business-to-consumer media experiences

Medianet Features and Capabilities

Medianet takes a layered approach to network architecture and to providing QoS features and capabilities in the network, as depicted in Figure 8-3.

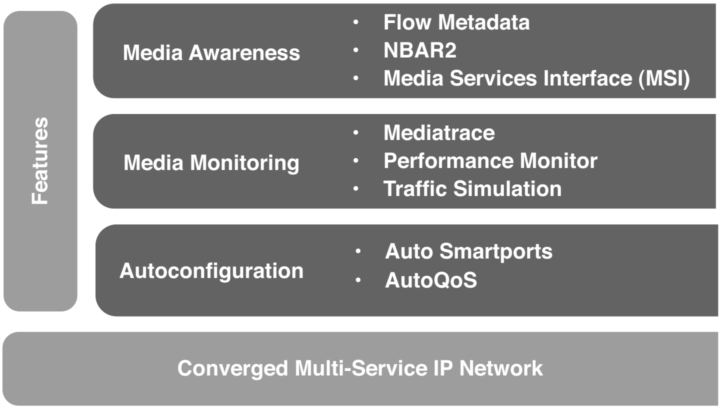


Figure 8-3 *Medianet: A Layered Feature Approach*

The Medianet features and capabilities within the framework consist of the following layers:

- **Autoconfiguration:** Provides for the simplified and ready-to-use deployment of rich-media endpoints such as IP surveillance cameras, IP phones, and digital media players
- **Media Monitoring:** Provides tools to enable capacity planning, performance monitoring, and dynamic troubleshooting to lower the network cost of ownership

- **Media Awareness:** Provides the ability to differentiate applications to enable end-to-end service assurance and optimal user experience

Each of these feature layers is discussed, from a functionality point of view, in more detail in the following sections.

Note To learn more about the latest Medianet capabilities and which products and software releases contain which features, see the Medianet data sheet at http://www.cisco.com/en/US/prod/collateral/routers/ps10536/data_sheet_c78-612429.pdf.

Autoconfiguration

Autoconfiguration capabilities facilitate the deployment (configuration and registration) of rich-media endpoints and reduce the ongoing operational costs of managing administrative changes and QoS features in the campus network.

Auto Smartports

The Smartports macros on the access switches provide static (the Smartports feature) and dynamic (the Auto Smartports feature) configurations to port or VLAN interfaces. Certain Smartports macros are predefined, or built in, within Catalyst IOS switch software, and can do the following:

- Configure ports to connect to Cisco IP phones (which includes the configuration and execution of AutoQoS VoIP on the switch port)
- Configure ports to connect to Cisco digital media players (DMPs)
- Configure ports to connect to Cisco IP Video Surveillance (IPVS) cameras
- Configure a connection to other Cisco Catalyst switches (using 802.1Q trunking)
- Configure other Cisco routers and Cisco wireless access points (among other devices)

AutoQoS

The AutoQoS features simplify access switch and access router QoS deployment. The AutoQoS feature set automatically provisions QoS best-practice designs for voice, IP-based video applications (such as IPVS, Cisco TelePresence, conferencing applications, and streaming video applications), and multiple types of data applications.

An administrator can automatically provision the best-practice QoS designs via a single interface-level command that corresponds to the endpoint type that the interface is connecting to. Examples of this provisioning include the following:

- **auto qos voip** [**cisco-phone** | **cisco-softphone** | **trust**]
- **auto qos trust** {**cos** | **dscp**}
- **auto qos video** [**cts** | **ip-camera**]
- **auto qos classify** {**police**}

The AutoQoS features and capabilities were covered in Chapter 2, "IOS-Based QoS Architectural Framework and Syntax Structure," and are also detailed in Appendix A, "AutoQoS for Medianet."

Media Monitoring

The Medianet media monitoring capabilities enable you to see in real time the actual performance of the network, either on an actual application flow or session or on a simulated one to gauge the impact of a potential application on your network. Media monitoring enhances visibility into traffic flows on the network and helps reduce operating costs with faster troubleshooting of applications using rich-media streams.

Media monitoring consists of three features:

- Mediatrace
- Performance Monitor
- IPSLA Video Operation (IPSLA VO) (traffic generation and simulation)

These three features form a suite of tools to enable the network operator to perform media performance monitoring and aids in significantly simplifying troubleshooting.

Mediatrace

Mediatrace is similar in concept to the familiar traceroute capability that prints out a hop-by-hop log of the path that a particular (simulated) packet follows through a network. Mediatrace is enhanced for real-time applications and returns hop-by-hop information from the intermediate network elements that a particular traffic flow is following between its source and destination points in a network. At every hop, the information returned to the network operator provides a view of the treatment experienced by the flow at this hop, including real-time performance statistics.

The traceroute feature sources the probes from an IP address that belongs to the router, usually the outbound interface toward the destination. The Mediatrace feature has the extra capability that the source address can be the actual address of the camera (or device). This proves helpful when multiple paths (often used for load balancing) exist from the router to the destination. The Mediatrace feature can therefore tell you both the inbound and outbound interfaces, by name, at each hop taken by that specific flow, provided each hop supports Mediatrace. If not, a combination of the Mediatrace and traceroute information could be helpful: Traceroute can discover all the hops between two points in the network (as long as only a single path exists), and Mediatrace can discover the actual path but may not report all hops.

This wealth of information allows the network operator to see quickly where packets may be retagged (and QoS services therefore lost) or where packets are dropped or where congestion may exist. Like the traceroute facility, the Mediatrace feature is initiated from a single point in the network (where the operator is). From there the path of the flow is traced, each element along the path is queried for its statistics, and all the information is returned in a single display to the initiating element where the operator can easily analyze it. Once the network is configured for Mediatrace, the network operator does not require command-line (CLI) access to the network elements any longer and can simply request statistics for a particular flow.

Mediatrace Configuration

A Mediatrace poll is initiated from a particular element in the network, usually an access switch. The Mediatrace Responder capability must be enabled on each network element that you want to collect flow information from, such as a router or a switch. (This is a one-time configuration and, once configured, the service is always running on this device.) The Mediatrace Initiator must be configured on the network element where you want to initiate the poll from and collect the output for display.

The Mediatrace Responder CLI is enabled at the global level of each element that you require to report information when a poll is sent. The Mediatrace Initiator CLI is enabled at the global level on the network element where you want to send the poll from and collect the responses. Figure 8-4 shows a sample network configuration with Mediatrace Responders enabled on all the switches and routers in a possible Medianet path, and the Mediatrace Initiator enabled on one of the edge routers.

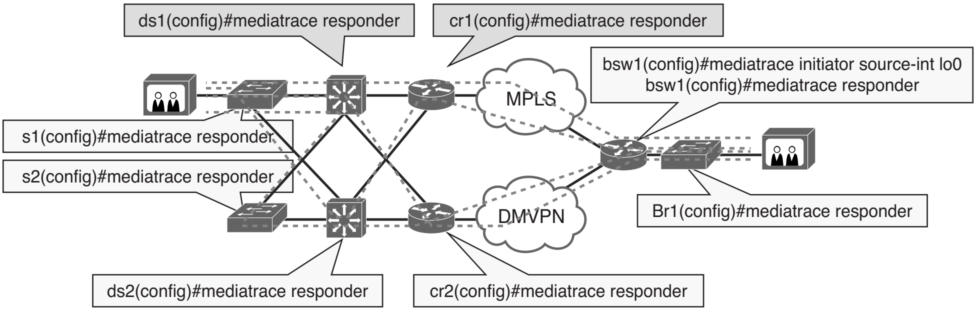

ds1(config)#mediatrace responder
cr1(config)#mediatrace responder
bsw1(config)#mediatrace initiator source-int lo0
bsw1(config)#mediatrace responder
MPLS
s1(config)#mediatrace responder
s2(config)#mediatrace responder
DMVPN
Br1(config)#mediatrace responder
ds2(config)#mediatrace responder
cr2(config)#mediatrace responder

Figure 8-4 *Mediatrace Configuration*

Mediatrace Operation

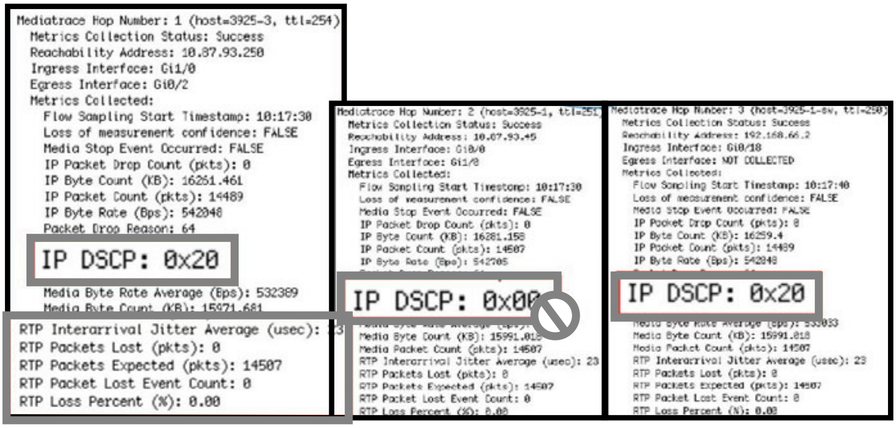
A poll for network statistics is initiated from the network element configured as the Mediatrace Initiator by using the **mediatrace poll** command. The command has many options and capabilities; for details, see the *Mediatrace Command Reference* document in the “Further Reading” section later in this chapter.

Example 8-1 demonstrates an example of a poll for gathering performance monitoring information (designated by the **perf-mon** option on the command) for a particular flow.

Example 8-1 *Mediatrace Performance Monitor Poll*

```
 ! This command initiates a performance monitor request for the
 ! flow path from source IP address 10.87.93.162 to destination
 ! IP address 10.87.80.11
CAT3750# mediatrace poll path source 10.87.93.162 destination 10.87.80.11 perf-mon
Started the data fetch operation.
Waiting for data from hops.
This may take several seconds to complete...
Data received for hop 0
Data received for hop 1
Data received for hop 2
Data fetch complete.
```

Figure 8-5 provides example of the output gathered and displayed by the **mediatrace poll** command. In this output, you see results for three of the hops in the path: in hop 1 and 3 the differentiated services code point (DSCP) of the packet is marked with the value 0x20, but at hop 2 it is marked as 0, which could signify a lack of QoS treatment at this element and likely constitutes a misconfiguration on this element or one upstream from it. With this troubleshooting information, a network drop in QoS treatment on a real-time flow can be quickly isolated and fixed.



```
Mediatrace Hop Number: 1 (host=3925-3, ttl=254)
  Metrics Collection Status: Success
  Reachability Address: 10.87.93.250
  Ingress Interface: Gi1/0
  Egress Interface: Gi0/2
  Metrics Collected:
    Flow Sampling Start Timestamp: 10:17:30
    Loss of measurement confidence: FALSE
    Media Stop Event Occurred: FALSE
    IP Packet Drop Count (pkts): 0
    IP Byte Count (KB): 16261.461
    IP Packet Count (pkts): 14489
    IP Byte Rate (Bps): 542048
    Packet Drop Reason: 64
 IP DSCP: 0x20
    Media Byte Rate Average (Bps): 532389
    Media Byte Count (KB): 15971.681
RTP Interarrival Jitter Average (usec): 23
RTP Packets Lost (pkts): 0
RTP Packets Expected (pkts): 14507
RTP Packet Lost Event Count: 0
RTP Loss Percent (%): 0.00
```

```
Mediatrace Hop Number: 2 (host=3925-1, ttl=251)
  Metrics Collection Status: Success
  Reachability Address: 10.87.93.45
  Ingress Interface: Gi0/0
  Egress Interface: Gi1/0
  Metrics Collected:
    Flow Sampling Start Timestamp: 10:17:30
    Loss of measurement confidence: FALSE
    Media Stop Event Occurred: FALSE
    IP Packet Drop Count (pkts): 0
    IP Byte Count (KB): 16281.158
    IP Packet Count (pkts): 14507
    IP Byte Rate (Bps): 542705
IP DSCP: 0x00
    Media Byte Count (KB): 15991.018
    Media Packet Count (pkts): 14507
    RTP Interarrival Jitter Average (usec): 23
    RTP Packets Lost (pkts): 0
    RTP Packets Expected (pkts): 14507
    RTP Packet Lost Event Count: 0
    RTP Loss Percent (%): 0.00
```

```
Mediatrace Hop Number: 3 (host=3925-1-sw, ttl=250)
  Metrics Collection Status: Success
  Reachability Address: 192.168.66.2
  Ingress Interface: Gi0/18
  Egress Interface: NOT COLLECTED
  Metrics Collected:
    Flow Sampling Start Timestamp: 10:17:40
    Loss of measurement confidence: FALSE
    Media Stop Event Occurred: FALSE
    IP Packet Drop Count (pkts): 0
    IP Byte Count (KB): 16259.4
    IP Packet Count (pkts): 14489
    IP Byte Rate (Bps): 542848
IP DSCP: 0x20
    Media Byte Rate Average (Bps): 533033
    Media Byte Count (KB): 15991.018
    Media Packet Count (pkts): 14507
    RTP Interarrival Jitter Average (usec): 23
    RTP Packets Lost (pkts): 0
    RTP Packets Expected (pkts): 14507
    RTP Packet Lost Event Count: 0
    RTP Loss Percent (%): 0.00
```

Figure 8-5 *Sample Mediatrace Output*

The box at the bottom left of Figure 8-5 highlights some of the large number of performance metrics returned by the **mediatrace poll** command, including a plethora of Real-time Transport Protocol (RTP) statistics.

Note For details of all the metrics returned by this command, see the *Media Monitoring Configuration Guide* document in the "Further Reading" section later in this chapter. (Within this document, look at the section on "Mediatrace," and within that at the section look for "Metrics That You Can Collect Using Cisco Mediatrace.")

The Mediatrace capability uses the Resource Reservation Protocol (RSVP) as a transport protocol. There is, however, no concept of call admission control (CAC) or bandwidth reservation here; the RSVP envelope is merely used as a transport protocol for the media monitoring information, meaning that the rest of the capabilities usually associated with RSVP are not used here. Note that RSVP does not have to be explicitly enabled in the configuration of the network nodes; the command **mediatrace responder** automatically turns on the RSVP capability for its own purposes.

With Cisco Prime Collaboration Manager (CPCM), you can also get the results in a graphic form. This is useful for network operators who do not have CLI access to any (or all) of the network elements in the flow path. As long as the basic Mediatrace feature (Responder and Initiator) is turned on by someone with CLI access, the network operator can do performance polling and troubleshooting from the CPCM console without needing further CLI access to the intermediate network elements. In addition, the amount of data generated by one flow over several hops could be overwhelming from CLI access.

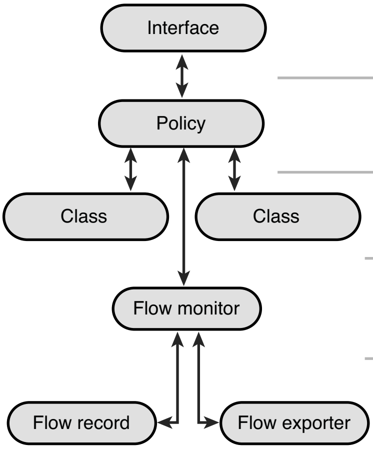
Performance Monitor

The second tool in the Media Monitoring toolbox is Performance Monitor. This feature enables you to see in real time the actual performance of the network. Whereas Mediatrace is more of a problem isolation tool, Performance Monitor is a monitoring tool allowing visibility into flows regardless of whether a problem has been reported on the network.

Proactive performance monitoring is especially important for video traffic because high-quality interactive video traffic is highly sensitive to network issues. Performance monitoring is also highly useful for gauging the impact of new or planned applications on the network.

Performance Monitor leverages Cisco Flexible NetFlow (FnF) and the configuration is therefore very similar. Cisco FnF provides statistics on packets flowing through a router and is a standard tool for acquiring IP operational data from IP networks.

Figure 8-6 shows an overview of the configuration elements of the Performance Monitor feature.



Attach a Performance Monitor service policy to an interface using **the service-policy type performance-monitor** command.

Create a Performance Monitor policy using the **policy-map type performance-monitor** command. This command is used to associate the flow monitor and one or more classes with the policy.

Create one or more classes using the **class-map** command. This command is used to specify the filtering criteria.

Create a Performance Monitor flow monitor using the **flow monitor type performance-monitor** command. This command is used to associate a flow record and an optional flow exporter with the flow monitor.

Create a Performance Monitor flow record using the **flow record type performance-monitor** command. Create a flow exporter using the flow exporter command. The flow record is used to specify match and collect fields, and the exporter is used to specify the destination for exporting data.

Figure 8-6 *Performance Monitor Configuration*

Example 8-2 shows a simple Performance Monitor configuration.

Example 8-2 *Performance Monitor Configuration*

```
 ! This configuration monitors TCP metrics on HTTP traffic
 ! Define the access list to match HTTP traffic
Router(config)# ip access-list extended HTTP
Router(config-ext-nacl)#)# permit tcp any any eq www

 ! Classify all HTTP traffic into the class TCP-CLASS
Router(config)# class-map match-all TCP-CLASS
Router(config-cmap)# match access-group name HTTP

 ! Define the default NetFlow capabilities in the policy
Router(config)# policy-map type performance-monitor TCP-METRIC
Router(config-pmap-c)# class TCP-CLASS
Router(config-pmap-c)# flow monitor inline
Router(config-pmap-c)# record default-tcp

 ! Attach the policy to monitor performance on the interface
Router(config)# interface GigabitEthernet0/2
Router(config-if)# service-policy type performance-monitor input TCP-METRIC
Router(config-if)# service-policy type performance-monitor output TCP-METRIC
```

In addition to NetFlow collectors and analyzers (where results can be viewed in GUI form), the output of the Performance Monitoring metrics can also be seen by using the

CLI command **show performance monitor status**. Output can additionally be gathered by using Simple Network Management Protocol (SNMP) tools collecting the NetFlow Management Information Base (MIB) information.

IPSLA Video Operation (Traffic Simulator, IPSLA VO)

The Cisco IOS IP Service Level Agreement (SLA) is a standard Cisco IOS Software feature that has been available for many years. It allows you to understand IP service levels by performing active monitoring of network performance. It can be used for network troubleshooting, network readiness assessment, and health monitoring.

Cisco IP SLA Video Operation (IPSLA VO) is an enhancement to the standard IP SLA feature set that allows for the simulation of various types of video traffic on the network. This enhancement helps to assess the impact and service levels of the video traffic. Devices that can act as a source or a responder for an IPSLA VO are limited to Cisco routers and switches that are capable of providing platform-assisted video traffic generation and reflection.

An IPSLA VO differs from other IP SLA operations in that all traffic is one way only, with a responder required to process the sequence numbers and time stamps locally and to wait for a request from the source before sending the calculated data back.

Example 8-3 shows the salient portion of the IPSLA VO configuration.

Example 8-3 *IPSLA VO Configuration*

```
! This configuration turns on IP SLA operation number 10
Router(config-term)# ip sla 10
!
! A TelePresence profile is enabled for this IPSLA VO
Router(config-ip-sla)# video 192.168.2.17 997 source-ip 192.168.2.16 source-port 555
profile telepresence
```

Table 8-1 summarizes the video traffic profiles that exist as of Cisco IOS Release 15.2(2)T.

Table 8-1 *Video Traffic Profiles*

Profile	Device	Traffic Characteristics
iptv	IP television	2.6 Mbps
ipvsc	IP video surveillance camera	2.2 Mbps
telepresence	Cisco TelePresence 1080P	6.6 Mbps
CP-9900-CIF-15-384kbps	Cisco CP-9900 Phone	CIF 15 fps 384 Kbps
CP-9900-CIF-30-1000kbps	Cisco CP-9900 Phone	CIF 30 fps 1000 Kbps
CP-9900-QCIF-10-79kbps	Cisco CP-9900 Phone	QCIF 10 fps 79 Kbps

Profile	Device	Traffic Characteristics
CP-9900-QCIF-15-99kbps	Cisco CP-9900 Phone	QCIF 15 fps 99 Kbps
CP-9900-QCIF-30-249kbps	Cisco CP-9900 Phone	QCIF 30 fps 249 Kbps
CP-9900-VGA-15-1000kbps	Cisco CP-9900 Phone	VGA 15 fps 1000 Kbps
CP-9900-VGA-30-1000kbps	Cisco CP-9900 Phone	VGA 30 fps 1000 Kbps
CTS-1080P-Best	Cisco TelePresence 1080p	30 fps 4000 Kbps
CTS-1080P-Better	Cisco TelePresence 1080p	30 fps 3500 kb/s
CTS-1080P-Good	Cisco TelePresence 1080p	30 fps 3000 kb/s
CTS-720P-Best	Cisco TelePresence 720p	30 fps 2250 kb/s
CTS-720P-Better	Cisco TelePresence 720p	30 fps 1500 kb/s
CTS-720P-Good	Cisco TelePresence 720p	30 fps 1000 kb/s
CTS-720P-Lite	Cisco TelePresence 720p	30 fps 936 kb/s
custom	User-defined video traffic type	User-defined

Note More detailed configuration information on IP SLA Video Operation configuration is available in the *Configuring IP SLAs Video Operations* document referenced in the "Further Reading" section later in this chapter.

Media Awareness

Media awareness enables the network to be application and rich-media aware so that together with the video endpoints and applications it can provide an optimal QoE for end users and improved visibility for network operations staff.

Media awareness comprises four components:

- Flow metadata manages and distributes application attributes throughout the network, allowing appropriate policies to be applied and a self-learning aspect to the network traffic characteristics.
- Network Based Application Recognition 2 (NBAR2) provides enhancements to earlier NBAR capabilities integrating with flow metadata capabilities.
- Media Services Interface (MSI) is middleware included in Cisco video endpoints that announces the device's capabilities to the network.
- Media Services Proxy (MSP) produces flow metadata attributes that can be shared among network nodes on behalf of endpoints that do not make this information available themselves.

Flow Metadata

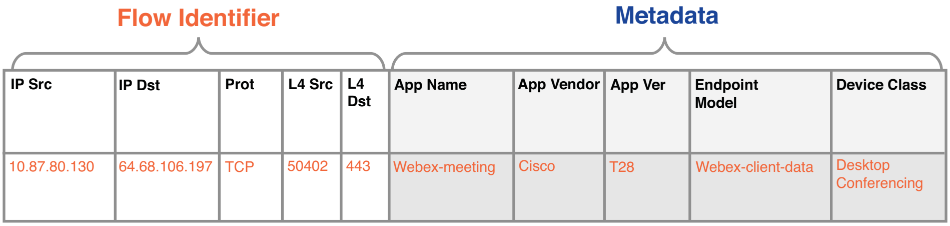
The flow metadata feature attempts to alleviate two significant difficulties with classifying rich-media application traffic. The first difficulty is that rich-media flows from an ever-growing variety of applications that use Hypertext Transfer Protocol (HTTP) as the transport layer (HTTP port 80 for unencrypted traffic, and HTTPS port 443 for encrypted traffic). This gave rise to the popular saying "HTTP is the new TCP," a phrase that alludes to the fact that application media types and flows can often no longer easily be derived and distinguished from the protocols and ports they use (the traditional five-tuple of flow identification).

Many rich-media applications (for example, WebEx) also have multiple sessions of differing traffic types open at the same time. The flow metadata feature introduces additional information (metadata) unique to an application flow that can be used for easier characterization and classification of the traffic streams.

The second difficulty in classifying rich-media traffic is that every node applying QoS policies has to identify and classify the flows and that the knowledge gained about the flow is available only to that node. With the flow metadata feature, certain attributes of an application traffic flow are noted (in <Attribute> <Value> pairs) and then collated in a control plane database and propagated to other network nodes, very loosely analogous to the concept of routing updates. In this way, what one node learns or knows about a flow can be leveraged by another node to apply QoS policies.

The flow metadata feature employs the concept of producers and consumers. Some network elements produce metadata information on application flows (usually the source or a node close to the source), while other nodes consume this information for the purposes of applying appropriate policies. Several Cisco endpoints (for example, the TelePresence endpoints) are instrumented to "announce" themselves into the network when they connect along with their metadata attributes; these types of applications and endpoints act as producers of metadata. The network nodes, such as the routers and switches in the infrastructure, are the consumers, and these nodes then match traffic based on these metadata attributes and impose the appropriate QoS policies.

The metadata on an application traffic flow provides much more in-depth information than the tradition five-tuple. The metadata is also independent of whether the payload stream is encrypted. The metadata attributes, as with the more traditional five-tuple, contain information that describes a flow but is not part of the flow itself. Figure 8-7 provides an example of the application information available with the flow metadata feature.


Flow Identifier
Metadata

IP Src	IP Dst	Prot	L4 Src	L4 Dst	App Name	App Vendor	App Ver	Endpoint Model	Device Class
10.87.80.130	64.68.106.197	TCP	50402	443	Webex-meeting	Cisco	T28	Webex-client-data	Desktop Conferencing

Figure 8-7 *Flow Metadata Contents*

The flow metadata information is propagated between nodes using RSVP PATH messages. As with the Mediatrace feature discussed earlier in this chapter, RSVP is used strictly as a transport protocol, and RSVP-specific configurations are not required to enable the feature. However, you should configure your firewalls to allow the RSVP messages to pass (RSVP is IP protocol 46); otherwise, metadata propagations will not traverse a firewall.

The flow metadata feature is enabled with the **metadata flow** and **metadata flow transmit** commands at the global configuration level. You can use the **show metadata application table** and **show metadata flow local-flow-id <n>** commands to print out the database of applications currently known to the node. More detailed configuration information on flow metadata configuration is available in the *Metadata Configuration Guide Metadata Command Reference* documents referenced in the "Further Reading" section later in this chapter.

Network Based Application Recognition 2

Traditional NBAR capabilities, triggered by the **match protocol** command, were discussed in Chapter 3 "Classification and Marking." NBAR2 extensions integrate the NBAR feature with the flow metadata feature by acting as a producer and consumer of metadata. These capabilities are triggered by the **match application** command.

The **match application** command allows classification of traffic streams based on their metadata, and includes the four options given in Example 8-4.

Example 8-4 *Flow Metadata Configuration Options*

```
! This configuration turns on IP SLA operation number 10
Router(config-cmap)# match application attribute ?
category     Category attribute to match
device-class Device Class attribute to match
media-type   Media type attribute to match
sub-category Sub Category attribute to match
```

Figure 8-8 shows a summary of the application classification possibilities of these four categories on the **match application** command.

The NBAR feature deployed by itself limits the visibility of its classification knowledge to the node where it is configured. The **match application** NBAR2 command, along with a **metadata flow transmit** command, allows NBAR2-produced application metadata to be propagated to neighboring nodes. Example 8-5 shows a configuration to achieve this.

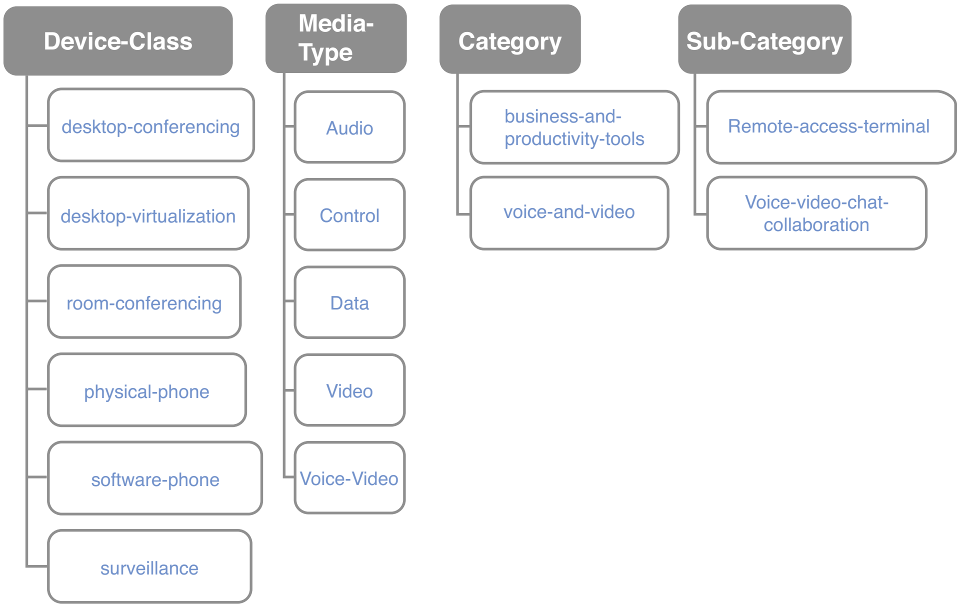


Figure 8-8 *NBAR2 Flow Metadata Configuration Options*

Example 8-5 *NBAR Integration with Metadata*

```
 ! These commands enable metadata to be collected on the node
 ! and transmitted to neighboring nodes
Router(config)# metadata flow
Router(config)# metadata flow transmit

 ! Define a class to classify/match WebEx application traffic
Router(config)# class-map CLASS-1
Router(config-cmap)# match application webex-meeting

 ! Define a policy for the class of traffic
Router(config)# policy-map POLICY-1
Router(config-pmap)# class CLASS-1
! <specify the QoS action for the specific application class>

! Attach the policy to the interface
Router(config)# interface gigabitethernet 0/0
Router(config-if)# service-policy output POLICY-1
```

Media Services Interface

The Media Services Interface (MSI) is a middleware software component embedded in video endpoints and collaboration applications. The MSI component of, for example the WebEx or Jabber clients, is responsible for the announcement of the application's flow metadata to the network and for other services allowing the endpoint to integrate seamlessly with the Mediatrace and Performance Monitor capabilities discussed earlier in this chapter. The application programming interface (API) is available for Windows and Linux operating systems and is downloadable from Cisco.com.

Media Services Proxy

The Media Services Proxy (MSP) is a capability on the network infrastructure to provide Medianet services for non-Cisco endpoints or Cisco endpoints without an embedded MSI component. The MSP resides on an access switch or router and learns information about devices and flows automatically.

The Cisco IOS device sensor feature gleans endpoint device information from protocols such as Cisco Discovery Protocol (CDP), Link Layer Discovery Protocol (LLDP), and Dynamic Host Control Protocol (DHCP). MSP leverages the device sensor framework to glean information from additional protocols such as Multicast Domain Name System (mDNS), H.323, and Session Initiation Protocol (SIP). This mechanism is illustrated in Figure 8-9.

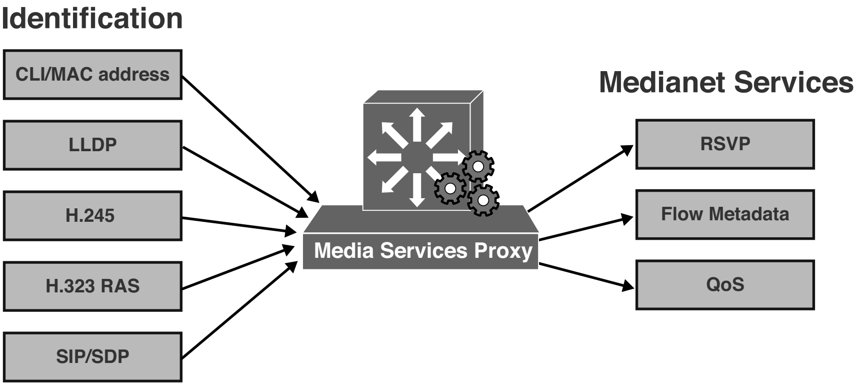


Figure 8-9 *Media Services Proxy as Metadata Producer*

Once information is available about devices connected to the network, this is shared in metadata databases so that Medianet services similar to endpoints with intrinsic Medianet capabilities can be provided to these third-party and non-Medianet-aware devices and endpoints.

Summary

Medianet is an end-to-end architectural network framework to deliver pervasive rich-media experiences with the best QoE. The architecture encompasses endpoints, network infrastructure, and cloud services. Medianet features and capabilities continue to be developed in Cisco IOS, and this chapter provided a brief overview of currently available capabilities, including the following:

- **Autoconfiguration:** The Auto Smartports switch feature automatically configures switch ports for the device connected to the port. The AutoQoS feature automatically configures detailed QoS features (such as classification, policing and queuing) on switches and routers.
- **Media monitoring:** The Mediatrace feature provides significantly enhanced troubleshooting capabilities by collating information about the path a media stream is taking through the network in one place. The Performance Monitor feature provides enhanced visibility into the network's performance with detailed statistics. The IPSLA VO feature allows for the simulation of a variety of video streams on the network to gauge performance.
- **Media awareness:** Provides the ability to differentiate applications and traffic flows based on unique attributes (metadata) that are not accessible with traditional five-tuple characterization. In addition, metadata is shared among network devices so that the network self-learns about traffic flows and can apply appropriate policies. The MSI provides middleware for endpoints to announce their metadata to the network, and the MSP provides proxy services for non-MSI-capable endpoints.

Further Reading

Overviews

Medianet Overview: http://www.cisco.com/go/medianet

Cisco Networking Capabilities for Medianet Datasheet: http://www.cisco.com/en/US/prod/collateral/routers/ps10536/data_sheet_c78-612429.pdf

Medianet QoS Design Strategy At-a-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qosmrn.pdf

AutoQoS for Medianet Campus Networks At-a-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/autoqosmediacampus.pdf

Medianet Architecture: http://cisco.com/web/solutions/trends/medianet/indepth.html

Medianet ASR 1000 QoS Design At-a-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoswanaggasraag.html

Medianet Reference Guide: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/Medianet_Ref_Gd/medianet_DG.pdf

Design Documents

Enterprise Medianet Quality of Service Design 4.0—Overview: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html

Medianet Campus QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html

Configuration Guides and Command References

Media Monitoring Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/media_monitoring/configuration/15-mt/mm-15-mt-book.html

Media Monitoring Command Reference: http://www.cisco.com/en/US/docs/ios/media_monitoring/command/reference/mm_book.html

Configuring IP SLAs Video Operations http://www.cisco.com/en/US/docs/ios-xml/ios/ipsla/configuration/12-2se/sla_video.html

IP SLAs Command Reference: http://www.cisco.com/en/US/docs/ios-xml/ios/ipsla/command/sla-cr-book.html

NBAR Configuration Guide, Cisco IOS Release 15M&T: http://cisco.com/en/US/docs/ios-xml/ios/qos_nbar/configuration/15-2mt/qos-nbar-15-2mt-book.html

Metadata Configuration Guide: http://cisco.com/en/US/docs/ios-xml/ios/mdata/configuration/15-mt/mdata-15-mt-book.html

Metadata Command Reference: http://cisco.com/en/US/docs/ios-xml/ios/mdata/command/mdata-cr-book.html

Media Services Proxy Configuration Guide: http://cisco.com/en/US/docs/ios-xml/ios/msp/configuration/15-mt/media-ser-prxy.html

Media Services Proxy Command Reference: http://cisco.com/en/US/docs/ios-xml/ios/msp/command/reference/guide/msp-cr-book.html

Resources and Services

Medianet Knowledge Base Portal: http://cisco.com/web/solutions/medianet/knowledgebase/index.html

Medianet Readiness Assessment Service: http://cisco.com/go/mra

Cisco Developer Network for Medianet: http://developer.cisco.com/web/mnets

Chapter 9

Application Visibility Control (AVC)

This chapter covers the following topics:

- Application visibility control (AVC) use cases
- How AVC works
- The AVC building blocks
- Performance considerations when using AVC

In the early years of IP networking, it was a fairly straightforward task to identify, classify, and control traffic based on the TCP or UDP port numbers. As applications have matured (particularly over the past decade), however, an increasing number of them have become difficult to identify in the traditional ways, such that standard classification based on the TCP or UDP port numbers is not always enough to implement a useful quality of service (QoS) strategy at the edges of the network.

For example, in the past, each application used a well-known TCP or UDP port numbers for almost everything. Email, web browsing, online video streaming, business intelligence tools, and so on all ran over well-known ports. Today, however, it is possible to run such things as WebEx, business intelligence tools, customer relationship management (CRM) applications, Outlook Web Access (OWA), instant messengers, and so on all over HTTP in a web browser. It has even been said that "HTTP has become the new TCP."

The challenge presented here is that traditional QoS classification tools typically rely on fields in the TCP/IP header to identify what kind of traffic they are dealing with. Today, it is not uncommon to see situations where most of the key applications (real time and otherwise) are all running through a web browser. In situations like this, a deeper understanding of the nature of the applications running within the HTTP session (or other upper-layer protocols) is the key to regaining both visibility and control of the network traffic. Although in a traditional sense HTTP and HTTPS are not really an "upper-layer" protocol, the reality of today's application demographics is that HTTP has really become a transport mechanism for so many new applications that you can almost consider it as an application layer within the application layer.

To illustrate the challenge this new paradigm brings, consider the following scenario. Your company may have strict policies on bandwidth-intensive applications such that any peer-to-peer (P2P), file-sharing, and video applications should be strictly limited to best effort within the network. Because these applications are embedded and possibly encrypted within HTTP sessions, however, there is no way to effectively enforce this QoS policy, rendering any existing QoS policies as almost valueless. What's more, you might not even know with any level of confidence how much these applications are even being used over your network, meaning that you cannot even analyze the situation to refine your QoS policy.

In the past, the focus was solely on protocols, but protocols are no longer the only important factor; the applications that run over the protocols are what need to be managed. These are precisely the kinds of problems that AVC was developed to solve. With a well-designed AVC deployment, you will be able to see many tangible benefits, including the following:

- Improved overall QoS and quality of experience (QoE) for all users through application-aware network optimization and control
- Proactive monitoring and end-to-end application visibility, which will assist with application performance troubleshooting
- Better prioritization of business-critical applications through the application of the QoS toolset
- Better control of HTTP embedded applications at the Internet edge, WAN edge, branch, and wireless aggregation points

AVC Use Cases

Considering that the main purpose of AVC is to offer greater visibility into the mechanics of the applications running on your network, which in turn gives you the ability to control these applications, several important use cases for AVC can be identified.

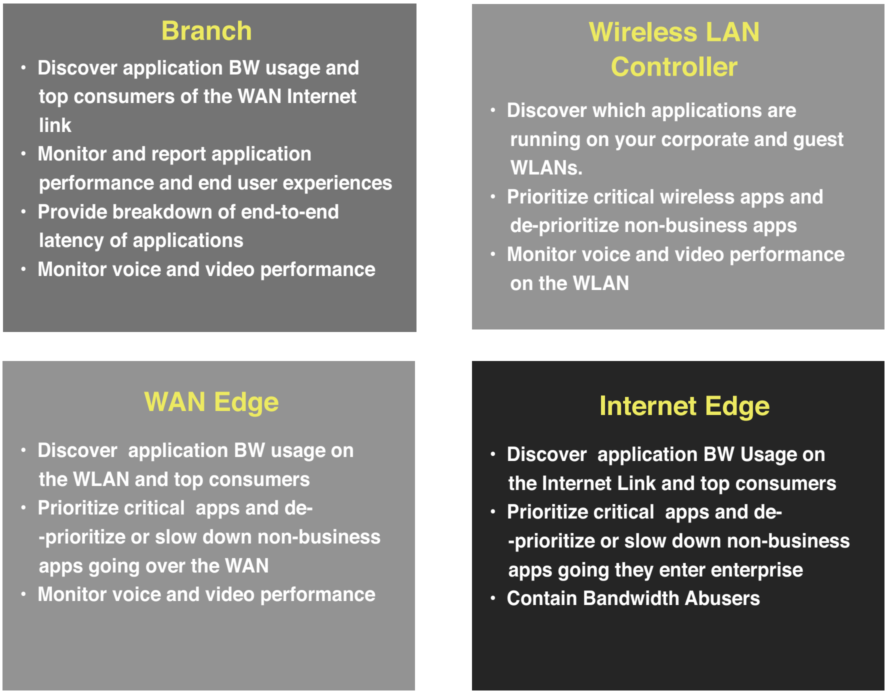
When AVC was first introduced by Cisco, the two most common use cases for AVC was to deploy it at the Internet edge and WAN aggregation points of the network on a Cisco ASR 1000 router. Enabling AVC at the Internet edge provides some clear advantages, as the bandwidth at the Internet edge point of presence (POP) is always limited to some specific amount, and there is a tangible benefit to both see and control the traffic coming and going from this point in the network. The same can be said of the WAN aggregation points, where an ASR 1000 might be used. If a WAN has hundreds or even thousands of spokes, the limited bandwidth available to each remote branch offices gives a very valid reason to want to understand the application behavior running across the WAN.

Over time, AVC has also been introduced into the Integrated Services Routers (ISR) G2 routers. With ISR G2 support, AVC is now being deployed at branch offices and other WAN and Internet edge POPs that do not have an ASR 1000.

Another place where AVC has been developed is in Cisco wireless LAN controllers (WLCs). In many ways, this is a natural place to deploy AVC because so much traffic aggregates into a single controller (or cluster of controllers). With AVC deployed on the WLC, it is now possible to identify and control application usage not just for the users on the corporate wireless network, but also for the users on the guest wireless network who are using the same network infrastructure as everybody else. Using AVC in 802.11 wireless networks is discussed in more depth in Part IV, "Wireless Campus QoS Design," of this book.

Note AVC is supported on the Cisco Aggregation Services Routers (ASR) 1000 starting with IOS XE Software Release 3.4S, on the Cisco ISR G2 routers starting with IOS Software Release 15.2(4)M2, and on the WLC starting with AireOS 7.4.

In summary, Figure 9-1 illustrates the four typical use cases where AVC is most often deployed in today's networks.



Branch

- Discover application BW usage and top consumers of the WAN Internet link
- Monitor and report application performance and end user experiences
- Provide breakdown of end-to-end latency of applications
- Monitor voice and video performance

Wireless LAN Controller

- Discover which applications are running on your corporate and guest WLANs.
- Prioritize critical wireless apps and de-prioritize non-business apps
- Monitor voice and video performance on the WLAN

WAN Edge

- Discover application BW usage on the WLAN and top consumers
- Prioritize critical apps and de--prioritize or slow down non-business apps going over the WAN
- Monitor voice and video performance

Internet Edge

- Discover application BW Usage on the Internet Link and top consumers
- Prioritize critical apps and de--prioritize or slow down non-business apps going they enter enterprise
- Contain Bandwidth Abusers

Figure 9-1 *Four Typical Use Cases of AVC*

How AVC Works

As the name indicates, the goal of AVC is twofold:

- Provide you with visibility into the applications that are transiting the network
- Provide a mechanism to control these applications

Because so many applications today are using generic protocols, such as HTTP, if you are to have any hope of controlling these applications, there needs to be a better approach than just identifying TCP or UDP port numbers. AVC is a solution that enables network devices, such as routers, to understand applications, not at the TCP/IP layers, but rather at the application layer itself using technologies such as deep packet inspection (DPI).

AVC is not actually a single technology. Rather, it is a solution that involves more of a multistep *design process* that links several technology elements together—and the process itself is called AVC. The AVC design process involves three important steps:

1. Discover applications with Network Based Application Recognition 2 (NBAR2).
2. Collect, aggregate, and export application information and its performance metrics with Flexible NetFlow (FNF) and analyze this data with a reporting tool.
3. Define QoS controls based on the information received from the first two steps.

Figure 9-2 illustrates the AVC design process.

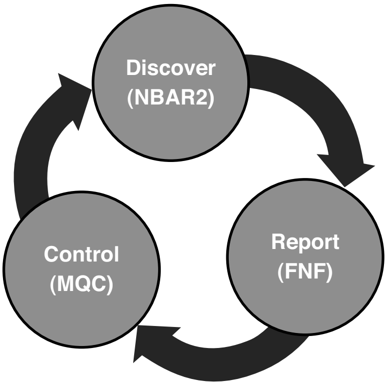


Figure 9-2 *The AVC Design Process*

Figure 9-2 shows that the design and management of QoS through AVC is more of a continuous, cyclical process that is always being refined. When dealing with the dynamic way applications are used and are continually changing in networks, this becomes a practical approach.

When you begin to deploy application layer QoS controls in your network, the task might seem almost impossible because you simply do not have a way to understand the application demographics of your network. Furthermore, you do not know how much data each application is consuming on WAN links, Internet edge POPs, or WLCs. To even begin the process of control, you first have to understand the behavior of the various applications that are being used on your network. For example, you might want to find out how much data is being used for social networking sites, how much bandwidth is being consumed by YouTube, or how many concurrent connections are P2P traffic.

NBAR2 is the AVC technology feature set that performs DPI on traffic flows thus helping to identify various application types. NBAR2 is currently supported on most ISR G2 devices, ASR 1000 series routers, and AireOS wireless LAN controllers. Also, NBAR2 is leveraged as a DPI engine in the Application Services Appliance (ASA) CX application firewall.

However, if the goal is to provide a robust application layer QoS policy, the ability to simply identify and classify a long list of applications is not enough. There can be hundreds or even thousands of different applications that are being used on any given network, and unless there is an easy way for you to interpret and digest this maze of information, it will provide very little benefit in ultimately applying the correct QoS controls in the network.

This is where FNF comes in. FNF is an enhancement to the classic NetFlow feature set that has been supported by Cisco for many years. Similar to earlier versions of NetFlow, FNF can characterize IP traffic and identify its source, traffic destination, and timing; however, FNF adds a critical new capability that empowers the AVC solution: FNF can also leverage the NBAR2 engine so that it can report on DPI application information. With the ability to export detailed application information supplied by NBAR2 and relevant performance metrics such as traffic volume and latency, you can now use network management tools that collect the FNF reports, which will give an accurate picture of the application performance and behavior at any point in your network where AVC is enabled.

Armed with this new information provided by the FNF reporting tool, it is now possible for you to deploy the right application-specific QoS tools at various points within the network. Figure 9-3 summarizes the four AVC building blocks that were just discussed.

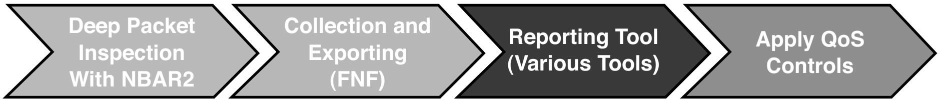

Deep Packet Inspection With NBAR2
Collection and Exporting (FNF)
Reporting Tool (Various Tools)
Apply QoS Controls

Figure 9-3 *The Four AVC Building Blocks*

Consider an example of how you might use this four-step AVC process in a real-world situation. Imagine that the managers in your company have raised concerns about application misuse within the network. They suspect that a large number of employees are using and spending a lot of time on applications such as BitTorrent, Facebook, and YouTube. Not only does this waste the company's time, but it also consumes a significant amount of the available Internet and WAN bandwidth. To address this question and to determine whether this concern is valid, you turn on NBAR2 on both the WAN aggregation and

Internet edge routers and export the data to a reporting tool via FNF. You might discover that YouTube in particular is consuming a huge amount of bandwidth and beginning to starve other applications that also use port 80. To address this situation, you might enable QoS control mechanisms at various spots in the network (that are NBAR2 capable) to mark down and throttle the impact that YouTube is having on your network.

This simple example demonstrates how the four-step process must be followed to correctly respond to the application behavior that is being observed in the network. This chapter will discuss more in-depth examples of how to specifically use AVC in various points within your network.

To summarize, the four key building blocks of the AVC solution are as follows:

- **Building block 1:** NBAR2
- **Building block 2:** FNF
- **Building block 3:** AVC reporting tools
- **Building block 4:** Apply the QoS controls

The following sections examine these building blocks in greater detail.

The AVC Building Blocks

As explained in the previous section, deploying AVC involves four building blocks. These are discussed in the subsections that follow.

Building Block 1: NBAR2

The classic NBAR feature set has been supported in Cisco IOS platforms (such as the ISR family) for many years. As discussed in Chapter 3, “Classification and Marking,” NBAR enables the router to examine packets at the application layer. Classic NBAR had a fairly limited list of application signatures and was thus limited in its DPI capabilities.

NBAR2 (also known as next-generation NBAR) is a complete redesign of the NBAR feature set, based on another Cisco product: the Services Control Engine (SCE). NBAR2 includes advanced classification techniques with many more signatures than were available with classic NBAR. In addition, NBAR2 supports a new sophisticated signature fidelity capability, which greatly helps to improve overall accuracy of the signature matches.

In addition, NBAR2 has been adopted as a Cisco cross-platform protocol classification mechanism, meaning the same NBAR2 features are supported across a wide variety of Cisco platforms. NBAR2 supports more than 1500 applications and subclassifications, and Cisco has been adding more than hundred new signatures per year to the protocol pack.

One key feature of NBAR2 is that signatures are delivered through a protocol pack. A protocol pack is a set of protocols developed and packaged together in a single file. Protocol packs allow the distribution of application and protocol signature updates to the IOS router without forcing an upgrade and a reload of the Cisco IOS Software. Although new protocol packs may be downloaded and used on a router, at any point in time there can be only a single "active" protocol pack running on the device. If a new protocol pack is downloaded, it replaces the current protocol pack. Each IOS image also has a built-in default protocol pack. If the active protocol pack is unloaded, the IOS image returns to its in-built default protocol pack.

Note With traditional NBAR, protocol application modules were called Packet Description Language Modules (PDLMs). In contrast to a PDLM, a protocol pack used in NBAR2 is a single compressed file that contains multiple PDL files. Protocol packs allow users to load a set of protocols together rather than load them separately.

Within the protocol pack, NBAR2 DPI organizes the signatures into major application categories, and then into subcategories. For example, some of the major categories are browsing, business and productivity tools, email, file sharing, and so on. Subcategories offer an even more granular organization of similar applications. For example, YouTube is found in the major category of Voice and Video under the subcategory of Streaming. In addition, each application is given an attribute to identify whether this application is P2P, tunneled, or encrypted.

One example is WebEx and YouTube, which both share the same category of Voice and Video. However, the subcategories differ. YouTube has the subcategory of Streaming, and WebEx has the subcategory of Voice-Video-Chat Collaboration. Also, by default WebEx has the attribute of Encrypted set to Yes, whereas YouTube does not.

To summarize, NBAR2 organizes the application inspection in the following ways:

- **Application group:** Grouping of applications that are part of the same application suite or "brand." For example, Yahoo Messenger, Yahoo VoIP Messenger, and Yahoo VoIP-over SIP are grouped together under Yahoo Messenger Group.
- **Category:** Grouping of applications that support similar functionality from an end-user standpoint. Some examples include email, gaming, newsgroup, and so on.
- **Subcategory:** This provides a secondary grouping of applications with similar functionality from an infrastructure/networking standpoint. Examples include routing protocol, database, and streaming.
- **P2P technology:** An attribute is assigned indicating whether this application uses a P2P technology.
- **Tunnel:** NBAR2 assigns a Boolean attribute indicating if an application tunnels other protocols.

- **Encrypted:** NBAR2 assigns a Boolean attribute indicating if an application is encrypted.

NBAR has two key modes of operation, as discussed in the sections that follow:

- NBAR protocol discovery
- MQC traffic classification

Note You will notice that the terms *NBAR* and *NBAR2* are used somewhat interchangeably throughout this chapter. Because NBAR2 builds on the foundation of the original NBAR feature set, much of the syntax, command structure, and function is similar. Therefore, although both of these acronyms are used throughout this chapter, they basically refer to the same thing; but NBAR2 is the focus.

NBAR2 Protocol Discovery

The previous section described how NBAR2 is organized. Now we will examine how NBAR2 protocol discovery can be enabled and then used to classify applications. To simplify matters, NBAR2 leverages the same configuration steps as classic NBAR. There are two steps when using NBAR2:

1. NBAR protocol discovery must be enabled on a specific interface to gain an understanding of the applications in the network
2. Modular QoS command-line interface (MQC) is used for protocol classification. MQC is actually independent of protocol discovery, but it uses the NBAR2 DPI engine to first classify and then control applications.

NBAR protocol discovery enables you to identify and get real-time statistics of the applications that are currently running in your network. You can then use the information obtained from protocol discovery to create QoS classes and policies.

Note NBAR protocol discovery is the focus here, but another method that uses the NBAR DPI engine, FNF, is discussed later in the chapter.

As with classic NBAR, protocol discovery should be configured only on the specific interfaces that you want to do discovery on. When enabled, it provides real-time statistics on applications currently running through that interface. NBAR provides additional information such as bidirectional bit rate, packet count, and byte count statistics.

Example 9-1 demonstrates how to configure NBAR protocol discovery.

Example 9-1 *Enabling NBAR Protocol Discovery*

```
ASR1002(config)# interface GigabitEthernet0/0/3
ASR1002(config-if)# ip nbar protocol-discovery
```

A key aspect of NBAR2 is that it also supports IPv6 traffic flows. Example 9-2 illustrates how to enable NBAR protocol discovery for IPv6.

Example 9-2 *Enabling NBAR Protocol Discovery for IPv6*

```
ASR1002(config)# interface GigabitEthernet0/0/3
ASR1002(config-if)# ip nbar protocol-discovery ipv6
```

Once protocol discovery has been enabled, it is then possible to see what applications have been discovered on each interface. Example 9-3 shows output of a Top-N report, showing the top five applications that were discovered on interface GigabitEthernet0/0/3.

Example 9-3 *Displaying the Top 5 NBAR Application Activity Results*

```
ASR1002# show ip nbar protocol-discover top-n 5
 GigabitEthernet0/0/3
                   Input                    Output
                   -----                    ------
   Protocol        Packet Count             Packet Count
                   Byte Count               Byte Count
                   30sec Bit Rate (bps)     30sec Bit Rate (bps)
                   30sec Max Bit Rate (bps) 30sec Max Bit Rate
   --------------- ------------------------ ---------------------
   sunrpc          135                      2799106
                   21850                    4237830347
                   0                        0
                   4000                     47720000
   ftp             92340159                 186761181
                   4989796991               263959173216
                   211000                   11160000
                   284000                   15170000
   netflix         67438860                 84034631
                   51098349136              100709287113
                   2171000                  4223000
                   3278000                  6476000
   webex-meeting   45807530                 163458047
                   2497543722               129842885217
                   115000                   5998000
                   152000                   7799000
   bittorrent       59667396                156155174
                   12768822744             103187176646
                   555000                  4715000
                   697000                  5077000
```

The Top-N report can be very useful in helping to understand the types of protocols and applications that are running over any interface where NBAR protocol discovery has been enabled. This information is also conveniently available in the SNMP Management Information Base (MIB) CISCO-CLASS-BASED-QOS-MIB.

As you can see from Example 9-3, BitTorrent and Netflix (which are likely not serving any useful purpose to the business) are consuming a fairly massive amount of bandwidth and would be excellent candidates to be throttled down. Note, as well, that the expression *protocol discovery* is something of a misnomer: NBAR2 is doing much more than just discovering generic protocols; it is discovering the applications that are running over the upper-layer protocols.

Note In some cases, the output of the command **show ip nbar protocol-discovery** will list the protocol as Unknown. Three are three possible explanations for this:

- It is possibly because there is no available NBAR2 signature for this application. In this case, a custom signature can be created.
- The traffic flow may be asymmetrical. NBAR2 relies on symmetrical traffic flows so that it can see the full nature of the application.
- The system memory allocated to NBAR2 may have been exceeded (as discussed in more detail later in the "Performance Considerations When Using AVC" section).

NBAR2 MQC Traffic Classification

Now that NBAR protocol discovery has been reviewed, the following section examines NBAR traffic classification using MQC. Although NBAR protocol discovery is a mechanism for the router to examine and truly "discover" applications and protocols, on its own, it leaves no way to actually control the behavior of these applications.

In general, NBAR classification follows the same general MQC QoS classification syntax used for other protocols. Example 9-4 shows a generic class map that is used for NBAR application classification.

Example 9-4 *Constructing an NBAR Class Map*

```
ISRG2(config)# class-map [match-any|match all] <CLASS-MAP-NAME>
ISRG2(config-cmap)# match protocol <application|attribute>
```

Constructing a class map in this way allows you to match either on a specific application protocol (such as HTTP) or on an entire application protocol group. For example, suppose that you want to build a class map to match the larger application group of file-sharing. In this case, you are not matching a specific well-known application, but rather a larger group of applications. In this case, you must use the **attribute** keyword to match on either a major category or a smaller subcategory, as demonstrated in Example 9-5.

Example 9-5 *An NBAR2 Class Map That Matches File-Sharing*

```
ISRG2(config)# class-map MY-CLASS
ISRG2(config-cmap)# match protocol attribute category file-sharing
```

In this case, the class map matches on the major category attribute of **file-sharing**; however, in most cases, this will be much too broad, and a more specific match will be desired. For example, take a look at the attributes of a specific protocol (in this case, the **webex-meeting** protocol), as demonstrated in Example 9-6.

Example 9-6 *Examining the NBAR2 Protocol Attribute Details*

```
ISRG2# show ip nbar protocol-attribute webex-meeting
           Protocol Name :  webex-meeting
                category :  voice-and-video
            sub-category :  voice-video-chat-collaboration
       application-group :  webex-group
          p2p-technology :  p2p-tech-no
                  tunnel :  tunnel-no
               encrypted :  encrypted-yes
```

In this example, **webex-meeting** is the general protocol attribute, but this protocol is also part of the major category **voice-and-video**, in the subcategory of **voice-video-chat-collaboration**, and in the application group of **webex-group**. In addition, the other attributes are set to **no**, except for the **encrypted** attribute. All these fields are relevant when creating a specific class map.

NBAR2 also enables you to see what the subcategories are within a major category group and the individual applications within a subcategory. For example, to create a class map, you may want to better understand what applications are supported under the **voice-video-chat-collaboration** subcategory. The output in Example 9-7 shows various applications in this subcategory.

Example 9-7 *Examining the NBAR Applications in a Subcategory*

```
ISRG2# show ip nbar attribute sub-category voice-video-chat-collaboration
  aol-messenger          AOL Messenger Text Chat
```

```
 aol-protocol           America OnLine protocol
 bnet                   bnet
 cisco-phone            Cisco IP Phones and PC-based UC
 conference             chat
 cooltalk               Internet telephony tool
 cuseeme                CU-SeeMe desktop video conference
 directplay             DirectPlay
 fring-voip             Fring Voip
```

You can find all the subclassifications supported by NBAR2 by issuing an IOS command. Example 9-8 shows the output from ASR running IOS XE 3.7S.

Example 9-8 *Looking at the NBAR2 Parameter Subclassifications*

```
ASR1000# show ip nbar parameter subclassification
 Protocol        Parameter            Parameter type
 --------        ---------            --------------
 share-point     Blog                 enum
 share-point     Document             enum
 share-point     Admin                enum
 share-point     Calendar             enum
 vnc             file-transfer        enum
 edonkey         search-file-name     regexp
 edonkey         file-transfer        regexp
 edonkey         text-chat            regexp
 rtp             payload-type         multi
 rtp             video                enum
 rtp             audio                enum
 webex-meeting   payload-type         multi
 webex-meeting   video                enum
 webex-meeting   audio                enum
 kazaa2          file-transfer        regexp
 gnutella        file-transfer        regexp
 fasttrack       file-transfer        regexp
 citrix          app                  regexp
 citrix          ica-tag              integer
 http            mime                 regexp_url
 http            content-encoding     regexp_url
 http            location             regexp_url
 http            server               regexp_url
 http            from                 regexp_url
 http            referer              regexp_url
 http            user-agent           regexp_url
 http            host                 regexp_url
 http            url                  regexp_url_nl
```

Putting this together, consider Example 9-9, where a router running AVC has three NBAR2 class maps:

- One for SAP traffic
- One for general Internet browsing
- One that specifically matches on the HTTP URI (so-called subclassification) cisco.com.

Example 9-9 *Creating Class-Maps for Various Application Groups*

```
ISRG2(config)# class-map match-all SAP
ISRG2(config-cmap)# match protocol sap
 ! Class map to match only the "sap" protocol
ISRG2(config)# class-map match-any BROWNSING
ISRG2(config-cmap)# match protocol attribute category browsing
 ! Class map to match the entire category of "browsing"
ISRG2(config)# class-map match-any INTERNAL-DATA
ISRG2(config-cmap)# match protocol http url "*cisco.com"
 ! Creates a class-map to match on "cisco.com"
```

You can verify this configuration with the **show class-map** command:

The second major component of AVC is the export function of FNF, as discussed in the following section.

Building Block 2: Flexible NetFlow

The second key building block of the AVC solution is Flexible NetFlow (FNF). The previous section demonstrated how to enable NBAR2 protocol discovery. However, this approach is cumbersome to manage and provides an inelegant way to analyze application performance data. An alternative to NBAR2 protocol discovery is to use FNF to collect DPI NBAR2 information along with traffic statistics and export data to a reporting tool via either FNF NetFlow Version 9 or IP Flow Information Export (IPFIX).

Although NetFlow Version 9 is not on the IETF standard track (it is in the *informational* category), it is an open standard, meaning that Cisco has published the specifications of this tool, and there are many reporting partners who support it. Cisco has since worked with the IETF to standardize a flow export format known as IPFIX, which is on the IETF standards track. Unlike traditional NetFlow, which is restricted to using fixed fields, FNF collection and export uses flexible fields—meaning that only the relevant data corresponding to those fields is collected and reported on. When the router exports data to the NetFlow collector, the format of the flow record is also communicated up to collector so that the reporting tool understands how data is being sent.

FNF relies on NetFlow Version 9 and IPFIX to export flow record information. As Figure 9-4 illustrates, FNF expands the capabilities of traditional NetFlow to include a much broader array of information, spanning the L2–L7 protocol stack. It is the capability of NetFlow Version 9 to export detailed application statistics that enables AVC and facilitates the ultimate QoS controls that are applied.

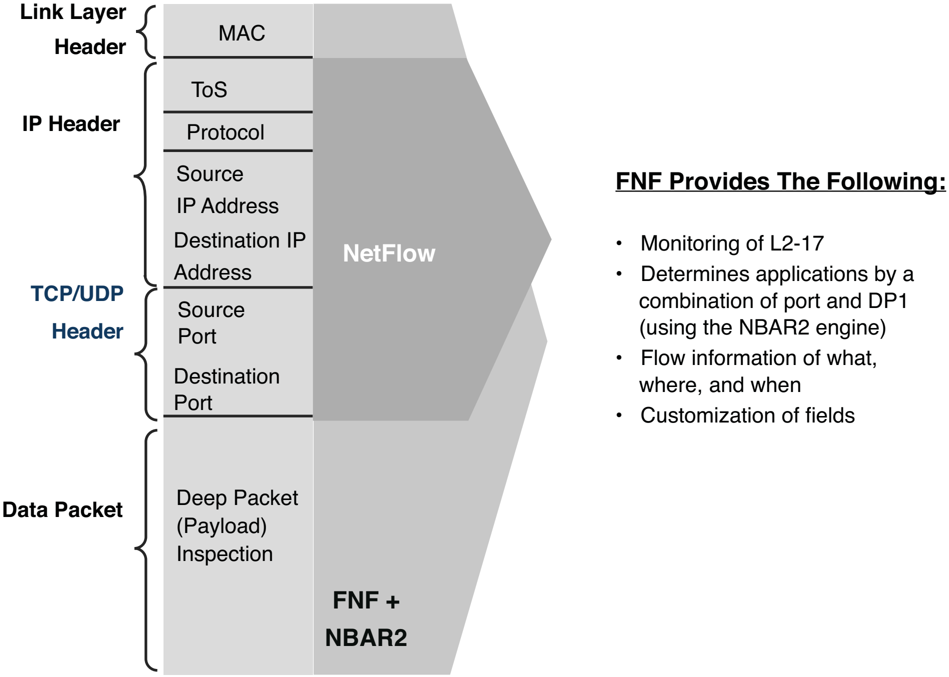


Figure 9-4 *FNF Coverage of the Protocol Stack*

Flexible NetFlow Key Fields and Non-Key Fields

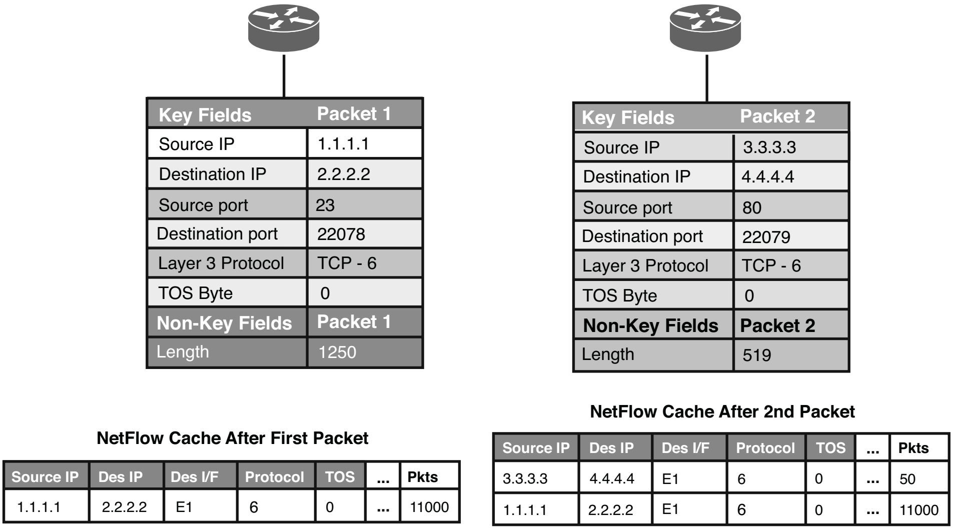
Before taking a closer look at how to configure FNF on a router, it is important to understand an important concept that is central to FNF: key and non-key fields.

Each packet that is forwarded by an interface where FNF is enabled is examined for a set of basic IP packet attributes that uniquely identify the flow. In FNF terminology, these are called *key fields*. You can think of key fields as an index to a database. These fields need to be unique for each flow record. Non-key field, by way of contrast, are attributes or characteristics of an existing flow.

For example, if a router were to examine some incoming packets and, after checking the flow cache, it recognizes that there is no existing flow record for these key fields, a new flow record is created. As this session continues to flow through the router, FNF examines the behavior of the flow (such as byte and packet count) and ultimately exports this information to the collector; however, this information is not used when creating the flow.

In this case, the source and destination IP addresses, source and destination ports, protocol type, and so on all identify the flow and are thus key fields. In contrast, the packet and byte count metrics are characteristics of the flow and are thus non-key fields.

Figure 9-5 illustrates how the key fields uniquely identify each flow record in the flow cache.



Key Fields	Packet 1
Source IP	1.1.1.1
Destination IP	2.2.2.2
Source port	23
Destination port	22078
Layer 3 Protocol	TCP - 6
TOS Byte	0
Non-Key Fields	**Packet 1**
Length	1250

Key Fields	Packet 2
Source IP	3.3.3.3
Destination IP	4.4.4.4
Source port	80
Destination port	22079
Layer 3 Protocol	TCP - 6
TOS Byte	0
Non-Key Fields	**Packet 2**
Length	519

NetFlow Cache After First Packet

Source IP	Des IP	Des I/F	Protocol	TOS	...	Pkts
1.1.1.1	2.2.2.2	E1	6	0	...	11000

NetFlow Cache After 2nd Packet

Source IP	Des IP	Des I/F	Protocol	TOS	...	Pkts
3.3.3.3	4.4.4.4	E1	6	0	...	50
1.1.1.1	2.2.2.2	E1	6	0	...	11000

Figure 9-5 *FNF Key Fields and Non-Key Fields After Two Packets*

Configuration of FNF

The steps to configure FNF on a router are as follows:

1. Configure the flow exporter.
2. Configure the flow record.
3. Configure the flow monitor.
4. Enable FNF on the relevant interface.

Step 1: Configure the Flow Exporter

The flow exporter identifies the destination where the FNF flow records are sent to by the router. FNF flow records are usually sent to some kind of a NetFlow collector or FNF management utility, such as Insight, Cisco Prime Infrastructure, or another network management tool. If necessary, it is possible for each flow to be exported to multiple collectors; however, in Example 9-10, FNF flows are exported to a single collector with the IP address of 172.16.1.1. Furthermore, if necessary, it is also possible to specify the UDP port number of the FNF collector.

Example 9-10 *Defining the FNF Flow Exporter*

```
ISRG2(config)# flow exporter MY-EXPORTER
ISRG2(config-flow-exporter)# destination 172.16.1.1
ISRG2(config-flow-exporter)# transport udp 2055
 ! Pay attention to the UDP port number, as this may be different
 ! depending on what netflow collector is used
```

You can verify this configuration with the **show flow exporter** *name* command:

Step 2: Configure the Flow Record

The second step is to configure FNF flow records. The flow record tells the router what to measure. For example, these might be byte and packet count, application name, and so on. Example 9-11 highlights how to do this for a traffic stream that is monitored on a WAN interface.

Notice from Example 9-11 that it is necessary to configure two flow records, one for the input (ingress) direction and one for the output (egress) direction. If FNF were to be used without NBAR2 (meaning application recognition is not the goal), the statement **match application name** can be skipped. NBAR2, once activated either through protocol discovery or through FNF **match application**, is bidirectional. That is, it will automatically inspect traffic in both directions on an interface regardless of whether FNF is applied in both directions or not. However, when you apply FNF in both in/out directions, it allows FNF to collect both egress and ingress traffic, which provides a much better view of the application behavior on the network. The following examples demonstrate FNF configured in both directions.

Example 9-11 *Configuring the FNF Flow Records*

```
ISRG2(config)# flow record INPUT-APP-RECORD
 ! Use whatever name suits best - "INPUT-APP-RECORD" is used here
ISRG2(config-flow-record)# match interface input
ISRG2(config-flow-record)# match flow direction
 ! These two commands identify this as an "input flow"
ISRG2(config-flow-record)# match application name [account-on-resolution]
 ! This command uses NBAR2 to classify the application
ISRG2(config-flow-record)# collect counter packets
ISRG2(config-flow-record)# collect counter bytes
<snip>
ISRG2(config)# flow record OUTPUT-APP-RECORD
 ! The output (egress) flow record
ISRG2(config-flow-record)# match interface output
ISRG2(config-flow-record)# match flow direction
 ! These two commands identify this as an "output flow"
```

```
ISRG2(config-flow-record)# match application name [account-on-resolution]
 ! This command uses NBAR2 to classify the application
ISRG2(config-flow-record)# collect counter packets
ISRG2(config-flow-record)# collect counter bytes
```

In Example 9-11, two different keywords are being used in the flow record. The **match** keyword refers to a field that is a key field and, as discussed earlier, is used to characterize and create flows. The **collect** keyword is used to denote non-key fields and is used for information that is to be added to the flow, such as byte and packet count metrics.

In Example 9-11, the **match application name** command is what enables NBAR2 to identify and classify the application. In some cases, however, NBAR2 might not be able to correctly classify the application until it has seen enough packets. In this case, you should use the **account-on-resolution** optional keyword, which allows the router to cache the counters and timers in the flow table until NBAR2 correctly classifies the application. For example, if a user were opening her web browser to download a web page, the first packets are the TCP handshake (SYN -> SYNACK -> ACK), followed next by an HTTP GET. Only at this point, after seeing the HTTP GET and the HTTP 200 response message from the web server, will NBAR2 be able to recognize this as an HTTP connection.

Until the application is correctly classified, the Application Name field is set to Unknown. When the application has been correctly classified, the router will start accounting in the FNF cache and will then add the counters previously cached in the flow table.

You can confirm the flow record configuration with the **show flow record** command.

Step 3: Configure the Flow Monitor

The third step is to configure the flow monitor. The flow monitor is where the flow record that was previously configured is connected with the flow exporter that was configured in Step 1. This approach allows great flexibility in customizing different flow records and how they can be exported. For example, by creating different flow monitors, it is possible to have different flow records that examine different fields and then have them sent to different collectors.

Another key point in configuring the flow monitor is that you need to set up a flow record for each direction that traffic is flowing. Example 9-12 shows how to configure FNF flow monitors. Note how two flow monitors have been configured here (in the same way two flow records were previously configured)—one for the input and one for the output directions.

Example 9-12 *Configuring the Input and Output FNF Flow Monitors*

```
ISRG2(config)# flow monitor MY-FLOW-MONITOR-IN
 ! Create the input flow monitor
ISRG2(config-flow-monitor)# record INPUT-APP-RECORD
```

```
 ! References the previously configured input flow record
ISRG2(config-flow-monitor)# exporter MY-EXPORTER
 ! Refers to the previously configured exporter "MY-EXPORTER"
<snip>
ISRG2(config)# flow monitor MY-FLOW-MONITOR-OUT
 ! Create the output flow monitor
ISRG2(config-flow-monitor)# record OUTPUT-APP-RECORD
 ! References the previously configured output flow record
ISRG2(config-flow-monitor)# exporter MY-EXPORTER
 ! Refers to the previously configured exporter "MY-EXPORTER"
```

You can confirm the flow monitor configuration with the **show flow monitor** command.

Step 4: Enable FNF on the Relevant Interface

The final step in configuring FNF is to attach the flow monitor to an interface, as demonstrated in Example 9-13. On a WAN router, this is usually the interface facing into the WAN, and on the Internet edge, it would be the interface looking toward the Internet. One key aspect to be aware of here is that the flow monitor is configured for a particular direction; so the two flow monitors configured in Step 3 must both be added, one for each direction.

Example 9-13 *Enabling the Flow Monitors on the Interface*

```
ISRG2(config)# interface GigabitEthernet3/0/0
ISRG2(config-if)# ip flow monitor MY-FLOW-MONITOR-IN input
 ! Attach the flow monitor in the ingress direction
ISRG2(config-if)# ip flow monitor MY-FLOW-MONITOR-OUT output
 ! Attach the flow monitor in the egress direction
```

You can confirm this configuration with the **show flow monitor** command.

Building Block 3: AVC Management and Reporting

The third building block of the AVC solution is the management and reporting tool, which supports FNF collector functionality. It is also the management and reporting tool that gives you the clearest idea of traffic and application patterns that are flowing through your network. There is no one single management and reporting tool that Cisco recommends using. In fact, at the time of this writing, Cisco supports several management tools, including Insight Reporter, Cisco Prime Network Analysis Module (NAM), and Cisco Prime Infrastructure. It is important to note that AVC uses open-standard flow export mechanisms, such as NetFlow Version 9 and IPFIX. Therefore, you should work within the Cisco reporting ecosystem to enable rich monitoring capabilities that fit your needs.

It is not the purpose of this book to provide a complete overview of each management tool; even so, a quick discussion of Insight Reporter is presented here.

Insight Reporter

Insight Reporter is a sophisticated management tool that can collect NetFlow 9 records and display them in a useful manner that helps administrators better understand what kinds of application layer QoS tools need to be implemented in the network. To summarize, Cisco Insight provides the following key features:

- Application visibility per interface:
  - Displays data usage (bandwidth, volume of traffic, duration, session, and so on)
  - Shows application popularity (top applications)
  - Monitors application transactions (Top-N reports, application trends)
- Data grouping and filtering:
  - Displays application groups and IDs (YouTube, BitTorrent, and so on)
  - Sorts applications into categories and subcategories
  - Which protocols are encrypted or tunneled
- Network demographics and online monitoring:
  - Top users in the network
  - Usage trends
  - Tracking of user online activity

Figure 9-6 shows a report from the Cisco Insight management tool where various protocol and application activity, as reported by FNF, is displayed.

Ultimately, tools like Cisco Insight and others help to provide you with a clearer picture of the application activity within the network. This information thus helps in the identification of where application layer QoS policies need to be enforced and how to structure these policies. This is discussed in the final AVC building block: AVC QoS controls.

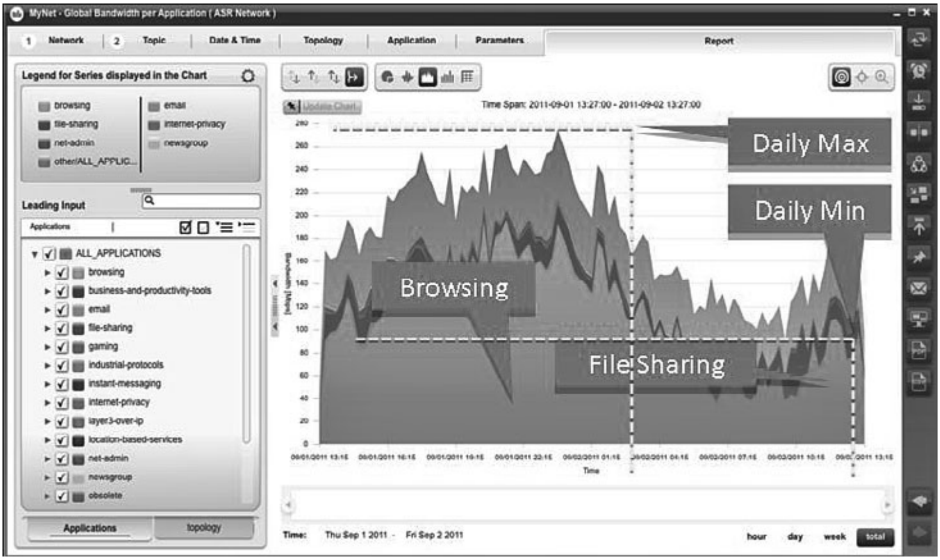


Figure 9-6 *Cisco Insight Management Tool Sample Report*

Building Block 4: AVC QoS Controls

The first three building blocks help you to understand the application dynamics inside the network. However, just knowing which applications are in use, and how much bandwidth they use, is not enough. Application performance is how users perceive the network performance. For example, you might want to offer guaranteed bandwidth to protect critical web-based customer relationship management (CRM) applications from network congestion, or perhaps provide low-latency handling to delay-sensitive business intelligence (BI) applications, or even strictly limit unwanted P2P applications from using the Internet edge resources.

The following two examples examine how to deploy AVC at the WAN edge (the first example) and at the Internet edge (the second example).

Deploying AVC QoS Controls at the WAN Edge

Consider an example where you want to implement a QoS policy at the WAN edge on an ASR 1000 router that identifies HTTP traffic and is able to classify the different applications within. Table 9-1 summarizes the proposed QoS handling at the ASR 1000 WAN edge router.

Table 9-1 *Example of AVC QoS Policies to Be Deployed on an ASR 1000 Internet Edge Router*

Application	Bandwidth	Priority
Citrix	Committed to 50%	High

Application	Bandwidth	Priority
Web browsing	30% of the remaining bandwidth = 15% of line bandwidth	Normal
Internal web browsing (web-based corporate tools)	60% of total web browsing bandwidth	Normal
Remaining	20%	Normal

Configuration of these QoS policies involves the following three steps:

1. Configure the NBAR2 application class maps (see Example 9-14).
2. Configure the appropriate policy maps (see Example 9-15).
3. Attach the policy map to the correct interface (see Example 9-16).

To begin, create class maps for each of the applications listed in Table 9-1. When matching a particular application where DPI is required, the NBAR2 engine is automatically utilized.

Example 9-14 *Step 1: Configure Application Class Maps*

```
ASR1000(config)# class-map match-all CITRIX
ASR1000(config-cmap)# match protocol citrix
 ! Matches citrix traffic
ASR1000(config)# class-map match-all BROWSING
ASR1000(config-cmap)# match protocol attribute category browsing
 ! Matches general HTTP browsing
ASR1000(config)# class-map match-any INTERNAL-BROWSING
ASR1000(config-cmap)# match protocol http url "*internalweb.com*"
 ! Matches any internal browsing traffic going to internalweb.com
```

You can verify this configuration with the **show class-map** command.

The second step is to configure the appropriate QoS policy maps. Example 9-15 illustrates how this may be done. Note in this example that the **BROWSING** parent class calls the child policy called **INTERNAL-BROWSING-POLICY**. This approach of using a parent and child policy allows internal browsing to use a specific percentage of the entire browsing bandwidth.

Example 9-15 *Step 2: Configure Policy Maps*

```
ASR1000(config)# policy-map INTERNAL-BROWSING-POLICY
ASR1000(config-pmap)# class INTERNAL-BROWSING
ASR1000(config-pmap-c)# bandwidth remaining percent 60
```

```
 ! This is the child policy of the "browsing" parent policy

ASR1000(config-pmap)# policy-map WAN-NETWORK-POLICY
ASR1000(config-pmap-c)# class CITRIX
ASR1000(config-pmap-c)# police percent 50
 ! Allocates 50 percent of bandwidth to Citrix
ASR1000(config-pmap)# class BROWSING
ASR1000(config-pmap-c)# bandwidth remaining percent 30
ASR1000(config-pmap-c)# service-policy INTERNAL-BROWSING-POLICY
 ! The parent policy calls the child policy to protect internal browsing
 ! when congestion occurs where more than 30% of the bandwidth
 ! is needed for browsing.
```

You can verify this configuration with the **show policy-map** command.

The third step is to apply the policy map to the interface, as demonstrated in Example 9-16.

Example 9-16 *Step 3: Attach the Policy Map to the Interface*

```
ASR1000(config)# interface GigabitEthernet0/0/0
ASR1000(config-if)# service-policy output WAN-NETWORK-POLICY
```

You can verify this configuration with the **show policy-map interface GigabitEthernet0/0/0** command.

The following section examines another example of deploying AVC, this time at the Internet edge.

Deploying AVC QoS Controls at the Internet Edge

Now consider an example where AVC QoS controls are deployed at the Internet edge of the network. In this case, the ASR 1000 is the Internet edge router. The company is looking to begin using a new Internet connection connected to a local ISP that supports 50 Mbps both inbound and outbound. The ASR physical interface is Gigabit Ethernet, so there will be a requirement for a multilevel hierarchical shaper that applies backpressure if the total traffic volume exceeds 50 Mbps.

The company is primarily using web-based applications across the Internet (such as web-based file-sharing, CRM, and BI applications). Considering that many of the employees are also using P2P applications, such file sharing, the company has decided to implement a QoS policy where P2P will be limited to 20 percent of the available bandwidth, and the critical applications that use web browsing will be given the remaining 80 percent of the available bandwidth.

To summarize, the QoS policy on the Internet edge router is as follows:

- The Internet connection is to be shaped down to 50 Mbps.
- Browsing is considered as critical traffic and will be allocated 80 percent of available bandwidth.
- P2P applications, such as file sharing, will be limited to 20 percent of available bandwidth.

Following the same three configuration steps shown in the preceding "Deploying AVC QoS Controls at the WAN Edge" section, the QoS policy may be configured as demonstrated in Example 9-17.

Example 9-17 *Configuring a QoS Policy at the Internet Edge*

```
ASR1000(config)# class-map match-all BROWSING
ASR1000(config-cmap)# match protocol attribute category browsing
 ! This class map matches all browsing traffic
ASR1000(config-cmap)# class-map match-all P2P
ASR1000(config-cmap)# match protocol attribute sub-category p2p-file-transfer
 ! This class map matches any P2P traffic with specific attention
 ! on the file-transfer sub-category

ASR1000(config)# policy-map CHILD-PRIORITY-POLICY
ASR1000(config-pmap)# class P2P
ASR1000(config-pmap-c)# bandwidth remaining percent 20
 ! This allocates 20% of remaining bandwidth to P2P class
ASR1000(config-pmap-c)# class BROWSING
ASR1000(config-pmap-c)# bandwidth remaining percent 80
 ! This allocates 80% of remaining bandwidth to BROWSING class

ASR1000(config-pmap-c)# policy-map PARENT-PRIORITY-POLICY
ASR1000(config-pmap-c)# class class-default
ASR1000(config-pmap-c)# shape average 50000000
 ! The Internet pipe is shaped down to 50 Mbps
ASR1000(config-pmap-c)# service-policy CHILD-PRIORITY-POLICY
 ! The parent policy (PARENT-EDGE-POLICY) first shapes the pipe
 ! down to 50 Mbps, then when back pressure is applied it calls
 ! the child policy (CHILD-PRIORITY-POLCIY) to prioritize traffic
 ! within the shaped rate

ASR1000(config)# interface GigabitEthernet0/0/2
ASR1000(config-if)# service-policy output PARENT-EDGE-POLICY
ASR1000(config-if)# description The Internet Facing Interface
 ! This attaches the INTERNET-EDGE-POLICY map to the Internet
 ! interface in the outbound direction
```

As before, you can verify this configuration with the following commands:

```
show class-map
show policy-map
show policy-map interface GigabitEthernet0/0/2
```

Another way to confirm this configuration has worked is to examine the results that are collected in the Cisco Insight Reporter. Figure 9-7 illustrates a report of "before and after" the QoS policy was applied to the interface.

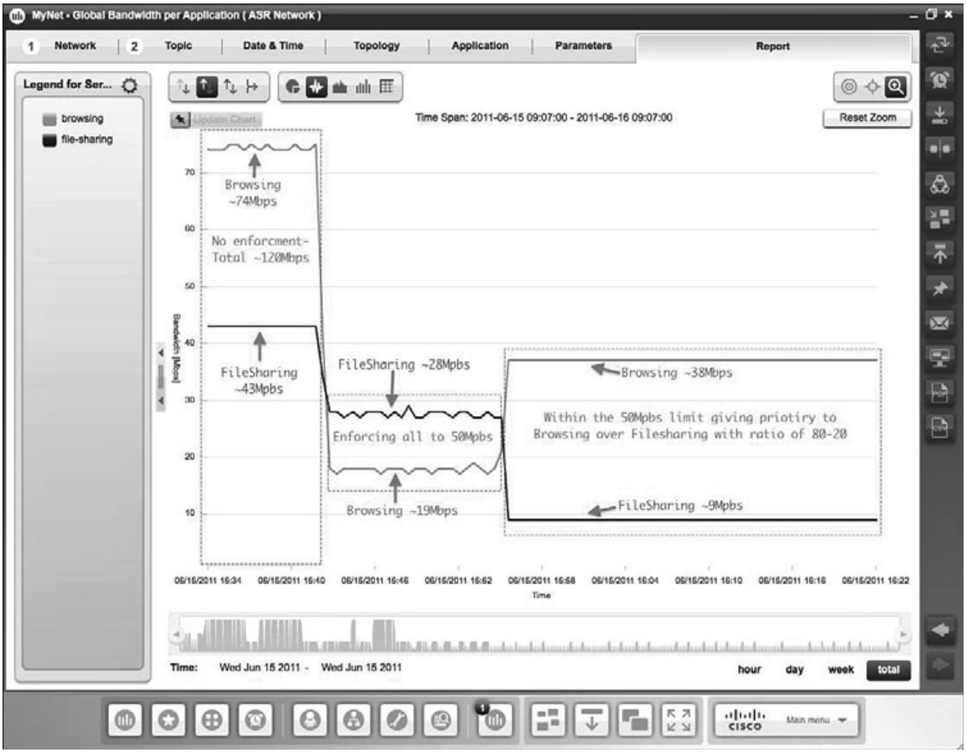


Figure 9-7 *Insight Reporter Showing the Effect of the AVC QoS Policy*

As you can see in Figure 9-7, before the QoS policy was put in place, the router's total outbound traffic volume reaches close to 120 Mbps in an environment where the total available bandwidth is only 50 Mbps—meaning that up to 70 Mbps of traffic is being dropped by the ISP. It is easy to see where the QoS policy is enabled because the bandwidth consumption changed drastically. First, the total outbound bandwidth is shaped to 50 Mbps, then the hierarchical policy was enabled. This has the effect of queuing the traffic classes as expected:, 80 percent for web browsing (equating to a maximum of 40 Mbps) and 20 percent for P2P file sharing (equating to a maximum of 10 Mbps).

This example shows how AVC is able to easily detect and then control multiple applications that are using HTTP as the fundamental transport mechanism.

Performance Considerations When Using AVC

Whenever NBAR2 or FNF are used in the network, care must be given to the performance impact it has on the device that is running it. Because NBAR2 is performing application layer DPI, there could be a considerable performance impact on both the CPU and the available system memory.

Because NBAR2 needs to inspect the entire connection flow, data for that flow is stored in DRAM, which is a finite resource. In each system, there is a default amount of memory allocated to hold NBAR2 flow information. Instead of being allocated as actual memory space, it is displayed as the maximum number of NBAR2 flows that are supported by the platform. For example, the maximum number of NBAR2 flows that are supported in an ASR 1000 can be displayed with the **show ip nbar resources flow** command, as demonstrated in Example 9-18.

Example 9-18 *Displaying the NBAR System Resources*

```
ASR1002# show ip nbar resources flow
NBAR flow statistics
        Maximum no of sessions allowed : 1000000
        Maximum memory usage allowed   : 367001 KBytes
        Active sessions                : 0
        Active memory usage            : 83453 KBytes
        Peak session                   : 4124
        Peak memory usage              : 83453 Kbytes
```

The key parts of this output are the maximum number of sessions allowed (which in this case is one million), and the peak number of sessions that are currently in use. As a general rule, it is recommended not to exceed 70 percent of available memory allocated to NBAR. If the number of active flows exceeds the maximum number allowed for NBAR, two things will happen:

- The router will generate a syslog warning you of the issue.
- The sessions in excess of the allowed amount will be classified as Unknown.

In addition to memory considerations, NBAR and FNF may also have a significant performance impact on the CPU. In an ISR G2 router, this is particularly important to watch because the CPU handles both the data plane and control plane. Therefore, on an ISR G2 router, the performance impact of NBAR has the potential to directly impact both the data forwarding performance and other control protocols, such as SNMP, routing protocols, and so on. On the ISR G2, you can monitor the CPU with the well-known **show proc cpu** command.

On the ASR family, the control and data forwarding plane are separated. On this platform, NBAR is performed on the Embedded Services Processors (ESP) module that is responsible for data forwarding, so the output of **show proc cpu** will be of little help (because this output shows the CPU performance of the route processor module, not the ESP). On the ASR, use the command **show platform hardware qfp active datapath** to examine the CPU performance of the ESP module itself to see what impact NBAR is having on the system, as demonstrated in Example 9-19.

Example 9-19 *Examining ASR 1000 ESP CPU and Memory Performance*

```
avc-asr1002a# show platform hardware qfp active datapath utilization summary
  CPP 0:                5 secs     1 min     5 min    60 min
Input:     Total (pps)       98        45        39        39
                 (bps)   325592    234000    103432    103400
Output:    Total (pps)       99        46        40        40
                 (bps)   343368    240440    111376    111328
Processing: Load (pct)        0         0         0         0
```

As highlighted in Example 9-19, the Processing: Load output shows you the CPU and memory performance of both the ESP switch forwarding fabric, but not the RP (you primarily care about the ESP performance when running NBAR2).

As a general rule, before enabling AVC on your routers or wireless LAN controllers, it is recommended to look up the NBAR2 and FNF performance characteristics of each platform to ensure that these features do not negatively impact the performance of your network.

Summary

This chapter introduced the concept and technology components of the AVC solution. As business and personal applications continue to evolve, more and more are now web based. This trend has rendered the traditional ways of application identification, based on TCP or UDP port numbers, somewhat limited in its capabilities. AVC was developed by Cisco to address this situation and to provide DPI capabilities on networking devices to enable application layer QoS classification and controls.

Four common use cases for AVC were discussed, including the use of AVC at the Internet edge, WAN edge or branch routers, and on wireless controllers. Today, AVC is widely support on the ASR 1000 family, ISR G2 routers, and on the Cisco wireless LAN controllers.

The four building blocks of AVC were next discussed. These are NBAR2, FNF, reporting tools, and QoS controls. The structure of NBAR2 was examined, and several NBAR2 configuration examples were provided. The chapter also discussed when to use NBAR protocol discovery. FNF was discussed in detail, including some key differences between classic NetFlow and FNF. You also learned how FNF uses the NBAR2 DPI engine to discover and classify applications on the network. Several configuration examples were given that illustrated a realistic deployment of FNF. In addition, the FNF reporting and collecting tools were introduced, including Cisco Insight. In the final building block, a demonstration of how to use the information collected by the FNF reporting tools was shown. This included two examples of how to implement QoS controls on application layer protocols at both the Internet edge and the WAN edge.

Finally, the effect of AVC on router and WLC performance was discussed. AVC could have a possibly detrimental effect on both CPU and DRAM if care is not taken. It was recommended to carefully examine the AVC performance characteristics of each platform before configuring it on any device.

Additional Reading

FNF technology white paper: http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/ps6965/prod_white_paper0900aecd804be1cc.html

NBAR2 protocol library: http://www.cisco.com/en/US/partner/prod/collateral/iosswrel/ps6537/ps6558/ps6616/product_bulletin_c25-627831.html

NBAR2 overview: http://www.cisco.com/go/nbar

Cisco Insight Reporter: http://www.cisco.com/en/US/prod/collateral/ps7045/ps6129/ps6257/ps6135/cisco_insight.html

This page intentionally left blank

Chapter 10

Business and Application QoS Requirements

This chapter examines current business trends impacting QoS designs and various application-class QoS requirements, including the following:

- Global trends in networking
- The evolution of video applications
- The explosion of media
- The phenomena of social networking
- The bring your own device demand
- The emergence of bottom-up applications
- The convergence of media subcomponents within multimedia applications
- The transition to high-definition media
- QoS requirements and recommendations by application class

QoS has proven itself a foundational network infrastructure technology required to support the transparent convergence of voice, video, and data networks. Furthermore, QoS has also been proven to complement and strengthen the overall security posture of a network; however, as business needs continue to evolve and expand, so does the role and requirements of QoS policies.

Successful QoS deployments are driven first and foremost by business/organizational objectives. Therefore, before diving into technical designs and configurations, it is beneficial for a network administrator to overview some of the business trends and application requirements relating to network QoS policies. After all, these requirements will shape the strategy and individual designs of the QoS deployment.

Global Trends in Networking

Cisco—in conjunction with top-tier service providers and large-enterprise customers—monitors Internet traffic statistics and trends and publishes an annual Cisco Visual Networking Index: Forecast and Methodology Report. These reports can be viewed at http://www.cisco.com/go/vni.

Highlights from the current report (at the time of this writing) reveal significant global trends in networking that have a direct impact on QoS requirements and designs, including the following:

- Global IP traffic has increased eightfold over the past 5 years, and will increase threefold over the next 5 years. (Overall, IP traffic will grow at a compound annual growth rate of 29 percent from 2011 to 2016.)
- The number of devices connected to IP networks will be nearly three times as high as the global population in 2016.
- A growing amount of IP and Internet traffic is originating with non-PC devices. (In 2011, 22 percent of IP traffic originated with non-PC devices, but by 2016 the non-PC share of IP traffic will grow to 31 percent.)
- Traffic from wireless devices will exceed traffic from wired devices by 2016.
- Video exceeded half of global consumer Internet traffic by year-end 2011. The sum of all forms of video will be approximately 86 percent of global consumer traffic by 2016.
- High-definition video on demand (VoD) surpassed standard definition by the end of 2011. By 2016, high-definition Internet video will comprise 79 percent of VoD.

The Evolution of Video Applications

When the first edition of this book was published (in 2004), there were basically only two broad types of video applications deployed over business networks:

- **Streaming video:** Generally describing unidirectional time-insensitive streaming flows, such as unicast or multicast VoDs. (Extensive application-level buffering compensated for excessive jitter.)
- **Interactive video:** Generally describing bidirectional and time-sensitive collaborative video flows, such as H.323-based video applications (typically operating at 384 Kbps or 768 Kbps)

Since then, however, video applications have evolved considerably—and continue to evolve—as illustrated in Figure 10-1.

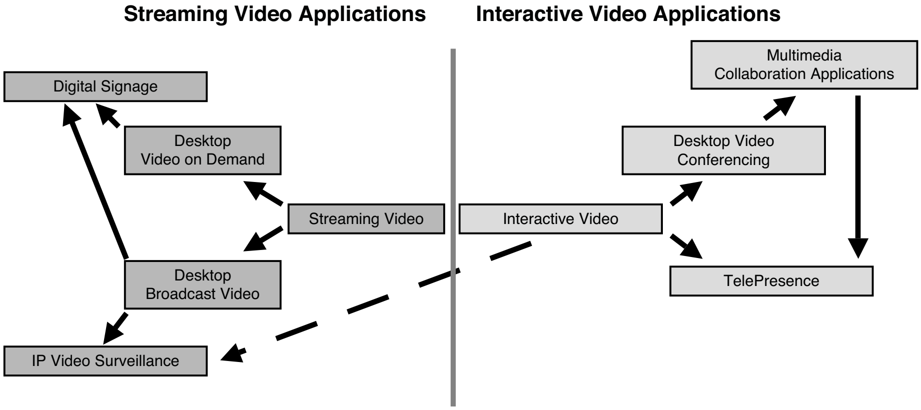


Figure 10-1 *Video Application Evolution*

Consider first the streaming video branch—the earliest sets of video applications were VoD streams to the desktop. VoD streams can include prerecorded content such as employee communications, training, e-learning, and social-interaction content. Today, due to the ease of content creation, on-demand content can either be professionally produced (top down) or self-produced (bottom up). It is important to also note that not all VoD content is necessarily business related; nonbusiness, entertainment-oriented content is widely available for on-demand video viewing (YouTube being the prime example). VoD applications may include live/broadcast video streams to the desktop. Broadcast streams may include company meetings, special events, internal corporate announcements, or similar content. Broadcast streaming video content is therefore usually professionally produced top-down content.

Thereafter, with the proliferation of flat-screen digital displays, it became increasingly apparent that the desktop is not the only display option for streaming video. Thus, digital signage began to emerge as another streaming video application (for both on-demand and broadcast video streams). Digital signage refers to centrally managed publishing solutions for delivering digital media to networked displays. Digital signage can be used to broadcast internal information, such as sharing up-to-date schedules and news where people need it most or providing real-time location and directional guidance. In addition, digital signage is an effective tool for marketing and advertising.

Around the same time that digital signage was being developed, the advantages that IP brought to video were being gradually being applied to the video-surveillance market. These advantages include the ability to forward live video streams to local or remote control centers for observation and efficient processing. Interestingly, video surveillance has a unique degree of interactivity not found in any other streaming video application—namely, that of having an observer "interact" with the video stream by sending control information to the transmitting video camera (for instance, to track an event in progress).

On the interactive video side of the video application hemisphere, there has also been considerable application evolution. Basic video-conferencing applications, which were initially dedicated room-based units, evolved into software-based PC applications. The factors behind this shift from room-based hardware to PC-based software were twofold:

- The convenience of immediate desktop collaboration (rather than having to book or hunt for an available video-conferencing-enabled room)
- The availability of inexpensive webcams

Desktop video conferencing may be used on a one-to-one basis or may support a few participants simultaneously.

Once video conferencing moved to software, a whole new range of communication possibilities opened up, which morphed desktop video-conferencing applications into multimedia collaboration applications. Multimedia collaboration applications share not only voice and video but also data applications, such as instant messaging, document and presentation sharing, application sharing, and other integrated multimedia features.

However, not all interactive video migrated to the desktop. Room-based video-conferencing solutions continued to evolve and leveraged advances in high-definition video and audio, leading to solutions like Cisco TelePresence. In addition, application-sharing capabilities—borrowed from multimedia-conferencing applications—were added to these high-definition room-based video-conferencing solutions.

Finally, it bears mentioning that video application evolution doesn't end here; it will no doubt continue to expand and morph over time as new demands and technologies emerge.

The Explosion of Media

Another factor driving the demand for rich-media applications over IP networks is the sheer explosion of media content, due largely to the removal of barriers of video production, distribution, and consumption.

Consider first how barriers to video production have been dramatically lowered: Two decades ago, video cameras were quite expensive, and only studios had any video-production/editing tools available to them. However, today almost every mobile phone, tablet, laptop, and digital still camera provides high-definition video-capture capability. Furthermore, with off-the-shelf software, amateur video users can produce high-quality final products that can rival those of professional studios.

Barriers to content distribution have all but disappeared with the advent of social networking sites like Facebook, YouTube, and others, which allow users to immediately post content with the touch of a button.

And finally, barriers to video consumption have also vanished. It's almost amusing to recall that it's only been about 10 to 15 years that bulky TV screens were the primary hardware on which to watch videos. Today, however, video content can be viewed on nearly any device, at any time and in any location.

The bottom line is that almost no barriers exist to inhibit video communication and therefore this incredibly effective medium is exploding over networks.

The Phenomena of Social Networking

Social networking may have started as a consumer phenomenon, but its appearance and effect on business networks was virtually inevitable as users began to number into the hundreds of millions.

Consider this: Via social networking websites, people generally share information about themselves and the things they are interested in, which facilitates interaction with others who share similar interests. Correspondingly, within the workplace, corporate directories and collaborative internal sites are mirroring the look, feel, and functionality of such social networking sites.

For example, Cisco has experienced the crossover of such social networking applications into the workplace, with applications like Cisco Vision (C-Vision). C-Vision started as an ad hoc service by several employees, providing a central location for employees to share all forms of media with one another, including video clips. Cisco employees shared information on projects, new products, competitive practices, and many other subjects via this service. The service was used by so many employees that Cisco's IT department had to assume ownership and subsequently scaled the service globally within Cisco. The result is a service where employees can become more effective and productive, quickly tapping into each other's experiences and know-how, all through the effectiveness and simplicity of rich-media applications.

The Bring Your Own Device Demand

Previously, employers provided desktop and laptop computers that were usually the most advanced tools to which an employee had access. With the explosion in wireless consumer devices, including laptops, netbooks, tablets, smartphones, e-readers, and others, employees typically have some of the most advanced productivity tools being used in their personal lives. Employees quickly asked their IT organizations: Why can't I use these tremendous productivity tools at work? Many IT organizations initially rejected the idea, citing security reasons and the inability to scale to approve and support more than a small handful of devices.

In recent years, the persistence of end users demanding to leverage their tablet computers and smartphones to extend their productivity, even if they had to purchase the devices themselves, has led many IT departments to adopt less-restrictive policies, allowing employees basic connectivity or, increasingly, full access to the IT network and corporate applications. This trend—dubbed bring your own device (BYOD)—is likely irreversible.

Many people had a desktop PC or laptop and added a mobile phone for voice calls. Mobile phones have largely been replaced with smartphones, which often are as powerful and capable as laptops and PCs, enabling a new class of uses and applications

There is speculation that in the future a single device will be used for all needs: computing, communications, and applications; however, today most believe there will continue to be different devices best suited to particular uses. For example, a laptop is not as portable as a smartphone, so people are likely to carry their smartphone for mobile communications. Tablets are powerful devices, as well, but it is likely laptops and PCs will still be used for document creation and publishing. This means people will more likely carry and use multiple devices and less likely that a single all-purpose device will emerge.

Increasingly, work is an activity that people do, not a place to which they go. Extended connectivity through mobile and remote access to the corporate network gives employees tremendous flexibility and increased productivity. It also leads to a blurring of the line between work time and personal time, with employees trading set work schedules for the flexibility of working when and where they want to, often interweaving work and personal tasks.

These trends in multiple per-user devices that may be used for both work and personal purposes presents not only extensive security challenges to administrators but also QoS challenges—particularly in considering which types of devices may be trusted. And which applications on these types of BYOD devices may be trusted.

The Emergence of Bottom-Up Applications

Traditionally, IT departments would dictate which applications and devices would be supported within their business environments; such deployments are sometimes referred to as top-down deployments.

In contrast, and as demonstrated in the C-Vision example, there is a growing trend of users driving application deployments within the enterprise from the bottom up. In other words, the user base either demands or just begins to use a given application, with or without formal management or IT support.

The combination of top-down and bottom-up applications and device proliferation places a heavy burden on the IT department as it struggles to cope with officially supported and unsupported-yet-highly-proliferated applications and devices.

The Convergence of Media Subcomponents Within Multimedia Applications

In the early days of converged networking, applications fit cleanly into their respective media buckets: Applications were voice, video, or data—but not combinations of these. This was primarily because hardware was dedicated to application types. For example, IP phones generated voice traffic, dedicated IP/VC equipment generated video, and PCs generated (predominantly) data traffic.

Today, however, because of the power and flexibility of hardware platforms—including laptops, tablets, and smartphones—many applications are multimedia hybrids, combining two or more of these media types. For example, Cisco WebEx can send video, audio, and data all at the same time. Similarly, Cisco TelePresence has high-definition audio and video flows and data-sharing capabilities.

Such multimedia applications present a dilemma to the network administrator: should he classify such applications as voice (to preserve the quality of the voice subcomponent)? Or as video (for the same reason)? Or simply as a data application (the lowest common denominator—from a QoS standpoint)? Or should he try to split out the media subcomponents of the application and provision these into separate classes of service? The QoS impact of these options is discussed in more detail in the "Multimedia Applications" section later in this chapter.

The Transition to High-Definition Media

One of the reasons traditional room-to-room video conferencing and desktop webcam-style video conferencing are sometimes questioned as less than effective communications systems is the reliance on standard or low-definition audio and video formats.

However, high-definition interactive media applications, like Cisco TelePresence, aptly demonstrate how high-definition audio and video can create a more authentic remote collaboration experience, where participants actually feel like they are in the same meeting room. In addition, IP video-surveillance cameras are migrating to high-definition video to have the digital resolutions needed for new functions, such as facial recognition and intelligent event triggering based on motion and visual characteristics.

Also, as previously noted, high-definition VoDs have already surpassed standard-definition VoDs over the Internet. Ss people become accustomed to high-definition video quality as consumers, they will increasingly demand it in their corporate environments as well.

QoS Requirements and Recommendations by Application Class

After having considered some high-level business trends affecting QoS, it is often helpful to review some application-level requirements so that the deployed QoS designs will sufficiently accommodate these.

End-to-end network applications can—for the most part—be grouped in five generic categories:

- Voice
- Video applications
- Multimedia applications
- Data applications
- Control plane traffic

Note This is not to say that other application types do not exist on the network. Some of these, however, may be local to a specific place in the-network (or PIN). For example, Fibre Channel over Ethernet (FCoE) is a unique application protocol, but is present only in the data center, and therefore is discussed in Part V, "Data Center QoS Design," of this book, rather than as an end-to-end application class.

Each of these application classes is discussed in turn.

Voice

The following list summarizes the key QoS requirements and recommendations for voice (bearer) traffic:

- **Voice requirements:**
    - One-way latency should be no more than 150 ms.
    - One-way peak-to-peak jitter should be no more than 30 ms.
    - Per-hop peak-to-peak jitter should be no more than 10 ms.
    - Packet loss should be no more than 1 percent.
    - A range of 20 to 320 Kbps of guaranteed priority bandwidth is required per call (depending on the sampling rate, the codec, and Layer 2 media overhead).
- **Voice recommendations (per RFC 4594 and RFC 3246):**
    - Voice traffic should be marked to Expedited Forwarding (EF) / differentiated services code point (DSCP) 46.
    - Voice should be treated with an EF per-hop behavior (PHB) (strict-priority queuing).
    - Voice should be admission controlled.

Latency can cause people to talk over one another if it is excessive. The goal commonly used in designing networks to support VoIP is the target specified by ITU standard G.114, which states that 150 ms of one-way end-to-end (from mouth to ear) delay ensures user satisfaction for telephony applications. A design should apportion this budget to the various components of network delay (propagation delay through the backbone, scheduling delay because of congestion, and serialization delay) and service delay (because encoding, decoding, and the de-jitter buffer).

VoIP streams use jitter buffers to reduce the effects of network jitter. Jitter buffers are used to change asynchronous packet arrivals into a synchronous stream by turning variable network delays into constant delays at the destination end systems. The role of the jitter buffer is to trade off between delay and the probability of interrupted playout because of late packets. Late or out-of-order packets are discarded.

Loss causes voice clipping and skips for VoIP packets. Packet-loss concealment (PLC) is a technique used to mask the effects of lost or discarded VoIP packets. The method of PLC used depends on the type of codec. A simple method used by waveform codecs such as G.711 is to replay the last received sample with increasing attenuation at each repeat; the waveform does not change much from one sample to the next. This technique can be effective at concealing the loss of up to 20 ms of samples.

Bandwidth for voice packets can be provisioned very accurately because voice packets are constant in size (a function of the codec) and constant in packet rate (by default 50 pps). When provisioning bandwidth for voice, however, the network administrator should remember to account for Layer 2 network protocol overhead.

RFC 4595 recommends that the voice class be marked EF (DSCP 46) and admission controlled. This class is recommended to be provisioned with an EF PHB. The EF PHB—defined in RFC 3246—is a strict-priority queuing service and so admission to this class should be controlled. Example traffic includes G.711 and G.729a VoIP calls.

Video Applications

Video applications—particularly high-definition video applications—create unique challenges and demands on IP networks, as can be seen by comparing voice and video at the packet level, as shown in Figure 10-2.

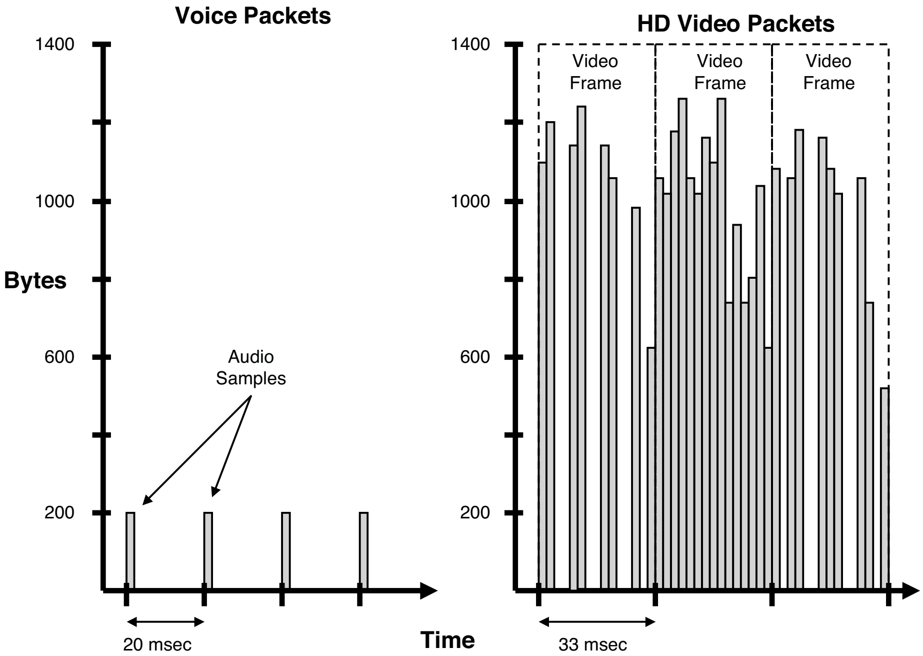


Figure 10-2 *VoIP Versus HD Video (At the Packet Level)*

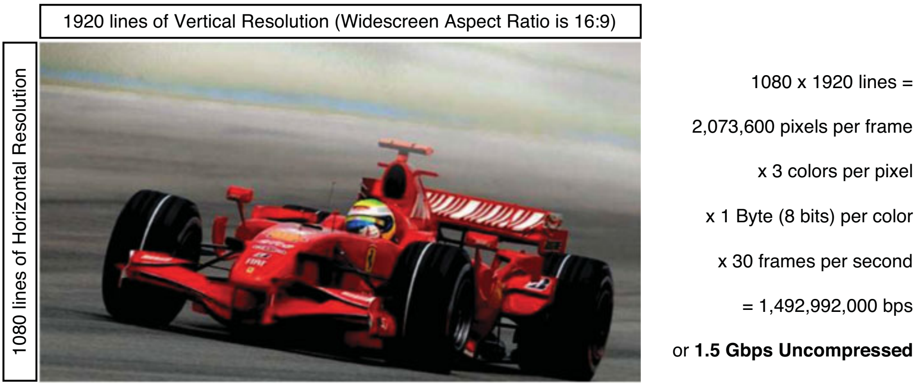
From Figure 10-2, it becomes apparent that—unlike VoIP packets—video packets vary in both packet sizes and packetization rates. This is because the amount of video information sent will continually vary in its degree of compression. The amount of compression achievable is essentially random, as it varies according to the video images themselves, including how many colors and textures are present, how much motion is occurring, and any panning, tilting, and zooming that may be in operation.

Video codecs use both spatial and temporal compression techniques:

- Spatial compression compresses similar groups of adjacent pixels within a single frame of video.
- Temporal compression compresses similar groups of pixels from one video frame to the next (or several frames of video).

As a result of both spatial and temporal compression techniques, video codecs can achieve impressive compression ratios, without which it would be nearly impossible to send video over even Gigabit Ethernet (GE) or 10GE networks.

For example, let's consider the case of compressing a full HD (1080p30) video stream. The first parameter (1080) refers to 1080 lines of horizontal resolution, which are factored by 1920 lines of vertical resolution (as per the 16:9 widescreen aspect ratio used in HD video formatting), resulting in 2,073,600 pixels per screen. The second parameter, p, indicates a progressive scan, which means that every line of resolution is refreshed with each frame (as opposed to an interlaced scan, which would be indicated with an i and would mean that every other line is refreshed with each frame). The third parameter 30 refers to the transmission rate of 30 frames per second. Although video-sampling techniques may vary, each pixel has approximately 3 bytes of color/luminance information. When all this information is factored together (2,073,600 pixels × 3 bytes × 8 bits per byte × 30 frames per second), it results in approximately 1.5 Gbps of information, as illustrated in Figure 10-3.


1920 lines of Vertical Resolution (Widescreen Aspect Ratio is 16:9)
1080 lines of Horizontal Resolution

1080 x 1920 lines =

2,073,600 pixels per frame

x 3 colors per pixel

x 1 Byte (8 bits) per color

x 30 frames per second

= 1,492,992,000 bps

or **1.5 Gbps Uncompressed**

Figure 10-3 *Full HD (1080p30) Uncompressed*

In contrast, various codecs, such as H.264, can compress such HD video flows to less than 5 Mbps—which represents a compression ratio of over 300:1!

The downside to such intense compression techniques is that these algorithms are highly sensitive to packet loss. So, lost information is generally exaggerated—in exactly the reverse operation as it is compressed (that is, both spatially and temporally). For instance, if a lost packet represents a significant degree of spatial compression, the resulting video image will show a large block-like artifact within a frame of video. If a lost packet represents a significant degree of temporal compression, the artifact will linger over several frames, until an uncompressed video frame (sometimes called a key frame) finally clears it.

Cisco lab testing has shown even a single HD video packet lost in 10,000 can be visually detected by an end user. Compared to VoIP—where 1 packet lost in 100 could be concealed from the end user—it can be said that HD video is 100 times more sensitive to packet loss than VoIP! Nonetheless, typical packet loss targets for HD video applications are generally set at 0.1 percent (which is only 1/10th the packet loss target of VoIP), as a minor amount of visual artifacts are generally acceptable during network busy periods; should these, however, not be acceptable, the packet loss target would have to be tightened accordingly.

Video applications can be broken down further into the broadcast video (unidirectional video) application class and the real-time interactive (bidirectional video) application classes.

Note These video application classes should not be construed as containing *only* video traffic, because these may also contain additional data flows (such as for sharing slides or other data along with the video stream).

Broadcast Video

The broadcast video application class is intended for unidirectional and inelastic video flows, such as broadcast TV, live events, and IP video surveillance.

Note Flow elasticity refers to an application's capability to adapt to detect packet loss. *Inelastic* flows either lack such ability or may be deliberately designed not to reduce quality in the event of packet loss. In contrast, *elastic* flows react to packet loss by reducing quality, resolution, and frame rates or by performing some other downgrading function.

Broadcast video requirements include the following:

- Packet loss should be no more than 0.1 percent.

Broadcast video recommendations (per Cisco's interpretation of RFC 4594* and RFC 3246) include the following:

- Broadcast video traffic should be marked to CS5 / DSCP 40.
- Broadcast video *may* be treated with an EF PHB (strict-priority queuing).
- Broadcast video should be admission controlled.

Note As discussed in Chapter 3, "Classification and Marking," Cisco has adopted RFC 4594 as its general differentiated services (DiffServ) QoS strategy, with the following exception: Cisco has swapped the marking recommendations of the RFC 4594 Broadcast Video class (of CS3) with the Signaling class (of CS5); therefore, Cisco has decided to mark broadcast video traffic as CS5 and signaling traffic as CS3. This is primarily because Cisco has been marking signaling to CS3 for over a decade (well before RFC 4594 was even drafted) and lacking a compelling business case to change the defaults on all its voice and video telephony products—which would correspondingly force their customer base to change all their network QoS policies relating to the Signaling class as well—has decided to continue doing so. Therefore, Cisco has swapped these marking recommendations such that it marks broadcast video to CS5 and continues to mark signaling to CS3. It is important to remember that RFC 4594 is an informational RFC and not a standard and therefore compliance—in full or in part—is not mandatory.

Broadcast video flows are usually unidirectional and include application-level buffering (as common to streaming video applications). So, this class does not have strict latency or jitter requirements. However, the stringent loss requirements (particularly for HD flows assigned to this class) may require the provisioning of a strict-priority service.

Traffic in this class should be marked Class Selector 5 (CS5/DSCP 40) and may be provisioned with an EF PHB (that is, a strict-priority service treatment, which may also be provisioned for traffic that is not necessarily marked to EF/DSCP 46). Admission to this class should be controlled (either by an admission control mechanisms or by explicit bandwidth provisioning). Example traffic includes live Cisco Digital Media System (DMS) streams to desktops or to Cisco Digital Media Players (DMPs), live Cisco Enterprise TV (ETV) streams, and Cisco IP Video Surveillance (IPVS).

Real-Time Interactive

The Real-Time Interactive service class is intended for real-time, bidirectional, inelastic, video applications. These flows are usually HD.

Real-time interactive video requirements include the following:

- One-way latency should be no more than 200 ms.
- One-way peak-to-peak jitter should be targeted at no more than 50 ms.

- Per-hop peak-to-peak jitter should be targeted at no more than 10 ms.
- Packet loss should be no more than 0.1 percent.
- Provisioned bandwidth will vary based on codec, resolution, frame rates, and additional data components and network overhead.

Real-time interactive video recommendations (per RFC 4594 and RFC 3246) include the following:

- Real-time interactive traffic should be marked to CS4 / DSCP 32.
- Real-time interactive *may* be treated with an EF PHB (strict-priority queuing).
- Real-time interactive should be admission controlled.

The latency target for the Real-Time Interactive class of 200 ms is relatively the same as the VoIP class; however, a moderate allowance is made to accommodate the extended processing times required by HD video codecs and compression algorithms. Similarly, the jitter target is quite similar to VoIP, but makes allowance for the serialization of larger packet sizes that generally comprise HD video packets. The loss target is the same as the Broadcast Video class at no more than 0.1 percent.

Note HD video is sent as frames, which typically span over multiple packets. Therefore, the key jitter metric in video quality is not necessarily individual packet jitter, but overall video-frame jitter. In other words, the entire frame of video (spanning multiple packets) must be received within the replay buffer's jitter allowance for the quality to be maintained.

Traffic in the Real-Time Interactive class should be marked CS4 (DSCP 32) and may be provisioned with an EF PHB, and therefore admission to this class should be controlled. An example application for this class is Cisco TelePresence.

Multimedia Applications

Because so many applications today have multimedia components, administrators often wrestle with the QoS dilemmas these present: Should they provision these applications in the voice or video classes to preserve the quality of the voice or video components? Or should they leave them as data? Or should they try to split out the media subcomponents of the application and provision these into separate classes of service?

Each option presents tradeoffs. The first (of provisioning multimedia applications within the voice or video classes) could potentially require significant real-time bandwidth increases (especially if these classes are being provisioned with an EF PHB). In addition, admission control mechanisms may only be supported to a limited extent (if at all) by many multimedia applications, which makes provisioning these in the voice and video classes quite risky to the quality of voice and video applications already deployed.

The second option (of just leaving these applications provisioned as data) may not provide adequate quality for the audio and video components of these multimedia applications.

The third option (of separating media components and provisioning these into separate classes of service) may seem excessively complex and might not even be technically feasible (because often media flows within an application may utilize random TCP/UDP ports or be multiplexed together). In addition, the admission control problem would remain. And finally, even if media component separation were feasible and controllable, it may still not be desirable because of a potential a lack of synchronization at the receiving end. (For example, audio and video flows may arrive at different times due to being treated differently while transiting the network.)

An alternative to these options is to recognize the unique requirements of multimedia applications and provision these accordingly. This, in fact, is the approach recommended by RFC 4594—namely, to provision multimedia applications into one of two separate multimedia traffic classes:

- Multimedia Conferencing, for bidirectional multimedia applications
- Multimedia Streaming, for unidirectional multimedia applications

The sections that follow provide overviews of these two multimedia classes.

Multimedia Conferencing

The Multimedia Conferencing service class is intended for elastic, bidirectional multimedia applications, such as desktop software multimedia/collaborative applications.

Multimedia conferencing requirements include the following:

- One-way latency should be no more than 200 ms.
- Packet loss should be no more than 1 percent.

Multimedia conferencing recommendations (per RFC 4594 and RFC 2597) include the following:

- Multimedia conferencing traffic should be marked to the AF4 class (AF41/AF42/AF43 or DSCP 34/36/38, respectively).
- Multimedia conferencing should be treated with an Assured Forwarding (AF) PHB, provisioned with a guaranteed-bandwidth queue with DSCP-based weighted random early detection (WRED).
- Multimedia conferencing should be admission controlled.

Traffic in this class should be marked as Assured Forwarding Class 4 (AF41/AF42/AF43 through DSCP 34/36/38, respectively) and should be provisioned with a guaranteed bandwidth queue with DSCP WRED enabled. Admission to this class should be controlled. In addition, traffic in this class may be subject to policing and re-marking. Example applications include Cisco Unified Personal Communicator, Cisco Unified Video Advantage, and the Cisco Unified IP Phone 7985G.

Multimedia Streaming

The Multimedia Streaming service class is intended for elastic, unidirectional multimedia applications, such as VoD streaming video flows.

Multimedia streaming requirements include the following:

- One-way latency should be no more than 400 ms.
- Packet loss should be no more than 1 percent.

Multimedia streaming recommendations (per RFC 4594 and RFC 2597) include the following:

- Multimedia streaming traffic should be marked to the AF3 class (AF31/AF32/AF33 or DSCP 26/28/30, respectively).
- Multimedia streaming should be treated with an AF PHB, provisioned with a guaranteed-bandwidth queue and DSCP-based WRED.
- Multimedia streaming may be admission controlled.

Traffic in this class should be marked as AF Class 3 (AF31/AF32/AF33 through DSCP 26/28/30, respectively) and should be provisioned with a guaranteed-bandwidth queue with DSCP-based WRED enabled. Admission control is recommended on this traffic class (though not strictly required), and this class may be subject to policing and re-marking. Example applications include Cisco Digital Media System VoD streams.

Data Applications

Most enterprises have several thousand data applications traversing their networks. So, it is impossible to apply a general characterization to these: Some applications (such as trading applications in investment banks) require real-time service, other applications (such as scientific research applications) may not require real-time servicing but do require a steady lossless service. And there are thousands of applications in between.

For decades, the concept of *mission-critical* data applications seemed to be the industry norm: certain applications would be set apart from others by applying a subjective "business relevance" factor, and these would be marked and provisioned in a dedicated service class. However, in reality, this concept often did more harm than good when it came to deploying QoS. This is because the inclusion of a purely nontechnical subjective factor usually sparked significant political and organizational debate and would often result in excessive delays in QoS deployments. For example, many (if not most) department heads would insist that *their* applications merited special status because of business relevance. However, if too many applications would be assigned to a dedicated mission-critical class, eventually these applications would simply be contending among themselves for a first in, first out (FIFO) queue. (In other words, the business would—to a large extent—be back to where it started before QoS was deployed.) Higher-level (and nontechnical) executives were often called on to make network QoS decisions, more so to appease their people than to optimize the network.

Much to their credit, the authors of RFC 4594 did away with the Mission-Critical Data class, preferring instead to view all data applications as business critical (with the exception of best effort and scavenger) and provision for these along purely technical requirements in one of two classes:

- Transactional Data (or Low-Latency Data)
- Bulk Data (or High-Throughput Data)

The sections that follow discuss these two data classes in turn.

Transactional Data (Low-Latency Data)

The Transactional Data service class (referred to as the Low-Latency Data service class in RFC 4594) is intended for interactive foreground data applications. *Foreground* applications refer to applications from which users are expecting a response—via the network—to continue with their tasks. Excessive latency in response times of foreground applications directly impacts user productivity.

Transactional data recommendations (per RFC 4594 and RFC 2597) include the following:

- Transactional data traffic should be marked to the AF2 class (AF21/AF22/AF23 or DSCP 18/20/22, respectively).
- Transactional Data should be treated with an AF PHB, provisioned with a guaranteed-bandwidth queue and DSCP-based WRED.

Traffic in this class should be marked AF Class 2 (AF21/AF22/AF23 or DSCP 18/20/22, respectively) and should be provisioned with a dedicated-bandwidth queue with DSCP WRED enabled. This traffic class may be subject to policing and re-marking. Example applications include data components of multimedia collaboration applications, enterprise resource planning (ERP) applications, customer relationship management (CRM) applications, database applications, and so on.

Bulk Data (High-Throughput Data)

This Bulk Data service class (referred to as the High-Throughput Data service class in RFC 4594) is intended for noninteractive background data applications.

Bulk data recommendations (per RFC 4594 and RFC 2597) include the following:

- Bulk data traffic should be marked to the AF Class 1 (AF11/AF12/AF13 or DSCP 10/12/214, respectively).

- Bulk data should be treated with an AF PHB, provisioned with a guaranteed-bandwidth queue and DSCP-based WRED.
- Bulk data is usually provisioned in a moderately provisioned queue to provide a degree of bandwidth constraint during periods of congestion, so as to prevent long TCP sessions from dominating network bandwidth.

The term *background applications* refers to applications from which users are not awaiting a response—via the network—to continue with their tasks. Such traffic includes any type of machine-to-machine communication, and therefore excessive latency in response times of background applications does not directly impact user productivity. Furthermore, because most background applications are TCP based, these applications (if left unchecked) could consume excessive network resources away from more interactive foreground applications. Assigning these bulk data flows to a moderately provisioned guaranteed-bandwidth queue allows these to be serviced according to their bandwidth requirements until link utilization has been maximized (this is because queuing policies only engage during periods of congestion), at which point these flows are constrained so as to prevent them from interfering with more-user-interactive foreground application traffic.

Traffic in this class should be marked AF Class 1 (AF11/AF12/AF13 or DSCP 18/20/22, respectively) and should be provisioned with a moderate, but dedicated bandwidth queue with DSCP WRED enabled. This traffic class may be subject to policing and re-marking. Example applications include email, backup operations, FTP/SFTP transfers, video and content distribution, and so on.

Best Effort Data

The Best Effort Data service class is the default class.

Best effort data recommendations (per RFC 4594) include the following:

- Best effort data traffic should be marked to DF (DSCP 0).
- Best effort data should be provisioned with a dedicated queue.
- Best effort data may be provisioned with a guaranteed-bandwidth allocation and WRED/RED.

Because only few of the typically thousands of applications on today's business networks are assigned to preferential (or deferential) service classes, the vast majority of applications continue to default to this service class. So, this default class should be provisioned with a dedicated queue, which may include a minimum bandwidth guarantee.

Traffic in this class is marked Default Forwarding (DF or DSCP 0) and should be provisioned with a dedicated queue. To improve throughput and to avoid TCP global synchronization, WRED is recommended to be enabled on this class. Technically speaking, though, because all the traffic in this class is marked to the same "weight" (of DSCP 0), the congestion avoidance mechanism is essentially RED.

Scavenger (Lower-Priority Data)

This Scavenger service class (referred to as the Lower-Priority Data service class in RFC 4594) is intended for nonbusiness traffic flows, such as entertainment-oriented data or media applications.

Scavenger recommendations (per RFC 4594 and RFC 3662) include the following:

- Scavenger traffic should be marked to CS1 (DSCP 8).
- Scavenger traffic should be assigned a minimally provisioned queue.

The approach of offering a "less than Best Effort" service class for nonbusiness applications (as opposed to shutting these down entirely) has proven to be a popular political compromise between IT departments and their user base. In addition, utilizing a Scavenger class allows for IT departments to provide a degree of control for bottom-up applications that their user base may introduced onto their networks but which they are not prepared to fully support. So, Scavenger class applications are permitted on enterprise networks as long as resources are always available for top-down business-critical applications. However, as soon the network experiences congestion, this class is the most aggressively dropped.

Traffic in this class should be marked CS1 (DSCP 8) and should be provisioned with a minimal bandwidth queue that is the first to starve should network congestion occur. Example traffic includes YouTube, Xbox Live/360 movies, iTunes, BitTorrent, and so on.

Control Plane Traffic

Up until this point, only data plane applications have been considered. However, control plane traffic should not be overlooked. Compared to the data plane, control plane traffic is relatively minor in volume, but these flows are critical to the overall functioning of the network infrastructure and to voice and video endpoints.

Some Cisco products—both hardware and software—often include some mechanisms to provide a degree of automatic protection to some control plane flows, but network administrators should still give them due consideration and (as necessary) explicit provisioning to ensure proper functioning.

Control plane traffic can be subdivided into the following classes:

- Network Control
- Signaling
- Operations/Administration/Management (OAM)

The sections that follow consider each of these control plane classes in turn.

Network Control

The Network Control service class is intended for network control plane traffic, which is required for reliable operation of the network infrastructure.

Network control traffic recommendations (per RFC 4594) include the following:

- Network control traffic should be marked to CS6 (DSCP 48).
- Network control traffic may be assigned a moderately provisioned guaranteed-bandwidth queue.

Traffic in this class should be marked CS6 (DSCP 48) and may be provisioned with a moderate, but dedicated, guaranteed-bandwidth queue. WRED should not be enabled on this class, because network control traffic should not be dropped. (If this class is experiencing drops, the bandwidth allocated to it should be reprovisioned.) Example traffic includes Extended Interior Gateway Routing Protocol (EIGRP), Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), Hot Standby Router Protocol (HSRP), Internet Key Exchange (IKE), and so on.

Signaling

The Signaling service class is intended for call-signaling traffic that supports IP voice and video telephony; essentially, this traffic is control plane traffic for the voice and video telephony infrastructure.

Signaling traffic recommendations (per Cisco's interpretation of RFC 4594*) include the following:

- Signaling traffic should be marked to CS3 (DSCP 24).
- Signaling traffic may be assigned a moderately provisioned guaranteed-bandwidth queue.

Note As previously noted, Cisco has swapped the RFC 4594 marking recommendations for broadcast video with call signaling, such that Cisco has decided to mark broadcast video traffic as CS5 and signaling traffic as CS3.

Traffic in this class should be marked CS3 (DSCP 24) and may be provisioned with a moderate, but dedicated, guaranteed-bandwidth queue. WRED should not be enabled on this class, because signaling traffic should not be dropped. (If this class is experiencing drops, the bandwidth allocated to it should be reprovisioned.) Example traffic includes Skinny Call Control Protocol (SCCP), Session Initiation Protocol (SIP), H.323, and so on.

Operations/Administration/Management

The OAM service class is intended for (as the name implies) network operations, administration, and management traffic. This class is important to the ongoing maintenance and support of the network.

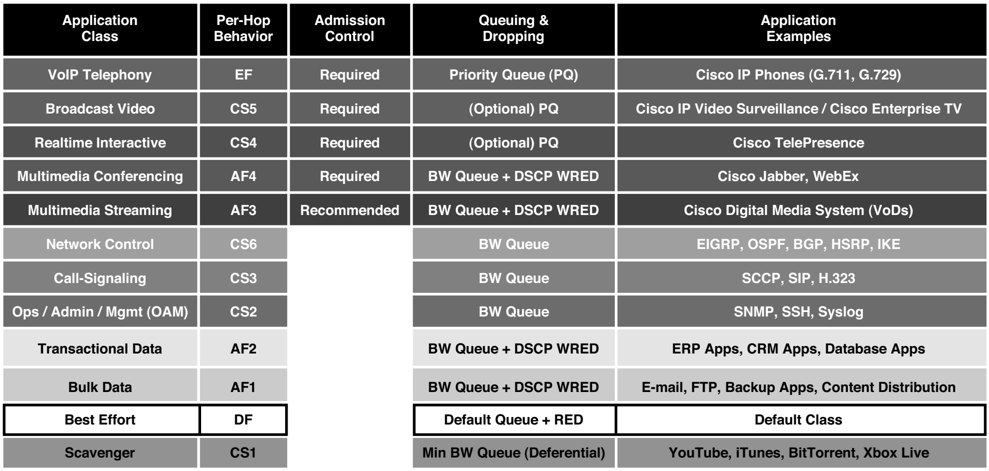
OAM traffic recommendations (per RFC 4594) include the following:

- OAM traffic should be marked to CS2 (DSCP 16).
- OAM traffic may be assigned a moderately provisioned guaranteed-bandwidth queue.

Traffic in this class should be marked CS2 (DSCP 16) and may be provisioned with a moderate, but dedicated, guaranteed-bandwidth queue. WRED should not be enabled on this class, because OAM traffic should not be dropped. (If this class is experiencing drops, the bandwidth allocated to it should be reprovisioned). Example traffic includes Secure Shell (SSH), Simple Network Management Protocol (SNMP), syslog, HTTP/HTTPS, and so on.

Cisco (RFC 4594-Based) QoS Recommendations by Application Class Summary

Figure 10-4 provides a summary chart of the application class recommendations discussed in this chapter.



Application Class	Per-Hop Behavior	Admission Control	Queuing & Dropping	Application Examples
VoIP Telephony	EF	Required	Priority Queue (PQ)	Cisco IP Phones (G.711, G.729)
Broadcast Video	CS5	Required	(Optional) PQ	Cisco IP Video Surveillance / Cisco Enterprise TV
Realtime Interactive	CS4	Required	(Optional) PQ	Cisco TelePresence
Multimedia Conferencing	AF4	Required	BW Queue + DSCP WRED	Cisco Jabber, WebEx
Multimedia Streaming	AF3	Recommended	BW Queue + DSCP WRED	Cisco Digital Media System (VoDs)
Network Control	CS6		BW Queue	EIGRP, OSPF, BGP, HSRP, IKE
Call-Signaling	CS3		BW Queue	SCCP, SIP, H.323
Ops / Admin / Mgmt (OAM)	CS2		BW Queue	SNMP, SSH, Syslog
Transactional Data	AF2		BW Queue + DSCP WRED	ERP Apps, CRM Apps, Database Apps
Bulk Data	AF1		BW Queue + DSCP WRED	E-mail, FTP, Backup Apps, Content Distribution
Best Effort	DF		Default Queue + RED	Default Class
Scavenger	CS1		Min BW Queue (Deferential)	YouTube, iTunes, BitTorrent, Xbox Live

Figure 10-4 *Cisco's RFC 4594-Based Application Class QoS Recommendations*

QoS Standards Evolution

Many IT managers and network administrators baulk when first presented with the RFC 4594-based 12-class model shown in Figure 10-4. So, keep in mind a couple of good points to prevent panic:

- RFC 4595 is not a standard, but rather resides within the "informational" category of RFCs. Therefore, conforming to it—in whole or in part—is the prerogative of the individual organization.
- Such a complex model may be implemented in a gradual and phased approach (as detailed in the following chapter).

Nonetheless, network managers and architects should begin considering such advanced QoS models; this is definitely the direction that this technology is being pushed in by evolving business and application demands and as reflected by various developments in QoS RFCs. Specifically, there are at least at least three recent RFC-based developments to show that QoS models are indeed evolving in complexity:

- RFC 2597, *Clarification*
- RFC 5865, *Proposed Standard*
- RFC 4594, *Update Draft*

The sections that follow examine each of these QoS RFC developments in turn.

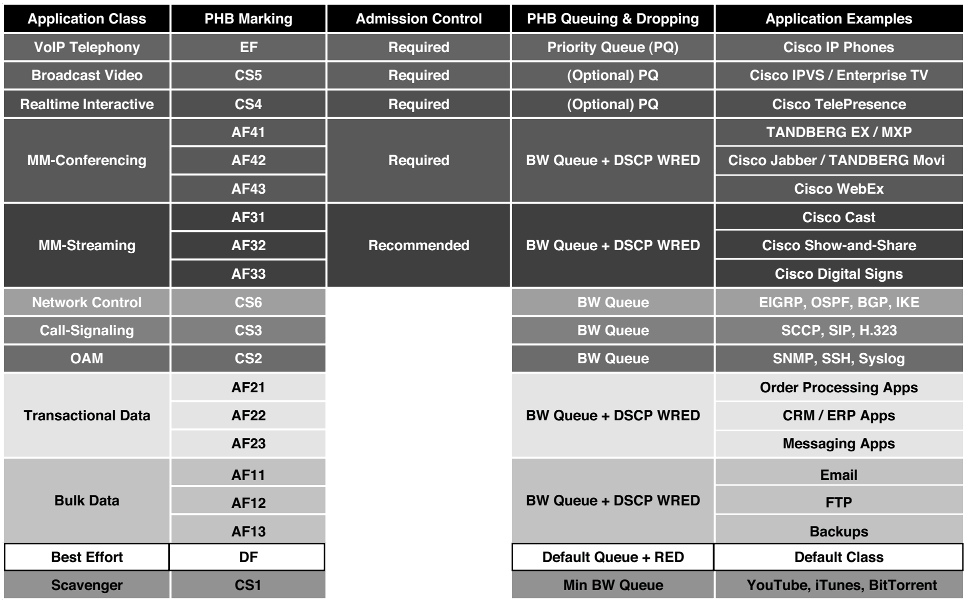
RFC 2597, *Clarification*

When RFC 2597 first defined the AF PHB, the industry-norm interpretation of it was that endpoint traffic would always be marked to the AF class's first drop precedence level; that is, endpoint traffic could only be marked to AF11, AF21, AF31, or AF41. Furthermore, it was generally understood that AF class traffic could *only* be marked to drop precedence level 2 or 3 by a policer.

However, shortly after Cisco acquired Tandberg in 2010, their product teams and IT departments faced a quandary as to how to differentiate various video-based traffic that Tandberg applications generated, as opposed to similar traffic that Cisco products already produced. One suggested approach was to increase the number of classes provisioned for video, but this was viewed as excessively complex. An alternative suggestion was to mark various multimedia-conferencing endpoints—not to AF41, but to AF42 or AF43 *directly on the endpoints* (and not by a midstream policer). Then these various video flows could be assigned to a single AF class queue with DSCP-based WRED providing interqueue QoS.

The Cisco teams hesitated to adopt this latter approach because it was viewed by some as a violation of the RFC 2597. However, a careful rereading of RFC 2597 (with the original authors) led to the clarification that nothing within it precluded endpoint marking to drop precedence 2 or 3.

This being the case, it became easily possible to expand the 12-class model shown in Figure 10-4 to 20 QoS classes—without adding additional queuing classes (simply by allowing endpoint marking to drop precedence 2 or 3 and enabling DSCP-based WRED on all AF class queues). Figure 10-5 illustrates an example of an expanded QoS model based on this clarification.



Application Class	PHB Marking	Admission Control	PHB Queuing & Dropping	Application Examples
VoIP Telephony	EF	Required	Priority Queue (PQ)	Cisco IP Phones
Broadcast Video	CS5	Required	(Optional) PQ	Cisco IPVS / Enterprise TV
Realtime Interactive	CS4	Required	(Optional) PQ	Cisco TelePresence
MM-Conferencing	AF41	Required	BW Queue + DSCP WRED	TANDBERG EX / MXP
	AF42			Cisco Jabber / TANDBERG Movi
	AF43			Cisco WebEx
MM-Streaming	AF31	Recommended	BW Queue + DSCP WRED	Cisco Cast
	AF32			Cisco Show-and-Share
	AF33			Cisco Digital Signs
Network Control	CS6		BW Queue	EIGRP, OSPF, BGP, IKE
Call-Signaling	CS3		BW Queue	SCCP, SIP, H.323
OAM	CS2		BW Queue	SNMP, SSH, Syslog
Transactional Data	AF21		BW Queue + DSCP WRED	Order Processing Apps
	AF22			CRM / ERP Apps
	AF23			Messaging Apps
Bulk Data	AF11		BW Queue + DSCP WRED	Email
	AF12			FTP
	AF13			Backups
Best Effort	DF		Default Queue + RED	Default Class
Scavenger	CS1		Min BW Queue	YouTube, iTunes, BitTorrent

Figure 10-5 *Expanded QoS Model Based on RFC 2597, Clarification*

RFC 5865, *Proposed Standard*

A second RFC-based development increasing QoS model complexity was the 2010 proposal for a standard on "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic" in the form of RFC 5865.

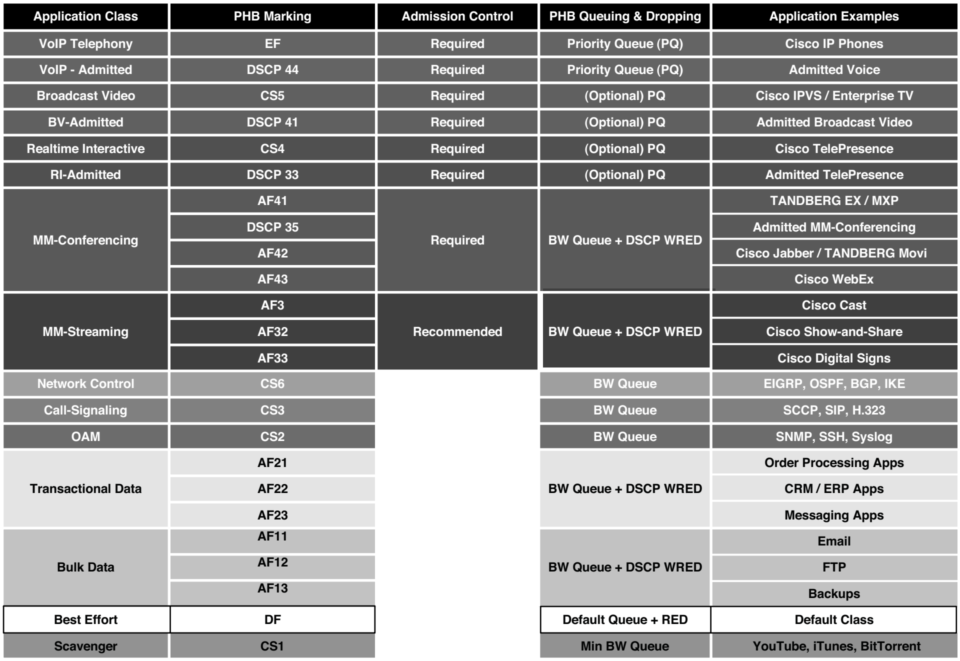
This RFC proposes the use of nonstandard code points to reflect an admission control decision. For example, it advocates the use of DSCP 44 to identify a voice flow that has been formally admitted onto the network (as opposed to the default voice marking which remains as DSCP 46/EF).

Furthermore, this RFC proposes that similar nonstandard code points can be used to reflect admission control decisions for each of the following video classes:

- Broadcast Video
- Real-Time Interactive
- Multimedia Conferencing

Note RFC 5865 makes the recommendation of a nonstandard code point to identify admission control decisions for these video classes, but it does not explicitly specify what these should be. (Only DSCP 44 for admitted voice traffic is explicitly specified in this RFC.)

This standard then has the potential ramification of adding four new QoS levels of service to previous models (one each for admitted voice, broadcast video, real-time interactive, and multimedia conferencing). Therefore, building on the previous QoS model, the number of overall application classes is being pushed as high as 24, as shown in Figure 10-6.



Application Class	PHB Marking	Admission Control	PHB Queuing & Dropping	Application Examples
VoIP Telephony	EF	Required	Priority Queue (PQ)	Cisco IP Phones
VoIP - Admitted	DSCP 44	Required	Priority Queue (PQ)	Admitted Voice
Broadcast Video	CS5	Required	(Optional) PQ	Cisco IPVS / Enterprise TV
BV-Admitted	DSCP 41	Required	(Optional) PQ	Admitted Broadcast Video
Realtime Interactive	CS4	Required	(Optional) PQ	Cisco TelePresence
RI-Admitted	DSCP 33	Required	(Optional) PQ	Admitted TelePresence
MM-Conferencing	AF41	Required	BW Queue + DSCP WRED	TANDBERG EX / MXP
	DSCP 35			Admitted MM-Conferencing
	AF42			Cisco Jabber / TANDBERG Movi
	AF43			Cisco WebEx
MM-Streaming	AF3	Recommended	BW Queue + DSCP WRED	Cisco Cast
	AF32			Cisco Show-and-Share
	AF33			Cisco Digital Signs
Network Control	CS6		BW Queue	EIGRP, OSPF, BGP, IKE
Call-Signaling	CS3		BW Queue	SCCP, SIP, H.323
OAM	CS2		BW Queue	SNMP, SSH, Syslog
Transactional Data	AF21		BW Queue + DSCP WRED	Order Processing Apps
	AF22			CRM / ERP Apps
	AF23			Messaging Apps
Bulk Data	AF11		BW Queue + DSCP WRED	Email
	AF12			FTP
	AF13			Backups
Best Effort	DF		Default Queue + RED	Default Class
Scavenger	CS1		Min BW Queue	YouTube, iTunes, BitTorrent

Figure 10-6 *Expanded QoS Model Based on RFC 2597 and RFC 5865*

RFC 4594, *Update Draft*

And finally, a third RFC-based development pushing QoS model complexity higher is a draft proposal to update RFC 4594. This draft (in its current form at the time of this writing) proposed 14 traffic classes using 28 different code point marking values, as shown in Table 10-1.

Table 10-1 *RFC 4594, Draft Proposal (as of July 2012)*

Service Class Name	PHB / Name	DSCP	Application Examples
Network Control	CS7	56	Network control plane protocols
	CS6	48	Internetwork routing & control plane protocols
Audio	EF	46	Voice bearer
	Voice-Admit	44	Admitted-Voice
Realtime-Interactive	CS5	40	Remote/virtual desktop
	CS5-Admit	41	Admitted-Realtime-Interactive
High-Definition Audio/Video	CS4	32	Conversational high-definition audio/video
	CS4-Admit	33	Admitted-High-Definition Audio/Video
Video	AF41, AF42, AF43	34, 36, 38	Audio/video conferencing bearer
Multimedia Conferencing	MC	29	Presentation data and application sharing
	MC-Admit	37	Admitted-Multimedia Conferencing
Multimedia Streaming	AF31, AF32, AF33	011000	Streaming audio and video on-demand
Broadcast	CS3	24	Broadcast TV, live events, video surveillance
	CS3-Admit	25	Admitted-Broadcast
Low-Latency Data	AF21, AF22, AF23	18, 20, 22	Client/server, web-based ordering, messaging
Conversational Signaling	A/V-Sig	17	Conversational signaling
OAM	CS2	16	Operations / Administration / Management
High-Throughput Data	AF11, AF12, AF13	10, 12, 14	Store-and-Forward applications
Low-Priority Data	CS1	8	Scavenger applications
Best Effort	CS0	0	Undifferentiated applications (default class)

Whether this draft proposal will replace RFC 4594—or if it does, what will be the final form of this proposal—remains to be seen.

However, what is apparent is that the QoS standards bodies are cognizant of developing business trends and evolving application requirements of QoS and are thinking as far ahead of the curve as possible so as to accommodate these needs in future QoS models.

Long story short: Expect QoS models to continue to evolve in complexity.

Summary

Successful QoS deployments are driven by first and foremost by business/organizational objectives. Therefore, this unique chapter broke away from the traditional technical discussion of this book and examined business trends in networking that impact QoS designs.

Global networking trends considered included the Internet's average annual traffic volume growth rate of nearly 30 percent, the increase in traffic from wireless and non-PC devices, and the emerging dominance of video-based traffic by volume.

Application trends were also considered, such as the evolution of video applications and the impact that reduced barriers to video production, distribution, and consumption has had on media explosion. Also discussed were the phenomena of social networking and the emergence of bottom-up applications and devices within the workplace. QoS design impacts of multimedia applications and high-definition media were also reviewed.

Following this, the QoS requirements and RFC 4594-based recommendations of 12 application classes were overviewed: Voice, Broadcast Video, Real-Time Interactive, Multimedia Conferencing, Multimedia Streaming, Transactional Data, Bulk Data, Best Effort, Scavenger, Network Control, Signaling, and OAM.

Finally, three key RFC developments were discussed (namely RFC 2597's clarification, RFC 5865's proposal, and RFC 4594's proposed update), which all show that the technology direction of QoS is continuing to evolve in complexity. Therefore, network administrators need to think ahead to accommodate not only their current but also their future QoS needs.

Further Reading

Cisco Visual Networking Index: Forecast and Methodology, 2011-2016: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

Cisco Medianet QoS Design Strategy—At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qosmrn.pdf

Cisco Enterprise Medianet Quality of Service Design 4.0—Overview: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html

RFC 4594, Configuration Guidelines for DiffServ Service Classes: http://www.ietf.org/rfc/rfc4594

RFC 5865, A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic: http://tools.ietf.org/html/rfc5865

RFC 4594, Update Draft: Standard Configuration of DiffServ Service Classes: http://datatracker.ietf.org/doc/draft-polk-tsvwg-rfc4594-update/

Chapter 11

QoS Design Principles and Strategies

Having reviewed the QoS toolset, and business and application requirements, you are now ready to pull all these together to form a cohesive end-to-end QoS strategy for your organization, which is the goal of this chapter.

Because there is no one-size-fits-all solution to individual business requirements, this chapter presents three (increasingly complex) levels of QoS model strategies:

- **Basic:** A 4-class QoS model strategy
- **Intermediate:** An 8-class QoS model strategy
- **Advanced:** A RFC 4594-based 12-class QoS model strategy

Also, some additional strategies are presented to address application class expansion and using QoS to improve the overall security of the network infrastructure.

Keep in mind that there is usually more than one solution to any given QoS challenge, especially with a rich QoS toolset at your disposal. To that end, the review of a few succinct best-practice design principles can help simplify and accelerate the strategic design process. The section that follows covers such QoS best-practice design principles.

QoS Best-Practice Design Principles

There is usually more than one tool to solve a given problem. Some tools can be crude yet effective; others can prove elegant and efficient. For example, a person could use a rock to drive a nail into a wall, but a carpenter (or most other sane people, for that matter) would probably prefer a hammer.

This principle holds true with QoS tools, too. For example, Part I, “QoS Design Overview,” presented an overview of various tools that could perform marking, including markers and policers. Both can achieve the result of marking a packet, yet they do so differently, and therefore these tools are best used in different design contexts. Therefore, to match the right QoS tool to the right QoS challenge, it is helpful to review a few QoS design best practices.

Hardware Versus Software QoS Best Practices

Design principle:

- Always enable QoS policies in hardware—rather than software—whenever a choice exists.

A fundamental QoS design principle is to always enable QoS policies in hardware—rather than software—whenever a choice exists.

Some Cisco routers (such as Cisco Integrated Services Routers [ISR]) perform QoS in software, which places incremental loads on the CPU. The actual incremental load will depend on the numerous factors, including the complexity and functionality of the policy, the volume and composition of the traffic, the speed of the interface, the speed of the CPU, the memory of the router, and so on.

In contrast, other devices (such as Cisco Catalyst switches) perform QoS in dedicated hardware application-specific integrated circuits (ASICs). Therefore, these switches can perform even the most complex QoS policy on maximum traffic loads at line rates on GE/10GE interfaces—all without any marginal CPU tax.

A final case exists, as some newer platforms (such as the Cisco Aggregation Services Routers [ASR]) perform QoS in a combination of hardware and software. That is to say, certain QoS functions (such as queuing) may be performed in dedicated hardware ASICs, and other functions (such as deep packet inspection) are still processed in software via the CPU.

Therefore, whenever a choice exists, Cisco recommends implementing QoS policies in devices that perform QoS operations in hardware—rather than software—because this will result in more efficient utilization of network infrastructure resources.

For example, suppose an administrator has the option of deploying classification and marking policies in a branch network in either a Catalyst switch that the endpoints directly connect to (in hardware) or at the LAN-edge interface of an ISR router that the switch connects to (in software). Because a choice exists as to where the policy should be designed, it would be more efficient to classify and mark within the Catalyst switch. There may be cases, however, where such a choice doesn't exist. Continuing the example: There might be a business need to perform NBAR deep packet inspection (DPI) on branch-originated traffic (which is not currently supported on Catalyst switches), and therefore the administrator would then have to apply the required classification and marking policies on the ISR router.

Classification and Marking Best Practices

Design principles:

- Classify and mark applications as close to their sources as technically and administratively feasible.
- Use DSCP marking whenever possible.
- Follow standards-based differentiated services code point (DSCP) per-hop behavior (PHB) markings to ensure interoperability and future expansion.

When classifying and marking traffic, a recommended design principle is to classify and mark applications as close to their sources as technically and administratively feasible. This principle promotes end-to-end differentiated services (DiffServ) and PHBs.

In general, it is not recommended to trust markings that can be set by users on their PCs or other similar devices because users can easily abuse provisioned QoS policies if permitted to mark their own traffic. For example, if an Expedited Forwarding (EF) PHB has been provisioned over the network, a PC user can easily configure all his traffic to be marked to EF, thus hijacking network priority queues to service his non-real-time traffic. Such abuse could easily ruin the service quality of real-time applications throughout the enterprise. However, if enterprise controls are in place to centrally administer PC QoS markings, it may be an acceptable design option to trust them.

Note Although it is generally recommended not to trust QoS markings from PCs and endpoint devices, this might not hold true on all design scenarios. For example, in WLANs (which leverage a shared media that all endpoints contend for), it is better to trust Layer 2 markings from endpoint devices so that these can leverage Wireless Multimedia (WMM) QoS provisioning over the air in the upstream direction. Further details on this are covered in Part IV, "Wireless Campus QoS Design," of this book. The key point here is that this general recommendation of not to trust endpoint devices does not universally apply.

Following this general rule, it is further recommended to use DSCP markings whenever possible, because these Layer 3 IP header markings are end to end, more granular, and more extensible than Layer 2 markings. Remember, Layer 2 markings are lost when media changes (such as a LAN-to-WAN/VPN edge). There is also less marking granularity at Layer 2. For example, 802.1Q/p class of service (CoS) supports only 3 bits (values 0–7), as does MPLS EXP. Therefore, only up to eight classes of traffic can be supported at Layer 2 and interclass relative priority (such as RFC 2597, *Assured Forwarding Drop Preference Markdown*) is not supported. Layer 3 DSCP markings allow for up to 64 classes of traffic.

As the line between enterprises and service providers continues to blur and because the need for interoperability and complementary QoS markings is critical, you should follow standards-based DSCP PHB markings to ensure interoperability and future expansion.

Because the enterprise Medianet marking recommendations are standards based—as previously discussed—enterprises can easily adopt these markings to interface with service provider classes of service. Network mergers—whether the result of acquisitions, mergers, or strategic alliances—are also easier to manage when using standards-based DSCP markings.

Policing and Markdown Best Practices

Design principles:

- Police traffic flows as close to their source as possible.
- Whenever possible, mark down according to standards-based rules.

There is little reason to forward unwanted traffic only to police and drop it at a downstream node, especially when the unwanted traffic is the result of DoS or worm attacks. Furthermore, the overwhelming volume of traffic that such attacks can create can cause network outages by driving network device processors to their maximum levels. Therefore, it is recommended to police traffic flows as close to their sources as possible. This principle applies also to legitimate flows, because worm-generated traffic can masquerade under legitimate and well-known TCP/UDP ports and cause extreme amounts of traffic to be poured onto the network infrastructure. Such excesses should be monitored at the source and marked down appropriately.

Whenever supported, markdown should be done according to standards-based rules, such as RFC 2597, *The AF PHB Definition*. For example, excess traffic marked to AFx1 should be marked down to AFx2 (or AFx3 whenever dual-rate policing—such as defined in RFC 2698—is supported). Following such markdowns, congestion management policies, such as DSCP-based weighted random early detection (WRED), should be configured to drop AFx3 more aggressively than AFx2, which in turn should be dropped more aggressively than AFx1.

Queuing and Dropping Best Practices

Design principles:

- Enable queuing policies at every node that has the potential for congestion.
- Whenever possible, assign each application class to its own dedicated queue.
- Use only platforms/service providers that offer a minimum of four standards-based queuing behaviors:
    - An RFC 3246 Expedited Forwarding PHB
    - An RFC 2597 Assured Forwarding PHB
    - An RFC 2474 Default Forwarding PHB
    - An RFC 3662 Lower Effort per-domain behavior

Business-critical applications require service guarantees regardless of network conditions. The only way to provide service guarantees is to enable queuing at any and every node that has the potential for congestion; regardless of how rarely this, in fact, may occur. This principle applies not only to campus-to-WAN/VPN edges, where speed mismatches are most pronounced, but also to campus interswitch links, where oversubscription ratios create the potential for instantaneous congestion and buffer overruns. There is simply no other way to guarantee service levels than by enabling queuing wherever a speed mismatch exists.

In addition, because each application class has unique service level requirements, each should optimally be assigned a dedicated queue. In such a manner, specific bandwidth allocations and dropping policies can be assigned to each discrete application class to meet its distinctive QoS requirements. Otherwise, if multiple application classes are assigned into a common queuing bucket, the administrator no longer can control if bandwidth resources are being shared among these application classes according to their individual requirements.

At a minimum, however, the following standards-based queuing behaviors should be supported on all platforms (or service provider links) for businesses deploying QoS for rich-media applications:

- Real-time queue (to support an RFC 3246 EF PHB service)
- Guaranteed-bandwidth queue (to support RFC 2597 AF PHB services)
- Default queue (to support an RFC 2474 DF service)
- Bandwidth-constrained queue (to support an RFC 3662 scavenger service)

Additional recommendations for each of these queuing services are discussed next.

EF Queue Recommendations: The 33% LLQ Rule

Design principles:

- Limit the amount of strict-priority queuing to 33 percent of link bandwidth capacity.
- Govern strict-priority traffic with an admission control mechanism.
- Do not enable WRED on this queue.

The real-time or strict-priority queue corresponds to the RFC 3246 EF PHB. The amount of bandwidth assigned to the real-time queue is usually variable. However, if the majority of bandwidth is provisioned with strict-priority queuing (which is effectively a first-in, first-out [FIFO] queue), the overall effect is a dampening of QoS functionality, both jitter-sensitive real-time applications (contending with each other within the FIFO priority queue) and also for non-real-time applications (because these might periodically receive wild bandwidth allocation fluctuations, depending on the instantaneous amount of traffic being serviced by the priority queue). Remember the goal of convergence is to enable voice, video, and data applications to transparently coexist on a single network. When real-time applications dominate a link, non-real-time applications fluctuate significantly in their response times, destroying the transparency of the converged network.

For example, consider a (45-Mbps) DS3 link configured to support two TelePresence (CTS-3000) calls with an EF PHB service. Assuming that both systems are configured to support the highest level of resolution (full HD-1080p) with all optional components enabled, each such call requires about 20 Mbps of strict-priority queuing. Before these TelePresence calls being placed, non-real-time applications have access to 100 percent of the bandwidth on the link. (To simplify the example, assume that there are no other real-time applications on this link.) However, once these TelePresence calls are established, all non-real-time applications would suddenly be contending for about 10 percent of the link. TCP windowing would take effect and many applications hang, time out, or become stuck in a nonresponsive state, which usually translates into users calling the IT help desk complaining about the network (which happens to be functioning properly, albeit in a poorly configured manner).

To obviate such scenarios, Cisco Technical Marketing has done extensive testing and has found that a significant decrease in non-real-time application response times occurs when real-time traffic exceeds one-third of link bandwidth capacity. In fact, both testing and customer deployments have shown that a general best queuing practice is to limit the amount of strict-priority queuing to 33 percent of link bandwidth capacity. This strict-priority queuing rule is a conservative and safe design ratio for merging real-time applications with data applications.

Note As previously discussed, Cisco IOS Software allows the abstraction (and thus configuration) of multiple strict-priority low-latency queues (LLQs). In such a multiple-LLQ context, this design principle would apply to the sum of all LLQs to be within one-third of link capacity.

It is vitally important to understand that this strict-priority queuing rule is simply a best practice design recommendation and is not a mandate. There might be cases where specific business objectives cannot be met while holding to this recommendation. In such cases, enterprises must provision according to their detailed requirements and constraints. However, it is important to recognize the tradeoffs involved with overprovisioning strict-priority traffic and its negative performance impact both on other real-time flows and also on non-real-time application response times.

Also, any traffic assigned to a strict-priority queue should be governed by an admission control mechanism to prevent these from starving bandwidth from non-real-time applications.

Finally, WRED—or any similar congestion avoidance mechanism—should never be enabled on the strict-priority queue. Traffic assigned to this queue is often highly drop-sensitive (for example, voice and especially HD video); therefore, early dropping should never be induced on these flows. If you notice drops on this queue (which would indicate tail—not early—drops), you should reprovision the queue's size (and admission control mechanisms) to ensure that dropping does not continue to occur.

AF Queue Recommendations

Design principles:

- Provision guaranteed-bandwidth allocations according to application requirements.
- Enable DSCP-based WRED on these queues.

At least one queue should be provisioned with an Assured Forwarding PHB. Per the standard, up to four queues can be provisioned with this service: one each for AF Class 1, AF Class 2, AF Class 3, and AF Class 4. These queues should have a bandwidth guarantee that corresponds with the application class requirements of the traffic assigned to it.

In addition, DSCP-based WRED should be enabled on these queues, such that traffic marked AFx3 is (statistically) dropped sooner and more often than AFx2, which in turn is (statistically) dropped more aggressively than AFx1.

DF Queue Recommendations

Design principles:

- Provision at least 25 percent of link bandwidth for the default Best Effort class.
- Enable WRED (effectively RED) on the default class.

The Best Effort class is the default class for all traffic that has not been explicitly assigned to another application class queue. Only if an application has been selected for preferential/deferential treatment is it removed from the default class. Because most enterprises have several thousand applications running over their networks, adequate bandwidth must be provisioned for this class as a whole to handle the sheer number and volume of applications that default to it. Therefore, Cisco recommends provisioning at least 25 percent of link bandwidth for the default Best Effort class.

In addition, WRED is recommended to be enabled on the default class to improve throughput and reduce TCP synchronization. Because all traffic destined to this class is to be marked to the same DSCP value (of 0), there is no "weight" component to the WRED dropping decision, and therefore the congestion algorithm is effectively random early detect (RED).

Scavenger Class Queue Recommendations

Design principles:

- Assign minimum bandwidth to the Scavenger class queue.
- WRED is not required on the Scavenger class queue.

Whenever the Scavenger queuing class is enabled, it should be assigned a minimal amount of bandwidth, such as 1 percent (or whatever the minimal bandwidth allocation that the platform supports).

Note On some platforms, queuing distinctions between bulk data and scavenger traffic flows cannot be made, either because queuing assignments are determined by class of service (CoS) values (and both of these application classes share the same CoS value of 1) or because only a limited amount of hardware queues exist, precluding the use of separate dedicated queues for each of these two classes. In such cases, the scavenger/bulk queue can be assigned moderate amount of bandwidth, such as 5 percent.

WRED is not required on the Scavenger class queue because traffic assigned to this queue has no implied “good-faith” service guarantee or expectation. WRED would only be active in the event of congestion anyway, and at that point this queue is already being very aggressively dropped. Therefore, there is little to gain by adding this feature. Finally, if enabled on a router performing QoS in software, enabling WRED on this class would actually prove wasteful because it will consume additional—and unnecessary—CPU resources.

Figure 11-1 summarizes these minimum queuing capabilities and recommendations.

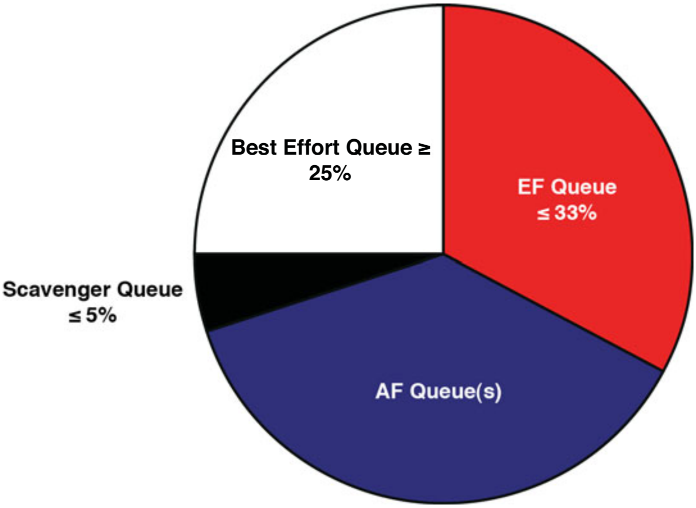


Figure 11-1 *Minimum Queuing Capabilities and Recommendations*

WRED Recommendations

Design principles:

- Enable DSCP-based WRED on AF queues.
- Enable WRED on the DF queue.
- Do not enable DSCP-based WRED on the EF queue.
- Do not enable WRED on the scavenger queue.
- Do not enable WRED on control traffic application class queues.
- Optional: Tune WRED thresholds consistently.

As already discussed, it is recommended to enable DSCP-based WRED on all AF queues (as per the AF PHB standard) and to enable WRED on the default best-effort queue (to improve application throughput).

In addition, as previously noted, it is not recommended to enable WRED on the EF queue (because traffic in this queue is usually drop sensitive), nor on the scavenger queue (because this is unnecessary and potentially wasteful).

Similarly, it is not recommended to enable WRED on any queue dedicated to servicing control traffic, such as routing, signaling, or management. Such traffic is vital to the proper operation of the infrastructure and therefore should never be early dropped (if at all). If you notice drops occurring on queues assigned to control traffic classes, these queues should be reprovisioned to accommodate.

Finally, on some platforms, the default WRED thresholds are inconsistent with others or do not fully use the queue depth. Therefore, these can be explicitly tuned to be consistent and compatible in the following manner:

- Set the minimum WRED thresholds for AFx3 to 60 percent of the queue depth.
- Set the minimum WRED thresholds for AFx2 to 70 percent of the queue depth.
- Set the minimum WRED thresholds for AFx1 to 80 percent of the queue depth.
- Set all WRED maximum thresholds to 100 percent.

Figure 11-2 shows these optional WRED tuning recommendations.

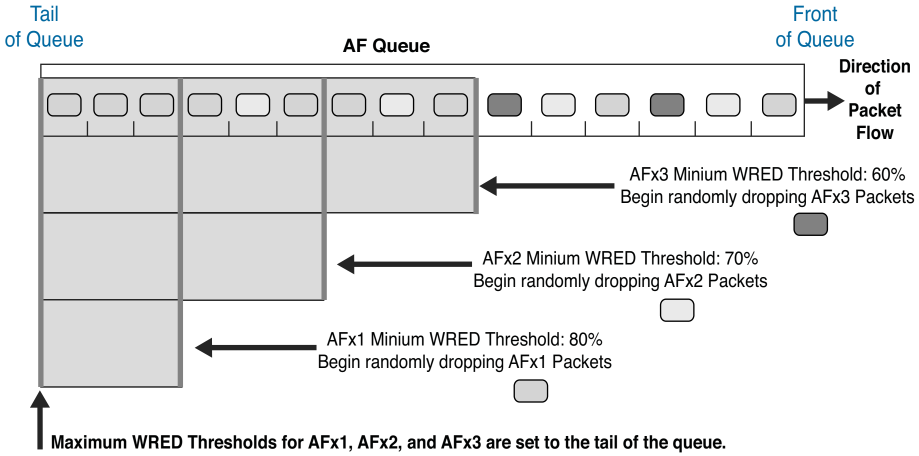

Tail of Queue
AF Queue
Front of Queue
Direction of Packet Flow
AFx3 Minium WRED Threshold: 60%
Begin randomly dropping AFx3 Packets
AFx2 Minium WRED Threshold: 70%
Begin randomly dropping AFx2 Packets
AFx1 Minium WRED Threshold: 80%
Begin randomly dropping AFx1 Packets
Maximum WRED Thresholds for AFx1, AFx2, and AFx3 are set to the tail of the queue.

Figure 11-2 *Optional WRED Tuning Recommendations*

QoS Design Strategies

Having reviewed these best-practice QoS design principles, it is now time to assemble them into an end-to-end strategy for a given business or organization. Obviously, there's never going to be a one-size-fits-all solution, because business objectives and constraints vary. Therefore, three generic models are presented as examples:

- 4-class model QoS strategy
- 8-class model QoS strategy
- 12-class model QoS strategy

In addition to these models, a strategy on application class expansion is presented as are strategies for using QoS to improve the overall security of the network infrastructure. The sections that follow discuss each of these strategies in turn.

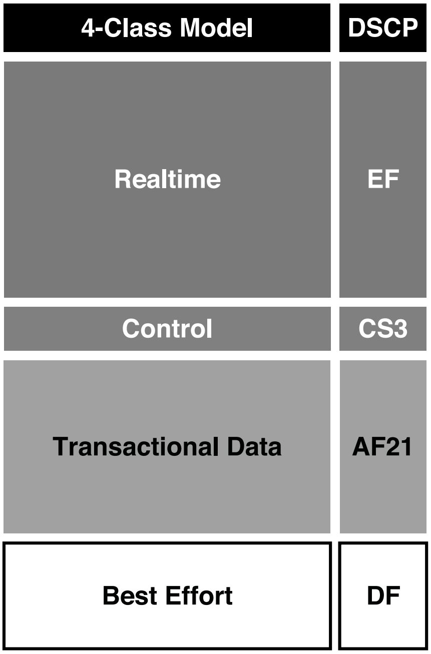
Four-Class Model QoS Strategy

The four-class QoS model represents a basic end-to-end QoS strategy. When businesses began deploying IP telephony, they required a three-class model (one each for voice, signaling, and best-effort/default). The four-class model adds only one additional class to these minimum requirements for IP telephony, as decided by the business objectives of QoS: The additional class may be an AF class for transactional data applications (as shown in the following example), or it may be an AF class for video/multimedia traffic, or it may even be a Scavenger class.

In this four-class QoS model example, the classes, markings, and treatments are as follows:

- **Voice:** Marked EF and treated with an EF PHB, but limited to 33 percent of strict-priority bandwidth.
- **Control:** Marked CS3 and provisioned with a guaranteed-bandwidth allocation of 7 percent. Although this class is primarily intended to service signaling traffic, other control traffic may also be serviced within this class (such as OAM).
- **Transactional data:** Marked AF21 and treated with an AF PHB, provisioned with guaranteed-bandwidth allocation of 35 percent, with DSCP-based WRED enabled.
- **Best-effort:** Marked DF and provisioned with a guaranteed-bandwidth allocation of 25 percent, with WRED-enabled.

Figure 11-3 shows the class structure and marking scheme for this four-class QoS model.



4-Class Model	DSCP
Realtime	EF
Control	CS3
Transactional Data	AF21
Best Effort	DF

Figure 11-3 *Four-Class QoS Model Class Structure and Marking Scheme Example*

Figure 11-4 shows the queuing strategy for this four-class QoS model.

This four-class QoS model is the basis for all corresponding four-class QoS model configurations that are presented throughout this book.

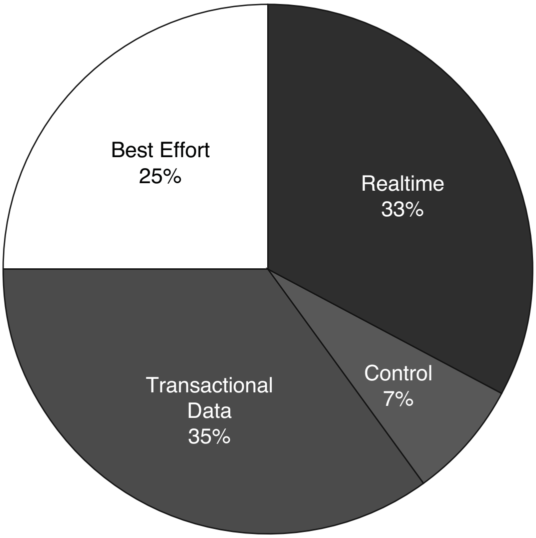


Figure 11-4 *Four-Class QoS Model Queuing Structure Example*

Eight-Class Model QoS Strategy

The eight-class QoS model builds on the previous four class model and includes two classes for multimedia traffic:

- Multimedia Conferencing
- Multimedia Streaming

Also new to the eight-class model is an explicit Network Control class (intended for routing protocols and other network infrastructure control traffic), and a less-than-best-effort Scavenger class.

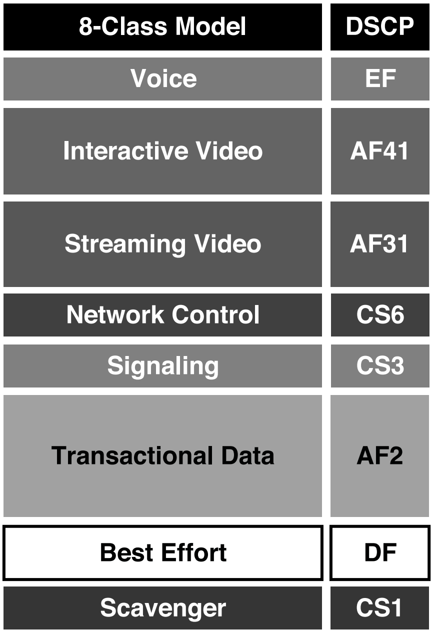
In this eight-class QoS model example, the classes, markings, and treatments are as follows:

- **Voice:** Marked EF and treated with an EF PHB, but limited to 10 percent strict-priority bandwidth.
- **Multimedia-conferencing:** Marked AF41 and *may be* treated with an EF PHB (borrowing from the Real-Time Interactive class provision within RFC 4594, as the Real-Time Interactive class is not being explicitly used in this model), but limited to 23 percent of strict-priority bandwidth.

  The strict-priority bandwidth assigned to both the Voice and Multimedia Conferencing class falls within the 33 percent EF queue guideline limit.
- **Multimedia-streaming:** Marked AF31 and treated with an AF PHB, provisioned with a guaranteed-bandwidth allocation of 10 percent, with DSCP-based WRED enabled.

- **Network control:** Marked CS6 and provisioned with an explicit guaranteed-bandwidth allocation of 5 percent; although this class is primarily intended to service routing traffic, other network control traffic may also be serviced within this class (such as OAM).
- **Signaling:** Marked CS3 and provisioned with a guaranteed-bandwidth allocation of 2 percent.
- **Transactional data:** Marked AF21 and treated with an AF PHB, provisioned with guaranteed-bandwidth allocation of 24 percent, with DSCP-based WRED enabled.
- **Best-effort:** Marked DF and provisioned with a guaranteed-bandwidth allocation of 25 percent, with WRED-enabled.
- **Scavenger:** Marked CS1 and treated with a bandwidth-constrained queue provisioned at 1 percent.

Figure 11-5 shows the class structure and marking scheme for this eight-class QoS model.



8-Class Model	DSCP
Voice	EF
Interactive Video	AF41
Streaming Video	AF31
Network Control	CS6
Signaling	CS3
Transactional Data	AF2
Best Effort	DF
Scavenger	CS1

Figure 11-5 *Eight-Class QoS Model Class Structure and Marking Scheme Example*

Figure 11-6 shows the queuing strategy for this eight-class QoS model.

This eight-class QoS model is the basis for all corresponding eight-class QoS model configurations that are presented throughout this book.

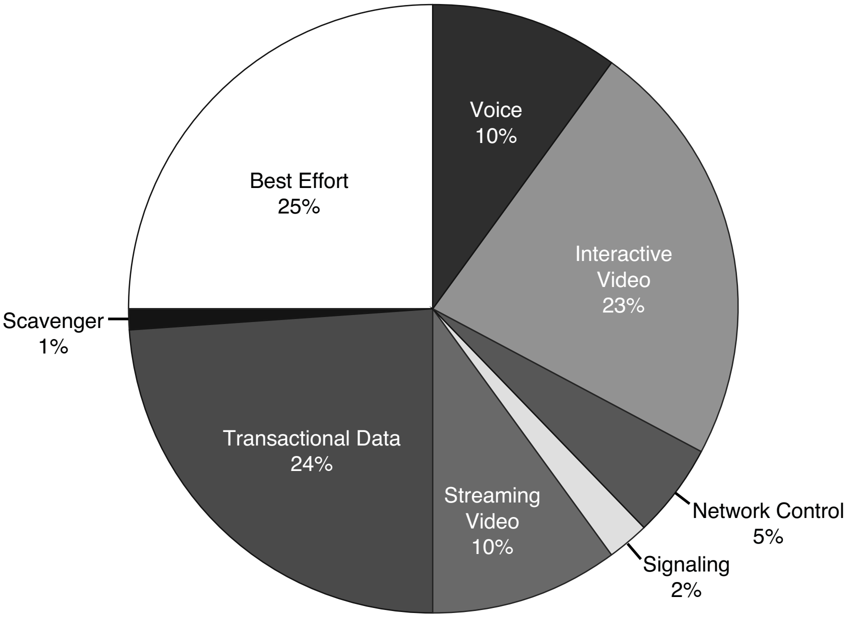


Figure 11-6 *Eight-Class QoS Model Queuing Structure Example*

Twelve-Class Model QoS Strategy

The 12-class model represents Cisco's interpretation of RFC 4594 (with the only change from the RFC's recommendations being the swapping of markings used for signaling with broadcast video, as previously discussed).

The 12-class QoS model builds on the previous 8-class model and includes 2 additional classes for inelastic multimedia traffic: Real-Time Interactive and Broadcast Video. Also new to the 12-class model is an explicit OAM class and a bandwidth-constrained class for Bulk Data.

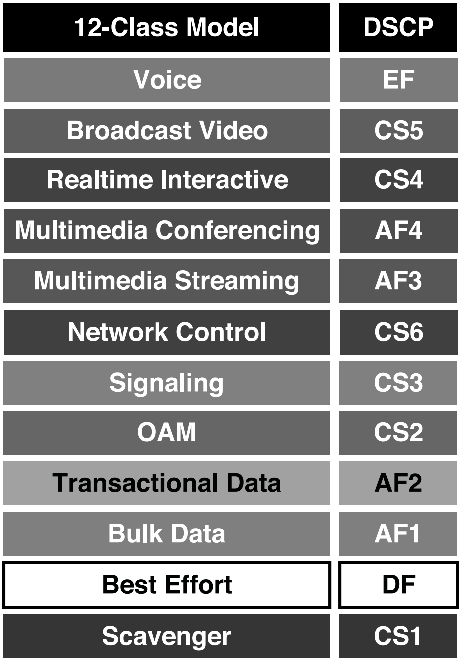
In this 12-class QoS model example, the classes, markings, and treatments are as follows:

- **Voice:** Marked EF and treated with an EF PHB, but limited to 10 percent strict-priority bandwidth.
- **Broadcast video:** Marked CS5 and *may be* treated with an EF PHB, but limited to 10 percent strict-priority bandwidth.
- **Real-time interactive:** Marked CS4 and *may be* treated with an EF PHB, but limited to 13 percent strict-priority bandwidth.

  Therefore the strict-priority bandwidth assigned to the Voice, Broadcast Video, and Real-Time Interactive class falls within the 33 percent EF queue guideline limit.
- **Multimedia conferencing:** Marked AF41 and treated with an AF PHB, provisioned with a guaranteed-bandwidth allocation of 10 percent, with DSCP-based WRED enabled

- **Multimedia streaming:** Marked AF31 and treated with an AF PHB, provisioned with a guaranteed-bandwidth allocation of 10 percent, with DSCP-based WRED enabled.
- **Network control:** Marked CS6 and provisioned with an explicit guaranteed-bandwidth allocation of 5 percent.
- **Signaling:** Marked CS3 and provisioned with a guaranteed-bandwidth allocation of 2 percent.
- **Management/OAM:** Marked CS2 and provisioned with a guaranteed-bandwidth allocation of 3 percent.
- **Transactional data:** Marked AF21 and treated with an AF PHB, provisioned with guaranteed-bandwidth allocation of 10 percent, with DSCP-based WRED enabled.
- **Bulk data:** Marked AF11 and treated with an AF PHB, provisioned with guaranteed-bandwidth allocation of 4 percent, with DSCP-based WRED enabled.
- **Best effort:** Marked DF and provisioned with a guaranteed-bandwidth allocation of 25 percent, with WRED-enabled.
- **Scavenger:** Marked CS1 and treated with a bandwidth-constrained queue provisioned at 1 percent.

Figure 11-7 shows the class structure and marking scheme for this 12-class QoS model.



12-Class Model	DSCP
Voice	EF
Broadcast Video	CS5
Realtime Interactive	CS4
Multimedia Conferencing	AF4
Multimedia Streaming	AF3
Network Control	CS6
Signaling	CS3
OAM	CS2
Transactional Data	AF2
Bulk Data	AF1
Best Effort	DF
Scavenger	CS1

Figure 11-7 *Twelve-Class QoS Model Class Structure and Marking Scheme Example*

Figure 11-8 shows the queuing strategy for this 12-class QoS model.

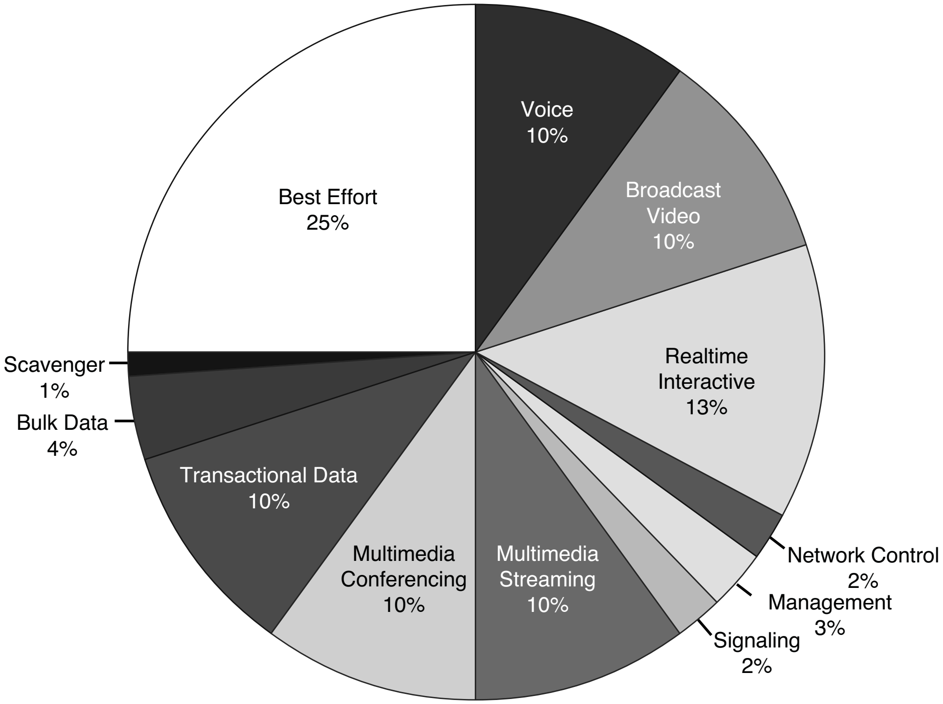


Figure 11-8 *Twelve-Class QoS Model Queuing Structure Example*

This 12-class QoS model is the basis for all corresponding 12-class QoS model configurations that will be presented throughout this book.

Application Class Expansion QoS Strategies

As business objectives and application-requirements continue to evolve, likewise will an organization's QoS strategy. Although many businesses may be reluctant to deploy a complex QoS model (such as the 12-class model), they should at least consider that down the road they may need to meet evolving business demands.

Some points that merit such forethought include what marking values will need to change and when and how classes (and their respective bandwidth allocations) will be subdivided.

Cisco recommends the ongoing monitoring of application traffic volumes and service levels and the corresponding reprovisioning of these in a gradual manner, as required. This approach helps to maintain consistency of user expectations and experience and serves to relieve administrative burdens that drastic reengineering of QoS policies places on network operations staff. For example, Figure 11-9 illustrates a gradual, phased approach of migration from a basic 4-class QoS model through to a 12-class QoS model.

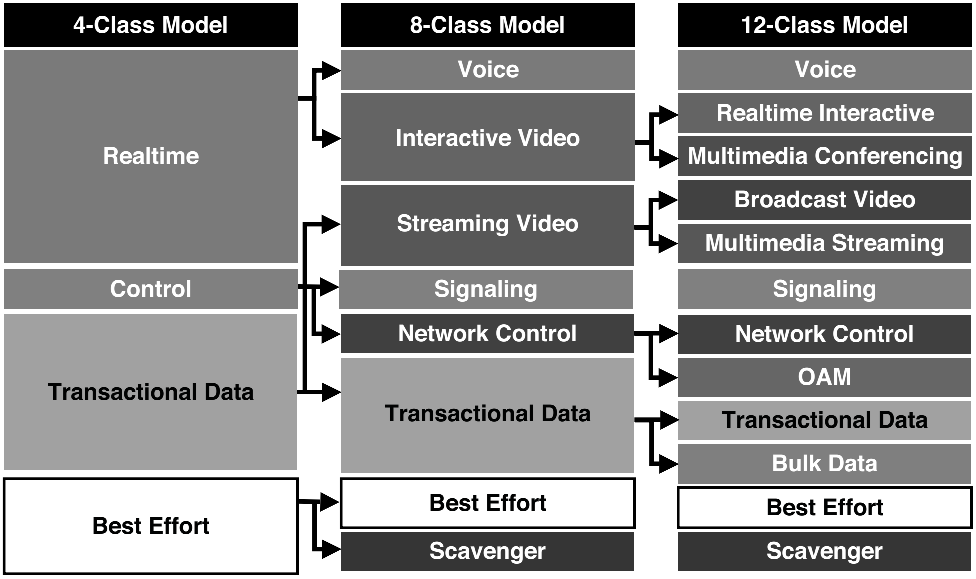


Figure 11-9 *Application-Class Expansion Strategy Example*

A tactic that network administrators can use to determine when a split in application classes would be needed would be to introduce multiple markings within a service class to track discrete application types within that class.

For example, consider a business that is using the eight-class model previously described. In this model, the Multimedia Conferencing class is marked to AF41. Now assume that this class was originally intended solely for desktop-based conferencing applications, but of late, several high-definition room-based systems (such as TelePresence units) have been deployed as a trial. To accommodate and track service levels and volumes that these TelePresence systems require, the network administrators may elect to mark these as CS4 and have *both* AF41 and CS4 traffic assigned to the single queuing class for multimedia-conferencing traffic. Then they can monitor the volume and quality of these TelePresence flows and, when appropriate, split this class into two: one each for real-time interactive (for TelePresence) and another for multimedia conferencing (for desktop conferencing applications).

When it comes time to split application classes, network administrators should maintain as much consistency and backward compatibility in bandwidth allocations as possible with previous models. In this manner, user expectations and experiences are maintained. Figure 11-10 shows how bandwidth may be incrementally and gradually subdivided—in a consistent manner—from a four-class model through to an eight-class model.

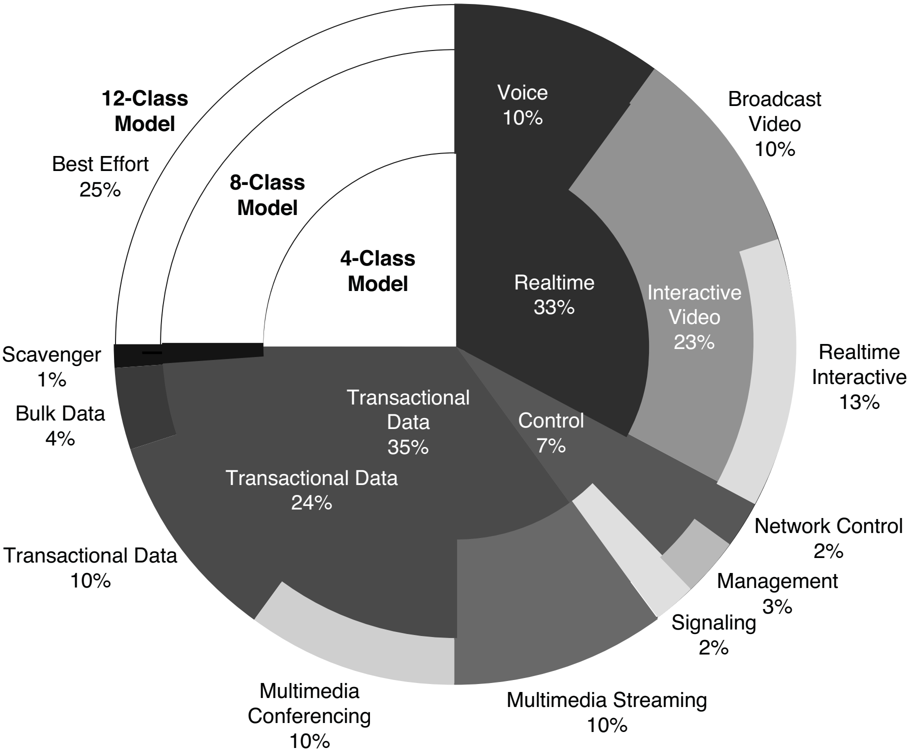


Figure 11-10 *Application-Class Expansion Queuing Strategy Example*

QoS for Security Strategies

Although the primary objective of most QoS deployments is to provision preferential—and sometimes deferential—services to various application classes, a strategic role of QoS policies may also be to provide an additional layer of security to the network infrastructure, especially in the case of mitigating denial-of-service (DoS) and worm attacks.

There are two main classes of DoS attacks:

- **Spoofing attacks:** The attacker pretends to provide a legitimate service but provides false information to the requester (if any).
- **Slamming attacks:** The attacker exponentially generates and propagates traffic until service resources (servers/network infrastructure) are overwhelmed.

Spoofing attacks are best addressed by authentication and encryption technologies. Slamming (also known as flooding) attacks, however, can be effectively mitigated through QoS technologies.

In contrast, worms exploit security vulnerabilities in their targets and disguisedly carry harmful payloads that usually include a self-propagating mechanism. Network infrastructure is usually not the direct target of a worm attack, but can become collateral damage as worms exponentially self-propagate. The rapidly multiplying volume of traffic flows eventually drowns the CPU/hardware resources of routers and switches in their paths, indirectly causing denial of service to legitimate traffic flows, as shown in Figure 11-11.

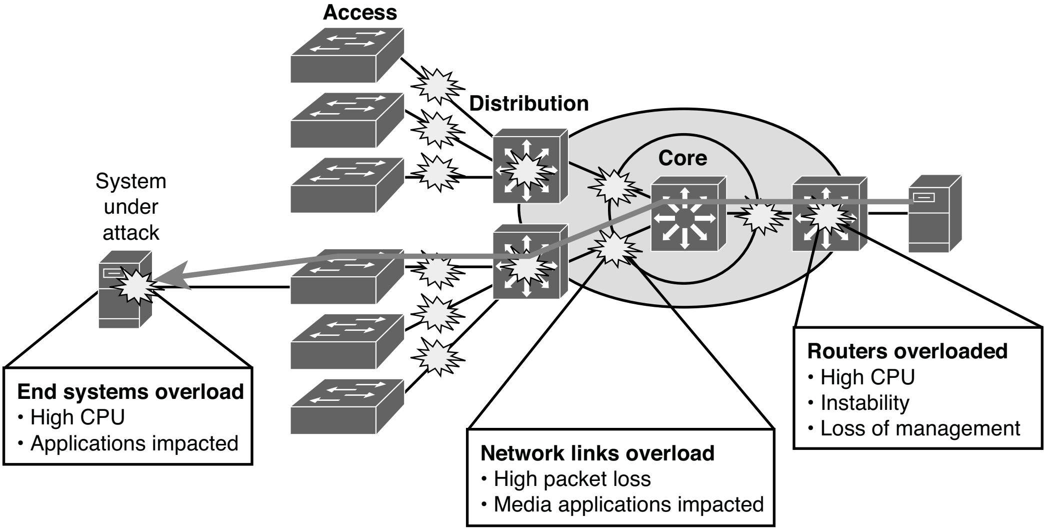


Figure 11-11 *Direct and Indirect Collateral Damage from DoS/Worm Attacks*

A reactive approach to mitigating such attacks is to reverse-engineer the worm and set up intrusion-detection mechanisms and access control lists / Network Based Application Recognition (ACLs/NBAR) policies to limit its propagation. However, the increased sophistication and complexity of worms make them harder and harder to separate from legitimate traffic flows. This exacerbates the finite time lag between when a worm begins to propagate and when the following can take place:

- Sufficient analysis has been performed to understand how the worm operates and what its network characteristics are.
- An appropriate patch, plug, or ACL is disseminated to network devices that may be in the path of worm. This task may be hampered by the attack itself, because network devices may become unreachable for administration during the attacks.

These time lags might not seem long in absolute terms, such as in minutes, but the relative window of opportunity for damage is huge. For example, the number of hosts infected with the Slammer worm doubled every 8.5 seconds on average and performed scans of more than 55 million other hosts within the first 11 minutes of its release!

A proactive approach to mitigating DoS/worm attacks within enterprise networks is to have control plane policing and data plane policing policies in place within the infrastructure that immediately respond to out-of-profile network behavior indicative of DoS or worm attacks. Control plane policing serves to protect the CPU of network devices—such as switches and routers—from becoming bogged down with interruption handling and thus not having enough cycles to forward traffic. Data plane policing—also referred to as Scavenger class QoS—serves to protect link bandwidth from being consumed by forwarding DoS/worm traffic to the point of having no room to service legitimate in-profile flows. In addition, data plane policing policies would also indirectly protect the control plane by reducing unnecessary packet processing requirements imposed on it.

Control Plane Policing Recommendations

A router or switch can be logically divided into four functional components or planes:

- Data plane
- Management plane
- Control plane
- Services plane

The vast majority of traffic travels through the router via the data plane. However, the route processor must handle certain packets, such as routing updates, keepalives, and network management. This is often referred to as *control and management plane traffic*.

Because the route processor is critical to network operations, any service disruption to the route processor or the control and management planes can result in business-impacting network outages. A DoS attack targeting the route processor, which can be perpetrated either inadvertently or maliciously, usually involves high rates of punted traffic (traffic that results in a processor interruption) that results in excessive CPU utilization on the route processor itself. This type of attack, which can be devastating to network stability and availability, may display the following symptoms:

- High route processor CPU utilization (near 100 percent).
- Loss of line protocol keepalives and routing protocol updates, leading to route flaps and major network transitions.
- Interactive sessions via the command-line interface (CLI) are slow or completely unresponsive due to high CPU utilization.
- Route processor resource exhaustion—resources such as memory and buffers are unavailable for legitimate IP data packets.
- Packet queue backup, which leads to indiscriminate drops (or drops due to lack of buffer resources) of other incoming packets.

Control plane policing (CPP for Cisco IOS routers or CoPP for Cisco Catalyst switches) addresses the need to protect the control and management planes, ensuring routing stability, availability, and packet delivery. It uses a dedicated control plane configuration via the Modular QoS CLI (MQC) to provide filtering and rate-limiting capabilities for control plane packets.

Figure 11-12 illustrates the flow of packets from various interfaces. Packets destined to the control plane are subject to control plane policy checking, as depicted by the Control Plane Services block.

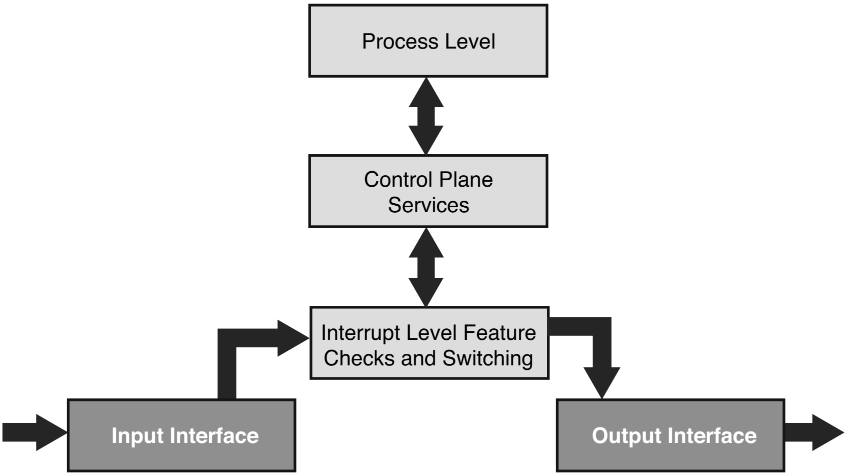


Figure 11-12 *Packet Flow Within a Switch/Router*

By protecting the route processor, CPP/CoPP helps ensure router and network stability during an attack. For this reason, a best-practice recommendation is to deploy CPP/CoPP as a key protection mechanism on all routers and switches that support this feature.

To successfully deploy CPP, the existing control and management plane access requirements must be understood. Although it can be difficult to determine the exact traffic profile required to build the filtering lists, the following summarizes the recommended steps necessary to properly define a CPP policy:

1. Start the deployment by defining liberal policies that permit most traffic.
2. Monitor traffic pattern statistics collected by the liberal policy.
3. Use the statistics gathered in the previous step to tighten the control plane policies.

Note Because the primary role of CoPP is a security mechanism, it is for the most part beyond the scope of discussion of this book, although Example 4-5 in Chapter 4, “Policing, Shaping, and Markdown Tools,” provides a CoPP policy example. For additional details on deploying CoPP, refer to platform-specific configuration guides on this feature.

Data Plane Policing Recommendations

The logic applied to protecting the control plane can also be applied to the data plane. Data plane policing has two components:

- Campus access-edge policers that meter traffic flows from endpoint devices and remark “abnormal” flows to CS1 (the scavenger marking value)
- Queuing policies on all nodes that include a deferential service class for scavenger traffic

Figure 11-13 illustrates these two components of data plane policing QoS strategy.

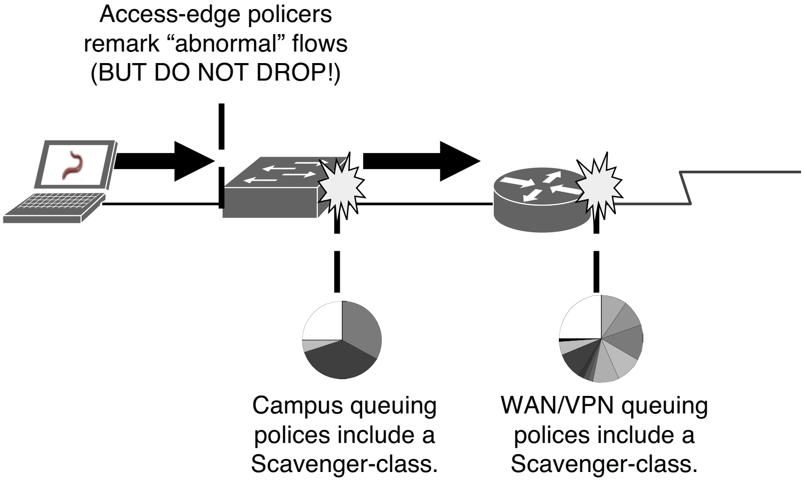


Figure 11-13 *Packet Flow Within a Switch/Router*

Most endpoint devices have fairly predictable traffic patterns and, therefore, can have metering policers to identify “normal” flows (the volume of traffic that represents 95 percent of the typically generated traffic rates for the endpoint device) versus “abnormal” flows (the remainder). For instance, it would be abnormal for a port that supposedly connects to an IP phone to receive traffic in excess of 128 Kbps. Similarly, it would be abnormal for a port that supposedly connects to a Cisco TelePresence System to receive traffic in excess of 20 Mbps. Both scenarios would be indicative of network abuse—either intentional or inadvertent. Endpoint PCs and wireless devices also have traffic patterns that can be fairly accurately baselined with statistical analysis

Note For example, for users of Windows-based systems, the Windows Task Manager (which can be selected by simultaneously pressing Ctrl+Alt+Del) can graphically display networking statistics (available from the networking tab). Most users are generally surprised at how low the average network utilization rates of PCs are during everyday use, as compared to their link speed capacities.

These access-edge metering policers are relatively unintelligent. They do not match specific network characteristics of specific types of attacks, but simply meter traffic volumes and respond to abnormally high volumes as close to the source as possible. The simplicity of this approach negates the need for the policers to be programmed with knowledge of the specific details of how the attack is being generated or propagated. It is precisely this unintelligence of such access layer metering policers that allow them to maintain relevancy as worms mutate and become more complex. The policers do not care how the traffic was generated or what it looks like; they care only how much traffic is being put onto the wire. Therefore, they continue to police even advanced worms that continually change their tactics of traffic generation.

For example, in most enterprises, it is quite abnormal (within a 95 percent statistical confidence interval) for PCs to generate sustained traffic in excess of 5 percent of link capacity. In the case of a Gigabit Ethernet access switch port, this means that it would be unusual in most organizations for an end user PC to generate more than 50 Mbps of uplink traffic on a sustained basis.

Note It is important to recognize that this value (5 percent) for normal endpoint utilization by PC endpoints is just an example. This value would likely vary from enterprise to enterprise and within a given enterprise (such as by departmental functions).

It is important to recognize that what is being recommended by the data plane policing QoS strategy is not to police all traffic to 50 Mbps and automatically drop the excess. Should that be the case, there would not be much reason to deploy Gigabit Ethernet switch ports to endpoint devices But rather, these campus access layer policers do not drop traffic at all; they only perform re-marking (if traffic rates appear abnormal). These policers are coupled with queuing polices on all network nodes that include a deferential service class for traffic marked as scavenger (CS1). Queuing policies only engage when links are congested; so, if links capacity exists, traffic is never dropped. It is only in scenarios where offered traffic flows exceed link capacity—forcing queuing polices to engage and queuing buffers to fill to capacity—that drops might occur. In such scenarios, dropping can either occur indiscriminately (on a last-come, first-dropped basis) or with a degree of intelligence (as would be the case if abnormal traffic flows were previously identified).

Let’s illustrate how this might work for both legitimate excess traffic and in the case of illegitimate excess traffic resulting from a DoS or worm attack.

In the former case, assume that the PC generates over 50 Mbps of traffic, perhaps because of a large file transfer or backup. Congestion (under normal operating conditions) is rarely if ever experienced within the campus because there is generally abundant capacity to carry the traffic. Uplinks to the distribution and core layers of the campus network are usually Gigabit Ethernet or 10 Gigabit Ethernet, which would require 1000 or 10000 Mbps of traffic (respectively) from the access layer switch to congest. If the traffic is destined to the far side of a WAN/VPN link, queuing and dropping usually occur even without the access layer policer because of the bottleneck caused by the typical campus-to-WAN/VPN speed mismatch. In such a case, the TCP sliding windows mechanism would eventually find an optimal speed (under 50 Mbps) for the file transfer. Access layer policers that mark down out-of-profile traffic to scavenger (CS1) would therefore not affect legitimate traffic, aside from the obvious re-marking. No reordering or dropping would occur on such flows as a result of these data plane policers that would not have occurred anyway.

In the latter case, the effect of access layer policers on traffic caused by DoS or worm attacks is quite different. As hosts become infected and traffic volumes multiply, congestion might be experienced even within the campus. For example, if just 11 end-user PCs on a single access switch begin spawning worm flows to their maximum Gigabit Ethernet link capacities, even 10 Gigabit Ethernet uplinks/core links will congest and queuing and dropping policies will engage. At this point, VoIP and media applications, and even best-effort applications, would gain priority over worm-generated traffic (because scavenger traffic would be dropped the most aggressively). Furthermore, network devices would remain accessible for administration of the patches/plugs/ACLs/NBAR policies required to fully neutralize the specific attack. WAN/VPN links would also be similarly protected, which is a huge advantage, because generally WAN/VPN links are the first to be overwhelmed by DoS/worm attacks. Scavenger class policies thus significantly mitigate network traffic generated by DoS or worm attacks.

Therefore, for network administrators to implement this data plane policing QoS strategy, they need to first profile applications to determine what constitutes normal as opposed to abnormal flows, within a 95 percent confidence interval. Thresholds demarking normal/abnormal flows vary from enterprise to enterprise and from application to application. Beware of overscrutinizing traffic behavior, because this could exhaust time and resources and could easily change daily. Remember, legitimate traffic flows that temporarily exceed thresholds are not penalized by the presented Scavenger class QoS strategy. Only sustained, abnormal streams generated simultaneously by multiple hosts (highly indicative of DoS/worm attacks) are subject to aggressive dropping only after legitimate traffic has been serviced.

To contain such abnormal flows, deploy campus access-edge policers (with large committed burst values) to re-mark abnormal traffic to scavenger (CS1). In addition, whenever possible, deploy a second line of policing defense at the distribution layer, leveraging tools such as microflow policers. And to complement these re-marking policies, it is necessary to enforce deferential Scavenger class queuing policies throughout the network.

A final word on this subject: It is important to recognize the distinction between mitigating an attack and preventing it entirely. Control plane policing and data plane policing policies do not guarantee that no DoS or worm attacks will ever happen, but serve only to reduce the risk and impact that such attacks could have on the network infrastructure. Therefore, it is vital to overlay a comprehensive security strategy over the QoS-enabled network infrastructure.

Summary

This chapter began by presenting several best-practice design principles that serve as a basis for end-to-end QoS strategies. These principles included the following:

- Always enable QoS policies in hardware—rather than software—whenever a choice exists.
- Classify and mark applications as close to their sources as technically and administratively feasible.
- Use DSCP marking whenever possible.
- Follow standards-based DSCP PHB markings to ensure interoperability and future expansion.
- Police traffic flows as close to their source as possible and mark down according to standards-based rules.
- Enable queuing policies at every node that has the potential for congestion.
- Whenever possible, assign each application class to its own dedicated queue.
- Use only platforms/service providers that offer a minimum of four standards-based queuing behaviors:
  - An RFC 3246 Expedited Forwarding PHB
  - An RFC 2597 Assured Forwarding PHB
  - An RFC 2474 Default Forwarding PHB
  - An RFC 3662 Lower Effort per-domain behavior
- Limit the amount of strict-priority queuing to 33 percent of link bandwidth capacity.
- Govern strict-priority traffic with an admission control mechanism.
- Do not enable WRED on the strict-priority queue.
- Provision guaranteed-bandwidth allocations and enable DSCP-based WRED on AF queues.
- Provision at least 25 percent of link bandwidth for the default Best Effort class and enable WRED (effectively RED) on the default class.
- Assign a minimum bandwidth to the Scavenger class queue.

In addition, end-to-end QoS strategies were presented for a

- 4-class QoS model
- 8-class QoS model
- 12-class QoS model

Finally, attention was given to how to best plan for application-class expansion and how QoS tools can be strategically used to improve network infrastructure security.

Further Reading

Medianet QoS Design Strategy At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qosmrn.pdf

Enterprise Medianet Quality of Service Design 4.0—Overview: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html

Chapter 12

Strategic QoS Design Case Study

This chapter is the first in a series of case study design chapters that will conclude each of the remaining parts of this book. The purpose of these case study design chapters is to apply the key design concepts and recommendations presented in each part—some of which might seem abstract and theoretical—in a practical, cohesive, and comprehensive manner. Each case study design chapter builds upon the previous and collectively represents the entire end-to-end quality of service (QoS) design of a fictional enterprise: Tifosi Software Inc.

Tifosi Software Inc.: Company Overview

Tifosi Software Inc. is a (fictional) company based in Santa Clara, California, that specializes in cloud computing software. Tifosi employs about 5000 people worldwide, with engineering and sales offices in North America (United States, Canada, and Mexico), South America (Brazil and Argentina), Europe (Belgium, France, Germany, Switzerland, and the United Kingdom), Asia-Pacific (Australia, China, Hong Kong, India, Japan, and Singapore), and the Middle-East/Africa (in Dubai, Israel, and South Africa).

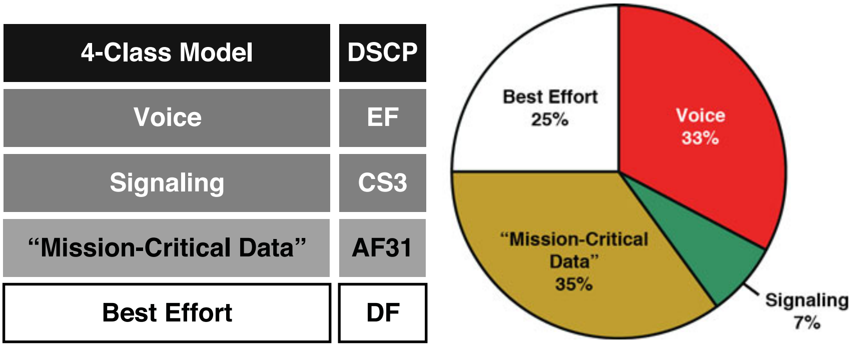
Original (Four-Class) QoS Model

Tifosi's original QoS model was fairly basic, but had been based on Cisco recommendations. It consisted of a four-class QoS model to support IP telephony and a locally defined "mission-critical data" traffic class. Explicitly, these classes included the following:

- **Voice:** Marked EF and provisioned with 33 percent strict-priority EF PHB
- **Signaling:** Marked CS3 and provisioned with 7 percent guaranteed bandwidth
- **"Mission-Critical Data":** Marked AF31 and provisioned with 35 percent AF PHB
- **Best Effort:** Marked DF and provisioned with a 25 percent bandwidth guarantee

Traffic for the mission-critical data class was selected by each director of Tifosi's major departments (including engineering, sales, marketing, finance and human resources), who submitted their top-three business-critical applications to the networking team. These, in turn, were all marked AF31 at the network edges and provisioned with an AF PHB end to end.

Figure 12-1 shows Tifosi Sofware's original QoS class and queuing models.



4-Class Model	DSCP
Voice	EF
Signaling	CS3
"Mission-Critical Data"	AF31
Best Effort	DF

Figure 12-1 *Tifosi Software Case Study: Original Four-Class QoS and Queuing Models*

Business Catalysts for QoS Reengineering

Several business requirements led Tifosi Software's CIO, Rod Bratton, to mandate a reengineering of the network QoS policies. These requirements included the following:

- The imminent deployment of 24 Cisco TelePresence systems—1 each in every major engineering and sales office, with some of the larger sites hosting more than one—with future plans for more to come; this deployment included a corresponding bandwidth upgrade in all WAN/VPN links
- The emerging popularity of multimedia collaboration applications, like Cisco WebEx and Jabber and Microsoft Lync
- The identification of "substantial" amounts of nonbusiness traffic on the network during work hours, such as YouTube, BitTorrent, iTunes downloads, and gaming traffic
- A desire for greater overall QoS policy-consistency, as Tifosi had acquired a few smaller companies over the years and had noticed significant disparity in the QoS models some of these companies had deployed versus their own

To this end, Rod approached Tifosi's senior network architect, Tom Spaghetti, to discuss these concerns. After lengthy discussion, Tom recommended to Rod that to improve QoS consistency and to meet their new business requirements, Tifosi should update their QoS strategy and base it on RFC 4594, a de facto industry standard for differentiated services (DiffServ) QoS design. Rod agreed in principle, but he had deep concerns about the technical complexity of deploying and operating a 12-class model as outlined in that RFC, which he though would simply be too big a "jump" from their current 4-class model. Furthermore, he pointed out that their IP telephony infrastructure was based on Cisco Unified Communication Managers (CUCM), and therefore, all their signaling traffic was marked by default to CS3 (and not CS5, as specified in the RFC). Tom agreed to give the matter more consideration and research and to follow up with Rod.

Proposed (Eight-Class) QoS Model

At their follow-up meeting, Tom shared with Rod that because RFC 4594 was an informational RFC (and not an Internet standard) complying with it—in whole or in part—was at the discretion of any given organization. Based on this, Tom suggested that Tifosi adopt an eight-class QoS model, which could be largely based on RFC 4594. Such an eight-class model would both expand—and slightly modify—their existing four-class model. Rod liked the idea, and after several more rounds of discussion on the per-class details, Tom proposed the following eight-class model:

- **Voice:** Marked EF and provisioned with an EF PHB with 10 percent strict-priority queuing
- **Realtime-Interactive:** Marked CS4 yet provisioned with an EF PHB with 23 percent strict-priority queuing
- **Signaling:** Marked CS3 and provisioned with 2 percent guaranteed bandwidth
- **Multimedia-Conferencing:** Marked AF41 and treated with an AF PHB, provisioned with guaranteed bandwidth allocation of 10 percent, with DSCP-based WRED enabled
- **Transactional Data:** Marked AF21 and treated with an AF PHB, provisioned with guaranteed bandwidth allocation of 25 percent, with DSCP-based WRED enabled
- **Bulk Data:** Marked AF11 and treated with an AF PHB, provisioned with guaranteed bandwidth allocation of 4 percent, with DSCP-based WRED enabled
- **Scavenger:** Marked CS1 and treated with a bandwidth-constrained queue provisioned at 1 percent
- **Best Effort:** Marked DF and provisioned with a guaranteed bandwidth allocation of 25 percent, with WRED-enabled

Tom's per-class reasoning was as follows:

- **Voice:** With the increase in bandwidth associated with the TelePresence deployments, the corresponding relative bandwidth allocated to the voice class could be reduced while still maintaining the same call volume of traffic.

- **Realtime-Interactive:** This new class could service TelePresence traffic and could be provisioned with an EF PHB (per RFC 4594) to provide this highly drop-sensitive application with the highest level of quality. Furthermore, the amount of bandwidth allocated to the Voice and Realtime-Interactive classes keeps real-time traffic to within one-third of a given link's capacity, in line with best-practice design recommendations.

- **Signaling:** With the increase in bandwidth, this class could be reduced in relative size while still maintaining the same volume of signaling traffic; traffic in this class could continue being marked CS3 (again drawing on the fact that RFC 4594 is not a standard and therefore may be modified as needed.)

- **Multimedia-Conferencing:** This new class could service all multimedia conferencing and collaborative applications, most of which were analyzed to exhibit an elastic behavior, such that implementing weighted random early detection (WRED) policies on this class would provide effective congestion avoidance.

- **Transactional Data:** This class (in conjunction with the Bulk Data class) could serve to replace the company-specific "Mission-Critical Data" class and make it more compliant with industry best practices. In addition, this would optimize the network to accommodate "foreground" data applications. Furthermore, the marking for this class could be changed to AF21 (from AF31) to allow for a potential future class for multimedia-streaming applications.

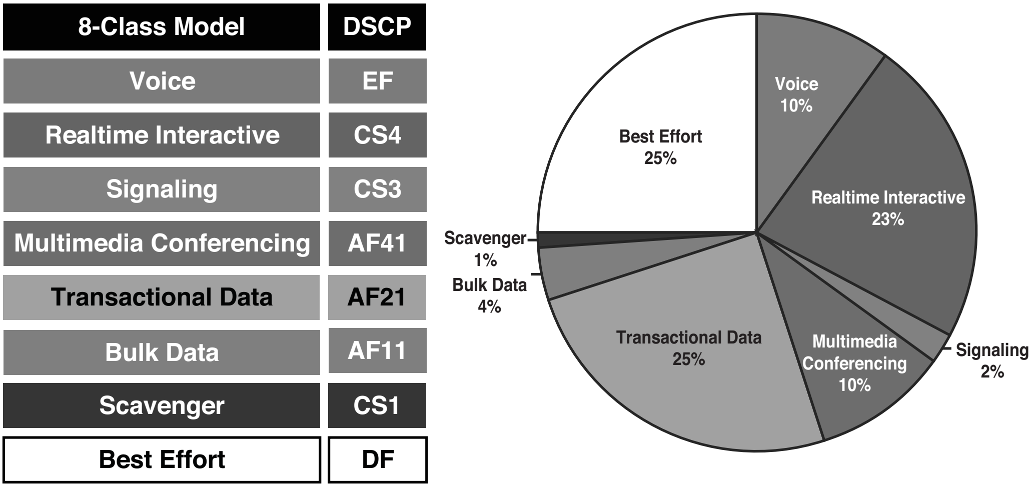
- **Bulk Data:** This class (along with the Transactional Data class) could serve to replace the "Mission-Critical Data" class. Specifically, this class could be used to provide a degree of bandwidth constraint—during periods of congestion—to "background" data applications.

- **Scavenger:** This class could serve to provide a maximum bandwidth constraint to nonbusiness applications during periods of congestion. Instead of shutting these down entirely (which would likely prove unpopular with the user base), these could be restricted and dropped aggressively in the event of bandwidth congestion. Otherwise, they would continue to be serviced by the network. In addition, this class could service all unmanaged applications and devices (such as BYOD devices) that do not meet the security and administrative requirements that the IT department has set. Finally, this class would also service traffic from guest-access VLANs (both wired and wireless).

- **Best Effort:** All remaining applications would continue to receive a default service that would (collectively) amount to no less than a quarter of any given link's bandwidth.

Figure 12-2 illustrates Tom's proposed eight-class QoS and queuing models.

After careful consideration, Rod decided to adopt Tom's proposed eight-class model for Tifosi Software. They then sat down and reviewed all the applications within the "Mission-Critical Data" class and subdivided these (mainly) into the Transactional Data or Bulk Data traffic classes depending on the traffic pattern of the application, primarily considering whether the application was a foreground or a background application. Some applications, such as streaming media applications, were provisioned into the Best Effort class for the time being, where they would be monitored until they comprised a significant enough degree of bandwidth, in which case they would be assigned their own dedicated application class in the future.



8-Class Model	DSCP
Voice	EF
Realtime Interactive	CS4
Signaling	CS3
Multimedia Conferencing	AF41
Transactional Data	AF21
Bulk Data	AF11
Scavenger	CS1
Best Effort	DF

Figure 12-2 *Tifosi Software Case Study: Proposed Eight-Class QoS and Queuing Models*

Finally, Rod sent out a memo to all departments overviewing the new QoS model, the reasons behind it, and the corresponding changes this would involve.

“Layer 8” Challenges

Shortly after Rod’s email was sent out outlining the new network QoS model, he received a flurry of responses, not a few of which were flames.

One of the most vocal opponents of the new QoS direction was Katrina Hating, the director of the HR department. Under the original QoS model, Katrina’s top-three applications serviced within the “Mission-Critical Data” application class were as follows:

- An Oracle-based enterprise resource planning (ERP) application managing all HR records and information
- Employee training videos on demand (VoDs) on company policies and employee issues
- Email

Under the new QoS model

- The Oracle ERP application would be serviced as transactional data.
- Training VoDs would be serviced as best effort.
- Email would be serviced as bulk data.

The reasoning behind these changes was that Rod and Tom recognized the Oracle ERP application as transactional in nature, being a foreground data application (for which user productivity would be directly impacted by network delays) and therefore a prime candidate for the Transactional Data class. The training VoDs really did not fit into the Transactional Data class, but would fit much better into a proposed future Multimedia Streaming class. However, Rod wanted to track the volume of these flows and then provision these as a dedicated class when they passed a specific benchmark threshold. In the meantime, because these streaming applications included significant application-level buffering capabilities, he saw no need to explicitly provision for them at this juncture. Finally, both Rod and Tom agreed that email—although important to business operations—better suited the profile of a bulk data application (rather than a transactional data application), because emails could be sent and received in the background, while users continue to work, and therefore network delays servicing email operations would not directly impact user productivity.

Rod explained these reasons to Katrina, both in person and in several email exchanges that followed. Nonetheless, Katrina remained unconvinced and felt that the new QoS policies were not only arbitrary but also unfair and that they would ultimately hinder the productivity of her department. She then went over Rod's head and raised the issue with the CEO of Tifosi Software, Lola Serengeti.

Lola promptly set up a meeting with Rod, Katrina, and several other department heads who had issues with the new policy to settle these immediately. After hearing each manager out, she grilled Rod extensively on the new QoS strategy. Rod defended the direction and added that QoS was inherently unfair and so would almost inevitably spark controversy at the organizational and political levels—which he joking referred to as the "eighth layer of the OSI networking model." The joke fell flat because no one in the room—or on the phone—had ever even heard of the OSI layers. Nonetheless, Lola backed Rod's decision—reasoning that, ultimately, this fell under his domain of responsibility—and mandated the deployment of the new QoS model.

After the meeting, and after having some time to reflect on the preceding events, Rod decided that going forward he would review changes to networking QoS policies with the executive staff *beforehand*—thoroughly explaining how these would serve to advance strategic business requirements and thus benefit the company—in a hope to circumvent such potential conflicts in the future.

Note This eight-class model—shown in Figure 12-2—will be used in all case study examples to follow.

Summary

This chapter introduced a fictional company—Tifosi Software—on which all case study chapters in this book are based.

Tifosi Software began with a four-class QoS model to service: voice, signaling, mission-critical data and best-effort applications. The applications comprising the "Mission-Critical Data" class were largely subjectively decided on, without having their traffic patterns and service level requirements objectively analyzed.

A number of business developments—including the deployment of Cisco TelePresence systems, a growing popularity of multimedia collaboration and conferencing applications, and the observance of a significant volume of nonbusiness traffic on the production network—led the company to reevaluate its current QoS strategy and consider how it may be improved and expanded to meet these evolving business needs.

An eight-class QoS strategy was then presented to accommodate these requirements and the rationale for each addition and modification was explained.

Following this, an example political backlash was illustrated (something extremely common to QoS deployments because, after all, QoS is an inherently unfair technology). As shown in this chapter, the key to handling these political challenges was to give extensive thought to the overall strategy, make objective (rather than subjective) decisions about applications and application classes, and garner executive endorsement before deployment whenever possible.

Additional Reading

Medianet QoS Design Strategy At-a-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qosmrn.pdf

Enterprise Medianet Quality of Service Design 4.0—Overview: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html

IETF RFC 4594, Configuration Guidelines for DiffServ Service Classes: http://www.ietf.org/rfc/rfc4594

This page intentionally left blank

Chapter 13

Campus QoS Design Considerations and Recommendations

The primary role of quality of service (QoS) in rich-media campus networks is not to control latency or jitter (as it is in the wide-area network [WAN] or virtual private network [VPN]), but to manage packet loss. In today's campus networks, endpoints are connecting to the network at Gigabit Ethernet (GE) speeds, and it takes only a few milliseconds of congestion to cause instantaneous buffer overruns resulting in packet drops at the edge or in the 10GE/40GE core.

For example, consider one of the deepest-buffered linecards available (at the time of this writing): the Cisco Catalyst 6900 Series 40 Gigabit Ethernet Interface Module (WS-X6904-40G-2T). This linecard offers 176 MB of buffers, shared across each pair of ports. Assuming each port is configured to use all eight available egress hardware queues with an equal proportion of buffers assigned to each queue, this allows for 11 MB of buffers per port-queue (176 MB / 2 ports / 8 queues). At 40-Gbps speeds, these queues would take a mere 2.2 ms of line-rate microbursts to fill to capacity (11 MB * 8 bits-per-byte / 40 Gbps), at which point they would begin to drop packets. As has been previously pointed out, rich-media applications—particularly high-definition (HD) video applications—could be severely impacted because they are extremely sensitive to packet drops, to the point where the end user can discern even 1 packet dropped in 10,000.

A secondary role that QoS often plays in the campus is to differentiate and manage application traffic, particularly at the access edge.

Therefore, several of the strategic QoS design principles discussed in Chapter 11, "QoS Design Principles and Strategies," apply to campus QoS designs, including the following:

- **Always perform QoS in hardware rather than software when a choice exists:** Cisco Catalyst switches perform QoS in dedicated hardware application-specific integrated circuits (ASICs) and therefore do not tax their main CPUs while administering QoS policies. You can therefore apply and efficiently enforce even complex QoS policies at GE/10GE line rates on these platforms.

- **Classify and mark applications as close to their sources as technically and administratively feasible:** This principle promotes end-to-end differentiated services/per-hop behaviors (DiffServ/PHBs). Sometimes, endpoints can be trusted to set class of service (CoS) or differentiated services code point (DSCP) markings correctly, but this is not always recommended because users could easily abuse provisioned QoS policies if permitted to mark their own traffic (either deliberately or inadvertently). For example, if DSCP Expedited Forwarding (EF) received priority services throughout the enterprise, a user could easily configure the network interface card (NIC) on a PC to mark all traffic to DSCP EF, thus hijacking his way into network priority queues to service their non-real-time traffic. Such abuse could easily ruin the service quality of real-time applications (like VoIP) throughout the enterprise.
- **Police unwanted traffic flows as close to their sources as possible:** There is little sense in forwarding unwanted traffic only to police and drop it at a subsequent node. This is especially the case when the unwanted traffic is the result of denial-of-service (DoS) or worm attacks.
- **Enable queuing policies at every node where the potential for congestion exists, regardless of how rarely this in fact might occur:** This principle applies both to campus edge and interswitch links, where oversubscription ratios create the potential for instantaneous congestion. Because of the speed of campus networks (GE/10GE), most switch platforms can accommodate only a few dozen milliseconds of buffering (at best) before these fill and packets begin to be dropped. Therefore, the only way to guarantee service levels in the campus and to prevent packet dropping on real-time traffic classes is to enable queuing policies on all nodes where the potential for congestion exists.
- **(Optional) Protect the control plane and data plane by enabling control plane policing:** This hardens the campus network infrastructure and data plane policing (Scavenger class QoS) on campus network switches to mitigate and constrain network attacks.

However, before these strategic QoS design principles can be translated into platform-specific configuration recommendations, a few additional campus-specific considerations need to be taken into account, including the following:

- Multi-Layer Switch (MLS) versus Modular QoS command-line-interface (MQC)
- Default QoS
- Internal DSCP
- Trust states and operations
- DSCP transparency
- Trust boundaries
- Port-based QoS versus VLAN-based QoS versus per-port/per-VLAN QoS
- EtherChannel QoS

- Campus QoS models
- Campus port QoS roles
- Campus AutoQoS
- Control plane policing (CoPP)

Each of these campus specific considerations is discussed in turn.

MLS Versus MQC

Design recommendation:

> If a switch supports both MQC and MLS, choose the MQC version of Cisco IOS Software; this will improve cross-platform QoS consistency.

Cisco Catalyst switches (at the time of this writing) use one of two configuration syntaxes:

- MLS
- MQC

MLS QoS is the older of the two syntaxes and usually has many commands prefaced with the **mls** prefix. For example, to enable QoS globally on a MLS-QoS-based switch, the command is (usually) **mls qos**.

MQC, however, is the IOS QoS syntax structure based on class maps, policy maps, and service policy statements.

Note MLS QoS does leverage class maps and policy maps for classification/policing policies, but not for queuing policies, whereas MQC leverages these for nearly all QoS functions.

In an effort to improve QoS cross-platform consistency, MQC was ported from IOS-based routers to Catalyst switches. This began with Cisco IOS Release 12.2(40)SG for the Catalyst 4500E-Supervisor 6-E.

Note The Cisco IOS Release 12.2(40)SG was also supported on the Catalyst 4900M, Catalyst 4948E, and the Supervisor Engine 6L-E.

As part of the porting process, many switch-specific commands/command sets needed to be added or adapted. For example, the concept of conditional trust (which is discussed in more detail later in this chapter) did not exist in IOS router MQC and therefore needed to be added for switch-based MQC.

In the design chapters that follow, the access layer Catalyst 3750-X QoS designs feature MLS QoS. The distribution layer Catalyst 4500E – Sup7E and the core layer Catalyst 6500 – Sup2T QoS designs feature MQC.

The biggest difference between these two syntaxes relates to queuing configurations; nonetheless, there are also some more subtle distinctions (both in syntax and in function) between MLS QoS and MQC, which are pointed out when necessary.

Default QoS

Design recommendation:

> If configuring QoS on an MLS QoS-based switch, globally enable QoS as a first step.

One such functional difference between MLS QoS and MQC is the default QoS state and behavior.

By default, MLS QoS is disabled, meaning that all MLS-related QoS functions on the switch are also disabled. For example, if a packet with Layer 2 or Layer 3 markings enters the switch, it will exit the switch with the same L2/L3 markings preserved. In addition, all packets are assigned to a default queue, which is scheduled in a first-in, first-out (FIFO) manner. Furthermore, although any QoS commands that you may configure (such as trust, classification, policing, or queuing commands) will show up in the configuration, *none* of these will be active until the global **mls qos** command is manually entered.

Note Once QoS is globally enabled on an MLS QoS switch, all ports are set to an untrusted state, meaning that without an explicit trust configuration, all packets will have their DSCP field re-marked to 0.

In contrast, with MQC, QoS is enabled by default, and there is no global command (because there is no need for one) to enable QoS. Nonetheless, the default treatment of packets is similar to MLS: L2/L3 packet markings are preserved by default and all packets are scheduled with FIFO queuing by default. Only when you configure a policy—via class maps and a policy map—and bind it to an interface with a service policy statement will there be a change to this default QoS behavior.

Internal DSCP

Design recommendation:

> Understand the internal DSCP concept as this may impact MLS QoS-based designs

MLS QoS switches perform QoS operations based on the concept of an “internal DSCP” value that is associated with each packet.

Note The term *packet* is being used loosely in this chapter to describe not only Layer 3 packets but also Layer 2 frames.

The internal DSCP (which is sometimes referred to as a *QoS label*, but is not to be confused with an Multiprotocol Label Switching [MPLS] label) is used by MLS QoS switches to determine whether a packet is to be re-marked, policed, or to which queue it is to be assigned or whether it should be dropped. The internal DSCP value may (or may not) be the same as the actual DSCP value of an IP (IPv4 or IPv6) packet. Furthermore, an internal DSCP value is generated even for non-IP protocols (such as Layer 2 protocols like Spanning Tree Protocol and non-IP Layer 3 protocols like IPX).

The manner in which the internal DSCP value is generated for a packet depends on the trust state of the port on which the packet was received, which is described next.

Trust States and Operations

Design recommendation:

> Understand the operation and implications of the various static and dynamic trust states.

There are three (static) trust states with which a switch port can be configured:

Note Technically, there are four static trust states if trust IP precedence is included. However, this option has long been relegated by trust DSCP and is therefore not considered in this discussion.

- **Untrusted:** A port in this trust state disregards any and all Layer 2 or Layer 3 markings that a packet may have and generates an internal DSCP value of 0 for the packet (unless explicitly overridden). This is the default port QoS trust state when **mls qos** has been globally enabled. However, a port may be explicitly set to an untrusted state with the interface configuration command **no mls qos trust**.

Note Most Cisco MLS-QoS-based switches include the **mls** prefix for these QoS commands. An exception exists across the Catalyst 4500/4900 MLS QoS-based switches (namely the Supervisor II+ through SupV-10GE series switches), which omits this prefix. However, because the 4500/4900 series switches are beyond the scope of this book (which instead focuses on the newer 4500E Supervisor 6-E/7-E series), the **mls** prefix will be included in these respective commands.

- **Trust CoS:** A port in this trust state accepts the 802.1p CoS marking of a 802.1Q tagged packet and uses this value—in conjunction with the CoS-to-DSCP mapping table—to calculate an internal DSCP value for the packet. The default CoS-to-DSCP mapping table multiplies each CoS value by a factor of 8 to calculate the default internal DSCP. (For example, CoS 1 maps to DSCP 8, CoS 2 maps to DSCP 16, and so on.) However, the default CoS-to-DSCP mapping table can be modified with the **mls qos map cos-dscp** global configuration command (for example, to map CoS 5 to the nondefault DSCP value of EF [46]). In the case of an untagged packet (such as a packet received from the native VLAN), the default internal DSCP value of 0 is applied. You can enable this port trust state with the interface configuration command **mls qos trust cos**.

- **Trust DSCP:** A port in this trust state accepts the DSCP marking of a packet and sets the internal DSCP value to match. In the case of a non-IP packet (such as an IPX packet), the default internal DSCP value of 0 is applied. This port trust state can be enabled with the interface configuration command **mls qos trust dscp**.

Figure 13-1 illustrates the operation of the untrusted, trust CoS, and trust DSCP states.

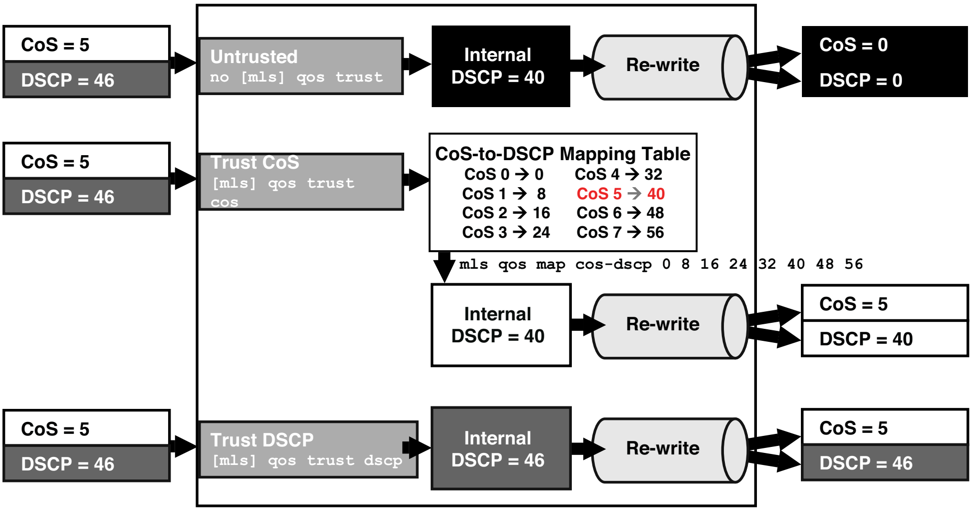


Figure 13-1 *Static Trust States and Operations*

In addition to the four static trust states described, Cisco Catalyst switches also support a dynamic trust state, where the applied trust state for a port can dynamically toggle, depending on a successful endpoint identification exchange and the configured endpoint trust policy. This feature is referred to as *conditional trust* and automates user/endpoint mobility. Conditional trust—in its current state—depends on device authentication via Cisco Discovery Protocol (CDP) and therefore is supported only on Cisco endpoint devices.

Note CDP is a lightweight, proprietary protocol engineered to perform neighbor discovery and so was never engineered to be used as a security authentication protocol. Therefore, CDP should not be viewed or relied on as secure, because it can be spoofed with relative ease.

Figure 13-2 illustrates conditional trust operation.

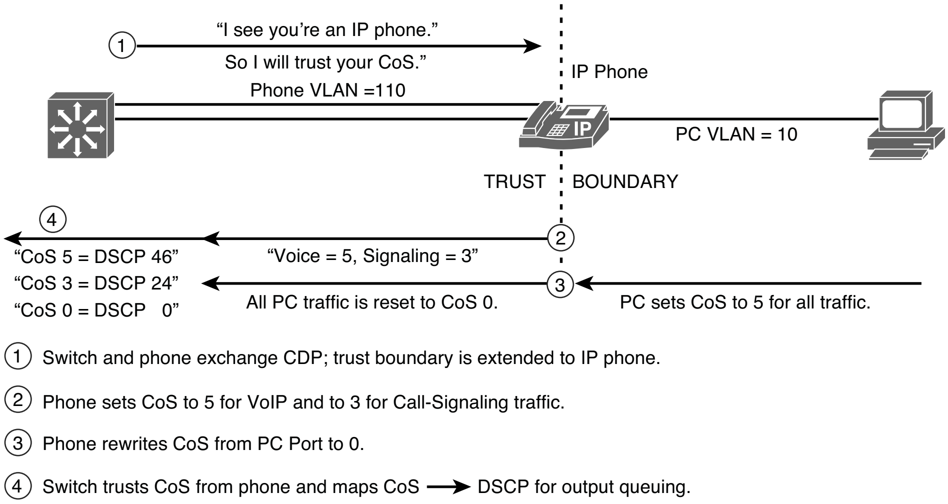

1
"I see you're an IP phone."
So I will trust your CoS."
Phone VLAN =110
IP Phone
PC VLAN = 10
TRUST
BOUNDARY
4
2
"CoS 5 = DSCP 46"
"CoS 3 = DSCP 24"
"CoS 0 = DSCP 0"
"Voice = 5, Signaling = 3"
3
All PC traffic is reset to CoS 0.
PC sets CoS to 5 for all traffic.
1 Switch and phone exchange CDP; trust boundary is extended to IP phone.
2 Phone sets CoS to 5 for VoIP and to 3 for Call-Signaling traffic.
3 Phone rewrites CoS from PC Port to 0.
4 Switch trusts CoS from phone and maps CoS → DSCP for output queuing.

Figure 13-2 *Conditional Trust Operation*

The sequence shown in Figure 13-2 is as follows:

1. The Cisco Catalyst access switch and Cisco IP phone exchange CDP information; after a successful exchange, the switch recognizes that the endpoint is an IP phone and—in accordance with the switch port's configured policy—can extend trust to it.
2. The Cisco IP phone sets CoS to 5 for VoIP and to 3 for call signaling traffic.
3. The Cisco IP phone rewrites CoS from the PC connected to the Cisco IP phone to 0.
4. The switch trusts CoS from phone and maps CoS-to-DSCP to generate internal DSCP values for all incoming packets.

Cisco endpoint devices supporting conditional trust include the following:

- Cisco IP phones via the **mls qos trust device cisco-phone** interface command
- Cisco TelePresence systems via the **mls qos trust device cts** interface command
- Cisco IP video surveillance cameras via the **mls qos trust device ip-camera** interface command
- Cisco digital media players via the **mls qos trust device media-player** interface command

Trust Boundaries

Design recommendation:

> Clearly delineate the trust boundary in the campus, separating endpoints into three classes:
>
> - Untrusted
> - Trusted
> - Conditionally trusted

Having reviewed the internal DSCP concept and trust state operations, you need to consider where to enforce the trust boundary (that is, the network edge at which packet markings are accepted—or not).

In line with the strategic QoS classification principle mentioned at the outset of this chapter, the trust boundary should be set as close to the endpoints as technically and administratively feasible.

The reason for the *administratively feasible* caveat within this design recommendation is that although many endpoints (including user PCs) technically support the ability to mark traffic on their NICs, allowing a blanket trust of such markings could easily facilitate network abuse, because users could simply mark all their traffic with Expedited Forwarding (EF), which would allow them to hijack network priority services for their non-real-time traffic and thus ruin the service quality of real-time applications throughout the enterprise.

Therefore, for many years it was advocated to not trust traffic from user PCs. However, more recently, various secure endpoint software has been released that allows for PC markings to be centrally administered and enforced. Such centrally administered software—along with quarantine VLANs for PCs that do not have such software installed—may present the option for network administrators to trust such secure endpoint PCs.

Therefore, from a trust perspective, there are three main categories of endpoints:

- **Conditionally trusted endpoints:** These include Cisco IP phones, Cisco TelePresence Systems, Cisco IP video surveillance cameras, and Cisco digital media players.

- **Trusted endpoints:** These can include centrally administered PCs and servers, IP video conferencing units, managed wireless access points, gateways, and other similar devices.
- **Untrusted endpoints:** Unsecure PCs, printers and similar devices.

Figure 13-3 illustrates example trust boundaries and configuration commands for each of these categories of endpoints.

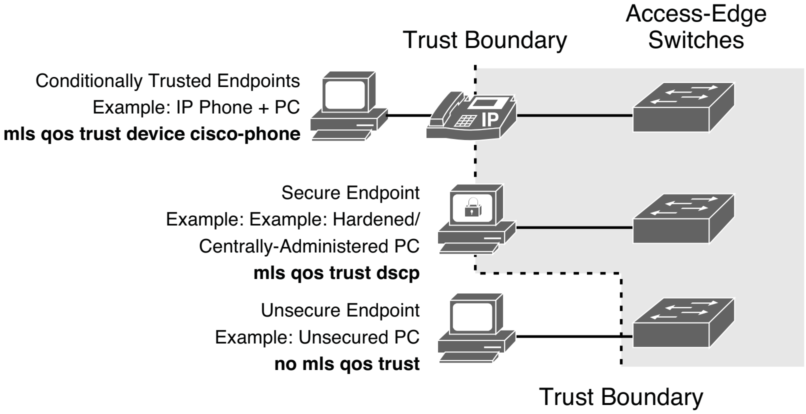


Figure 13-3 *Campus Trust Boundaries*

DSCP Transparency

Design recommendation:

> Be aware of the DSCP transparency feature; it may be handy in solving specific design requirements.

Regardless of how the internal DSCP is generated, it is important to note that as the packet exits the switch (unless explicitly overridden, as discussed in the following paragraph) the Catalyst switch sets the exiting IP packet's DSCP value to the final computed internal DSCP value. If trunking is enabled on the exiting switch port, the exiting packet's CoS value is also similarly set (but only to the first 3 bits of the final computed internal DSCP value, because CoS values are only 3 bits in length as compared to 6-bit DSCP values).

If you do not want the internal DSCP to overwrite the packet's ingress DSCP value, you can use the DSCP transparency feature, which is enabled by the **no mls qos rewrite ip dscp** global configuration command. When the DSCP transparency feature is enabled, the packet always has the same DSCP value on egress as it had on ingress, regardless of any internal QoS operations performed on the packet.

Port-Based QoS Versus VLAN-Based QoS Versus Per-Port/Per-VLAN QoS

Design recommendations:

- Use port-based QoS when simplicity and modularity are the key design drivers.
- Use VLAN-based QoS when looking to scale policies—such as classification, trust, and marking policies—without a corresponding increase in configuration length.
- Do not use VLAN-based QoS to scale (aggregate) policing policies.
- Use per-port/per-VLAN QoS when supported and when policy granularity is the key design driver.

QoS classification (including trust), marking, and policing policies on Cisco Catalyst switches can be applied in one of three ways:

- **Port-based QoS:** When a QoS policy is applied on a per-port basis, it is attached to a specific physical switch port and is active on all traffic received on that specific port (only). QoS policies are applied on a per-port basis by default. Figure 13-4 illustrates port-based QoS.

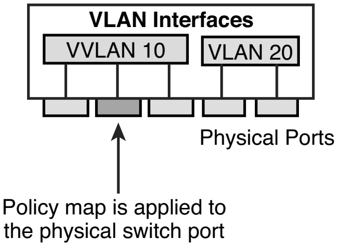


Figure 13-4 *Port-Based QoS*

- **VLAN-based QoS:** When a QoS policy is applied on a per-VLAN basis, it is attached to a logical VLAN interface and is active on all traffic received on all ports that are currently assigned to the VLAN. Applying QoS polices on a per-VLAN basis requires the **mls qos vlan-based** interface command on MLS-based QoS switches. Alternatively, on MQC-based switches the service policy is simply applied to the VLAN interface (rather than the physical interface). Figure 13-5 illustrates VLAN-based QoS.

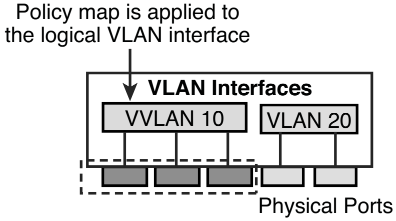


Figure 13-5 *Per-VLAN QoS*

- **Per-port/per-VLAN QoS:** When a QoS policy is applied on a per-port/per-VLAN basis, it is attached to specific VLAN on a trunked port and is active on all traffic received from that specific VLAN from that specific trunked port (only). Per-port/per-VLAN QoS is not supported on all platforms, and the configuration commands are platform-specific. Figure 13-6 illustrates per-port/per-VLAN QoS on voice and data VLANs (although this feature could be used on any VLANs carried by a trunked port).

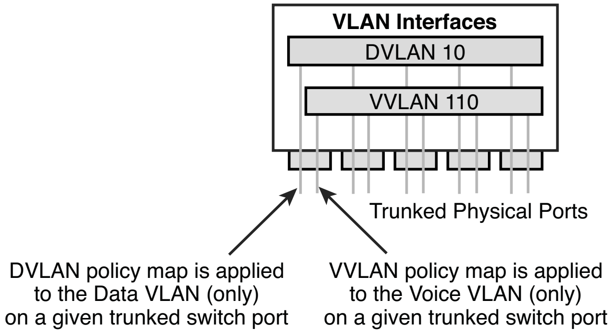

VLAN Interfaces
DVLAN 10
VVLAN 110
Trunked Physical Ports
DVLAN policy map is applied
to the Data VLAN (only)
on a given trunked switch port
VVLAN policy map is applied
to the Voice VLAN (only)
on a given trunked switch port

Figure 13-6 *Per-Port/Per-VLAN QoS*

These port/VLAN-based QoS options allow for efficiency and granularity in policy deployment. For example, marking policies may be more efficiently scaled when applied on a per-VLAN basis. However, policies requiring policing granularity are best performed on a per-port/per-VLAN basis. Specifically, if an administrator wanted to police VoIP traffic from IP phones to a maximum of 128 Kbps from each IP phone, this would best be achieved by deploying a per-port/per-VLAN policing policy applied to the voice VLAN on a given port. A per-port policer would not be sufficient in this case, unless additional classification criteria was provided to specifically identify traffic from the IP phone only; neither could a per-VLAN policy be used, because this would police the aggregate traffic from all ports belonging to the voice VLAN to 128 Kbps.

EtherChannel QoS

Design recommendation:

> Understand the platform-specific requirements of how to apply QoS policies to EtherChannel bundles—specifically, which policy elements (if any) are to be applied to the logical port channel interface and which policy elements are to be applied to the physical member interfaces.

Another case where logical versus physical interfaces has a bearing on QoS design is when provisioning QoS over EtherChannel interfaces. Multiple Gigabit Ethernet or 10 Gigabit Ethernet ports can be logically bundled into a single EtherChannel interface (which is also known as a port channel interface, because this is how it appears in the configuration syntax). From a Layer 2/Layer 3 standpoint, these bundled interfaces are represented (and function) as a single interface.

Keep in mind two important considerations when deploying QoS policies on EtherChannel interfaces:

- The first EtherChannel QoS design consideration relates to load balancing. Depending on the platform, load balancing on the port channel group can be done in various ways: by source IP address, by destination IP address, by source MAC address, by destination MAC address, by source and destination IP address, or by source and destination MAC address. Note that EtherChannel technology does not take into account the bandwidth of each flow. Instead, it relies on the statistical probability that given a large number of flows of relatively equal bandwidths, the load is equally distributed across the links of the port channel group. This might not always be true, however. In general, it is recommended to load balance based on the source and destination IP address, because this allows for statistically superior load distribution. When loads are balanced in this manner, packets belonging to a single flow retain their packet order.
- The second EtherChannel QoS design consideration is that EtherChannel technology does not take into account any QoS configuration on the individual Gigabit Ethernet links. Again, it relies on the statistical probability that given a large number of flows with different QoS markings, the load of those individual flows is equally distributed across the links of the port channel group. Given a failover situation in which one of the links of an EtherChannel group fails, the sessions crossing that link would be reallocated across the remaining links. Because EtherChannel technology has no awareness of QoS markings, it could easily reallocate more real-time flows across any one of the links than the link is configured to accommodate. This could result in degraded real-time services. Incidentally, this scenario could also occur in a nonfailover situation. Therefore, you should use caution when deciding to use EtherChannel technology versus a single higher-speed uplink port.

In addition, be aware that on certain platforms (such as the Catalyst 4500 and 6500), ingress QoS policies must be applied to the logical port channel interface, but on others these are applied to the physical port member interfaces. Egress policies (which are usually queuing policies) are *always* applied to the physical port member interfaces. Table 13-1 summarizes these EtherChannel QoS designs.

Table 13-1 *EtherChannel Logical Versus Physical QoS Policy Requirements by Platform*

Platform	QoS Policies Applied to the (Logical) Port Channel Interface	QoS Policies Applied to the (Physical) Port Member Interfaces
Catalyst 2960/3560/3750		Ingress Egress
Catalyst 4500	Ingress	Egress
Catalyst 6500	Ingress	Egress

Because a slight per-platform variation in EtherChannel QoS configuration exists, a design example is included within each switch platform family.

Campus QoS Models

Design recommendation:

> Follow the four steps of deploying campus QoS.

Generally speaking, deploying QoS in the campus requires four main steps:

1. Globally enable QoS (on MLS QoS-based switches only).
2. Apply an ingress QoS model:
   a. Configure trust or explicitly classify, mark, and police flows.
   b. Configure ingress queuing (if supported and required).
3. Apply an egress QoS model to assign flows to transmit queues and to enable dropping policies.
4. (Optional) Enable control plane policing (on platforms that support this feature).

Ingress QoS Models

Design recommendations:

- Deploy ingress QoS models such as trust, classification and marking/policing on all access-edge ports.
- Deploy ingress queuing (if supported and required).

The ingress QoS model applies either a port trust state or an explicit classification and marking policy to the switch ports (or VLANs, in the case of VLAN-based QoS). Also, this model may include optional ingress policers and ingress queuing (as required and supported).

To design such an ingress QoS model, not only does the end-to-end strategic QoS model (such as the 4-class, 8-class, and 12-class models presented in Chapter 11) need to be taken into account, but you also need to consider which application classes are present at the campus access edge (in the ingress direction). Another consideration is whether these application classes are sourced from trusted or untrusted endpoints.

Note that not every application class from the end-to-end strategic QoS model may be present at the campus access edge (in the ingress direction). Therefore, not every application class needs to be classified, marked, or policed at this node. For example, network control traffic should never be received from endpoints, and therefore this class is not needed at the campus access edge. A similar case can be made for operations, administration, and management (OAM) traffic, because this traffic is primarily generated by network devices and is collected by management stations, which are usually in a data center or a network control center (and not the campus in general). Also, multimedia streaming traffic would likely originate from data center servers and would be unidirectional to campus endpoints (and should not be sourced from campus endpoints). Therefore, classes such as these would likely not be needed to be provisioned for at the campus access edge.

In addition, you must give consideration as to whether application traffic is sourced from trusted versus untrusted endpoints. For example, voice traffic is primarily sourced from Cisco IP phones residing in the voice VLAN (VVLAN) and therefore can be trusted (optimally, by conditional trust polices to facilitate user mobility). However, voice traffic may also be sourced from PC softphone applications, such as Cisco Jabber. However, because such applications share the same UDP port range as multimedia conferencing traffic (specifically, UDP/RTP ports 16384–32767), from a campus access-edge classification standpoint, this renders softphone VoIP streams nearly indistinguishable from multimedia conferencing streams. Unless softphone VoIP packets can be definitively distinguished from multimedia conferencing flows, it is simpler and safer to classify and mark the UDP port range of 16384 to 32767 as multimedia conferencing flows (AF4). The alternative approach is to allow multimedia conferencing flows (marked AF4) to be admitted into strict-priority queue, which might present admission control complications.

Another twist may be presented when a single application class is sourced from more than one VLAN and, therefore, may require multiple class maps to correctly identify it. For example, signaling traffic may be sourced from the VVLAN, but also from the dynamic VLAN (DVLAN) (in the same case of softphone applications). The class map for VVLAN-sourced voice packets may match on CoS 3. However, the class map for DVLAN-sourced signaling traffic may require a TCP/UDP port/port-range.

In addition to explicit marking policies, optional policing policies may also be implemented on the campus access ingress edges to meter and manage flows. For example, voice flows could be policed to 128 Kbps, while remaining traffic from the VVLAN (which would for the most part be signaling traffic, with a negligible amount of management traffic) could be policed to 32 Kbps. Both VVLAN policers could be configured to drop violating flows, because VoIP and signaling traffic is well defined and behaved and traffic bursts in excess of these rates would indicate a network violation or abuse.

Data plane policing policies (as discussed in Chapter 11) can be applied to monitor transactional data, bulk data, and best effort flows, such that these flows are metered, with violations being re-marked to an increased drop precedence within a given AF class (such as AF12, AF22, AF32, or AF42 or even to AF13, AF23, AF33, or AF43 in the case of dual-rate policers). Alternatively, non-AF classes may be policed to be re-marked as scavenger/CS1. What is important is that these packets are not dropped on ingress. For example, each of these classes can be policed to re-mark at 10 Mbps.

Note This data plane policing rate (of 10 Mbps) is an example value. Such values could vary from enterprise to enterprise, even from department to department within an enterprise. *The key is to set data plane policing rates such that approximately 95 percent of traffic flows for a given application class fall below the metered rate*. For the sake of simplicity, a data plane policing rate of 10 Mbps is used for these application classes in these examples.

After all ports have been set to trust or classify and mark (and optionally police) traffic, ingress queuing policies may be applied (on platforms that require and support this feature). Ingress queuing details are discussed in the relevant platform-specific design chapters to follow. Figure 13-7 shows the general campus ingress models.

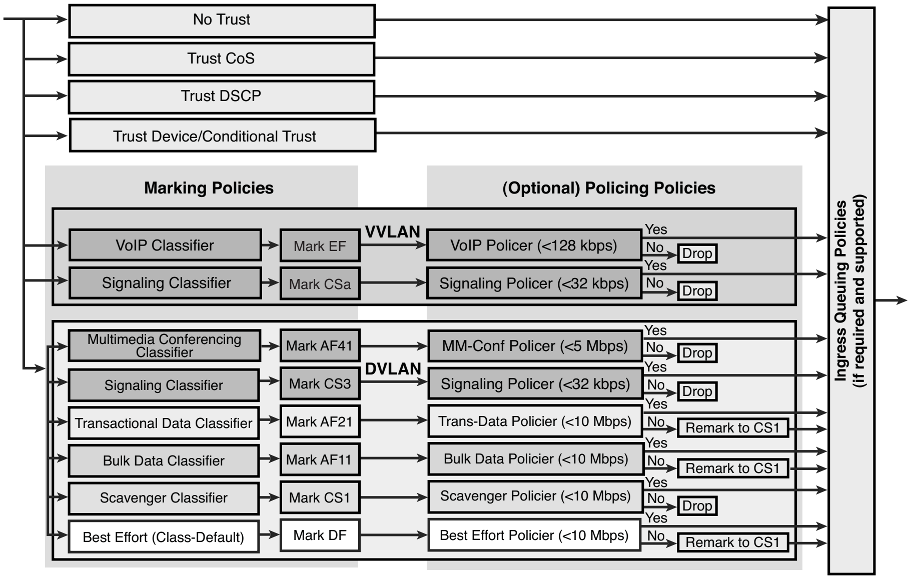

No Trust
Trust CoS
Trust DSCP
Trust Device/Conditional Trust
Marking Policies
(Optional) Policing Policies
VoIP Classifier
Mark EF
VVLAN
VoIP Policer (<128 kbps)
Yes
No
Drop
Signaling Classifier
Mark CSa
Signaling Policer (<32 kbps)
Yes
No
Drop
Multimedia Conferencing Classifier
Mark AF41
MM-Conf Policer (<5 Mbps)
Yes
No
Drop
Signaling Classifier
Mark CS3
DVLAN
Signaling Policer (<32 kbps)
Yes
No
Drop
Transactional Data Classifier
Mark AF21
Trans-Data Policier (<10 Mbps)
Yes
No
Remark to CS1
Bulk Data Classifier
Mark AF11
Bulk Data Policier (<10 Mbps)
Yes
No
Remark to CS1
Scavenger Classifier
Mark CS1
Scavenger Policier (<10 Mbps)
Yes
No
Drop
Best Effort (Class-Default)
Mark DF
Best Effort Policier (<10 Mbps)
Yes
No
Remark to CS1
Ingress Queuing Policies (if required and supported)

Figure 13-7 *Campus Ingress QoS Models*

Egress QoS Models

Design recommendations:

- Deploy egress queuing policies on all switch ports.
- Utilize a minimum 1P3Q*y*T queuing structure.

Cisco Catalyst switches use hardware-based queues. Therefore, the number of queues on a particular port is fixed. Each (nonpriority) queue may also have multiple drop thresholds. Therefore, Catalyst hardware queuing models are typically expressed using the nomenclature 1P*x*Q*y*T, where

- 1P represents a strict-priority queue.
- *x*Q represents *x* number of nonpriority queues.
- *y*T represents *y* number of drop thresholds per nonpriority queue.

As previously discussed (in Chapter 11), at a minimum the following standards-based queuing behaviors should be supported on all platforms when deploying QoS for rich-media applications:

- Real-time queue (to support an RFC 3246 EF PHB service)
- Guaranteed-bandwidth queue (to support RFC 2597 AF PHB services)
- Default queue (to support an RFC 2474 DF service)
- Bandwidth-constrained queue (to support an RFC 3662 Scavenger service)

On Catalyst platforms, these standards-based requirements translate to a minimum hardware queuing structure of 1P3Q*y*T, although more than four hardware-based queues per port is now supported on many Cisco Catalyst switches.

Note In addition, the bandwidth allocations and dropping recommendations outlined in Chapter 11 should also be applied.

Obviously, if more queues are supported, these should be leveraged to give more granular bandwidth guarantees to these respective application classes. Nonetheless, the general application class hierarchy is to provision real-time applications (such as voice, broadcast video, and real-time interactive) in a strict-priority queue, followed by control plane protocols (including network/internetwork control, signaling [which is control plane traffic for the voice/video infrastructure], and network management), followed by guaranteed bandwidth, non-real-time applications (including multimedia conferencing, multimedia streaming, and transactional data), followed by the default best effort, followed by bulk data and scavenger applications. Maintaining such an application class hierarchy serves to ensure consistent per-hop behaviors (PHBs).

Some platforms provide DSCP-to-queue mapping functionality, whereas others (such as some Catalyst 6500 linecards) are limited to CoS-to-queue mapping functionality only. In both cases, it is the value of the internal DSCP that decides the transmit queue to which the packet is assigned. However, in the case of CoS-to-queue mapping, internal DSCP values are assigned to queues in blocks of eight. For example, if CoS 1 is mapped to queue 1 (Q1), internal DSCP values 8 through 15 are assigned to Q1; if CoS 2 is assigned to queue 2, internal DSCP values 16 through 23 are assigned to Q2; if CoS 3 is mapped to queue 3, internal DSCP values 24 through 31 are assigned to Q3 (and so on). Essentially, CoS-to-queue mapping assigns the internal DSCP value that corresponds to the (CoS value * 8), along with the following seven internal DSCP values, to a given queue.

Finally, on some platforms, not only bandwidth allocations may be tuned but also buffer allocations. Per-queue buffer allocations can be *directly proportional* to per-queue bandwidth allocations (for example, the buffer allocations for the best effort queue may be set to 25 percent to match the bandwidth allocation for this queue) or these can be *indirectly proportional* (for example, a strict-priority queue which is being serviced in real-time would likely not need a corresponding 33 percent buffer allocation; whereas a bandwidth-constrained queue would benefit from deeper buffers to offset its minimal bandwidth allocation). Tuning buffer allocations is less impactful than tuning bandwidth allocations alone, but serves to complement the scheduling policies. Therefore, in the design chapters that follow (wherever possible) the strict-priority and less-than-best-effort queues are tuned to be indirectly proportional to their bandwidth allocations, while all other nonpriority preferential queues are tuned to be directly proportional to their bandwidth allocations.

Because Catalyst queuing models vary by hardware, there is no generic one-size-fits-all model, and therefore strategic 4-, 8-, and 12-class queuing models for each platform are discussed on a per-platform basis in the subsequent design chapters.

Campus Port QoS Roles

Design recommendation:

> Define consistent ingress and egress QoS policies for all switch port types, including ports connecting to
>
> - Untrusted endpoints
> - Trusted endpoints
> - Conditionally trusted endpoints
> - Network infrastructure components

The QoS policy elements discussed thus far can be grouped into roles that various switch ports (or interfaces) serve within the campus architecture, including the following:

- **Switch ports connecting to untrusted endpoints**
  - Endpoint examples include (unsecured/unmanaged) PCs, printers, or other devices.
  - Trust should be disabled on these ports.
  - Optional ingress marking or policing policies (such as data plane policing policies) may be configured on these ports.
  - Ingress queuing policies (if supported and if required due to oversubscription scenarios, such as switch stacks) may be configured on these ports.
  - Egress queuing policies that support (at a minimum) 1P3QyT queuing should be configured on these ports, preferably with DSCP-to-queue mapping.
- **Switch ports connecting to trusted endpoints**
  - Endpoint examples include secure/centrally managed PCs and servers, IP video surveillance (IPVS) units, IP conferencing stations, analog and video-conferencing gateways, and similar other devices.
  - Static trust policies should be configured on these ports, preferably DSCP trust for maximum classification and marking granularity.
  - Optional ingress marking or policing policies (such as data plane policing policies) may be configured on these ports.
  - Ingress queuing policies (if supported and if required due to oversubscription scenarios, such as switch stacks) may be configured on these ports.
  - Egress queuing policies that support (at a minimum) 1P3QyT queuing should be configured on these ports, preferably with DSCP-to-queue mapping.
- **Switch ports connecting to conditionally trusted endpoints**
  - Endpoint examples include Cisco IP phones, Cisco TelePresence Systems, Cisco IP video surveillance cameras and Cisco Digital Media Players.
  - Conditional trust policies should be configured on these ports, preferably in conjunction with DSCP trust extension, for maximum classification and marking granularity.
  - Optional ingress marking or policing policies (such as data plane policing policies) may be configured on these ports.
  - Ingress queuing policies (if supported and if required due to oversubscription scenarios, such as switch stacks) may be configured on these ports.
  - Egress queuing policies that support (at a minimum) 1P3QyT queuing should be configured on these ports, preferably with DSCP-to-queue mapping.

- **Switch ports connecting to network infrastructure**
  - Examples include wireless access points, access/distribution uplinks/downlinks, distribution/core uplinks/downlinks, core links, campus-to-WAN/VPN-edge links.
  - Static trust policies should be configured on these ports, preferably DSCP trust for maximum classification and marking granularity.
  - Optional ingress marking or policing policies (such as data plane policing policies) may be configured on these ports.
  - Egress queuing policies that support (at a minimum) 1P3QyT queuing should be configured on these ports, preferably with DSCP-to-queue mapping. However, switch platforms and linecards that support 1P7QyT queuing are preferred at the distribution and core layers for increased queuing granularity at these aggregation layers.

Figure 13-8 illustrates these campus port roles.

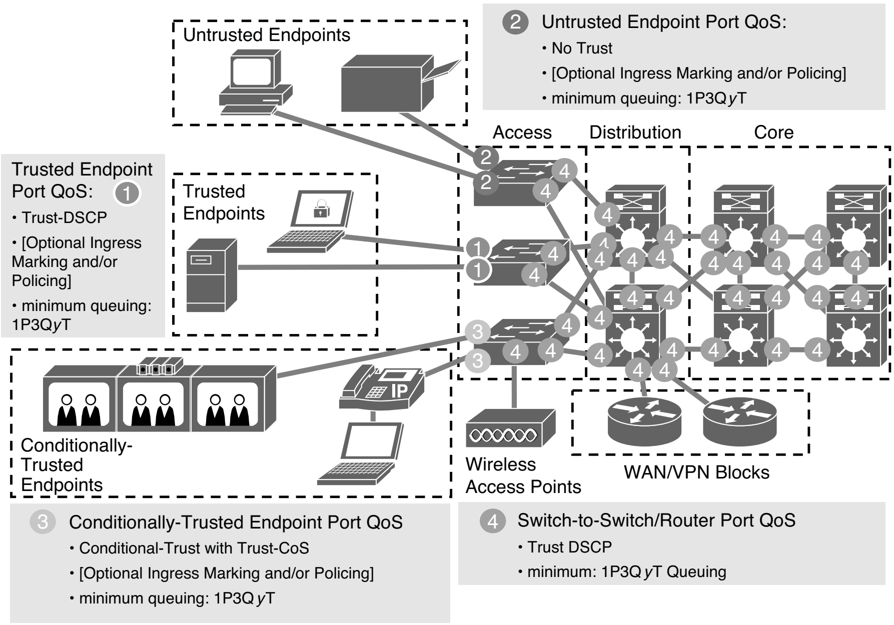

Untrusted Endpoints
2 Untrusted Endpoint Port QoS:
• No Trust
• [Optional Ingress Marking and/or Policing]
• minimum queuing: 1P3QyT
Access
Distribution
Core
Trusted Endpoint Port QoS: 1
• Trust-DSCP
• [Optional Ingress Marking and/or Policing]
• minimum queuing: 1P3QyT
Trusted Endpoints
IP
Conditionally-Trusted Endpoints
Wireless Access Points
WAN/VPN Blocks
3 Conditionally-Trusted Endpoint Port QoS
• Conditional-Trust with Trust-CoS
• [Optional Ingress Marking and/or Policing]
• minimum queuing: 1P3QyT
4 Switch-to-Switch/Router Port QoS
• Trust DSCP
• minimum: 1P3QyT Queuing

Figure 13-8 *Campus Port QoS Roles*

Campus AutoQoS

Design recommendations:

- Understand the benefits and limitations of the AutoQoS feature.
- When considering using the AutoQoS feature, keep in mind that it can be leveraged as a design template that can be modified and tailored.

Due to the complexity of some QoS policies, coupled with the large number of ports on typical Catalyst switches, QoS deployment can often become unwieldy. One option is to make liberal use of the **interface-range** configuration command to deploy policies to multiple interfaces at once. Another option is to use Automatic QoS (AutoQoS), if applicable.

To address customer demands for simplification of QoS deployment, Cisco developed the AutoQoS feature. AutoQoS is a macro that enables you to enter one or two simple AutoQoS commands to enable all the recommended QoS settings for one (or more) applications. In its first release, AutoQoS-VoIP, AutoQoS provisioned all the recommended QoS settings for IP telephony deployments for a specific switch port interface.

AutoQoS-VoIP for Catalyst switches supports three modes of operation, all of which are preceded by the **auto qos voip** interface configuration command:

- **cisco-phone:** This mode is intended for switch ports that may be connected to PCs or Cisco IP phones and sets the port to a conditional trust state, and configures mapping and queuing policies for QoS for VoIP.
- **cisco-softphone:** This mode is intended for switch ports that may be connected to PCs running Cisco IP Communicator or similar softphone software, and polices VoIP and signaling traffic and configures mapping and queuing policies for QoS for VoIP.
- **trust:** This mode is intended for switch ports that are within the trusted boundary (such as interswitch links, including uplinks and downlinks) or switch ports that are connecting to trusted endpoints, and sets the port to a static **trust-dscp** state and configures mapping and queuing policies for QoS for VoIP.

Note For additional details on AutoQoS-VoIP and the platform-specific commands and settings that it generates, see the respective platform's AutoQoS-VoIP documentation (see the platform-specific links at the end of each respective design chapter).

Some may naturally ask, why should I read these lengthy and complex QoS design chapters when I have AutoQoS?

First of all, keep in mind that AutoQoS-VoIP is an excellent tool for customers who want to enable QoS for VoIP (only), that have basic QoS needs, or do not have the time or desire to do more with QoS.

It is important to remember how AutoQoS developed. AutoQoS features are the result of Cisco QoS feature development coupled with Cisco QoS design guides based on large-scale lab testing. AutoQoS-VoIP is the product of the first *QoS Solution Reference Network Design (SRND) Guide* (published in 1999), and the AutoQoS-VoIP feature has not been significantly updated since. Therefore, if the business requirement for QoS were for IP telephony only, AutoQoS would be an excellent tool to expedite the QoS deployment.

However, in August 2010, an updated version of AutoQoS was released for the Catalyst 2960-G/S, 2975-GS, 3560-G/E/X, and 3750-G/E/X family of switches (with IOS Release 12.2(55)SE). This release was directly based on the recommendations put forward in the latest Enterprise QoS SRND (release 4.0) to support rich-media applications. In fact, the global configuration command for this version of AutoQoS is **auto qos srnd4**. Appendix A, "AutoQoS for Medianet," discusses the functionality and options of this latest AutoQoS feature and the details the QoS configurations that it automatically generates.

When considering AutoQoS, note that it does not have to be considered as an all-or-nothing tool. Rather, AutoQoS could be viewed as a starting template, which can be modified and tailored to specific needs and requirements, all the while expediting deployment and reducing the potential of human error (in the form of configuration typos).

A final consideration relating to AutoQoS is that these policies can also be deployed via Auto SmartPorts. Auto SmartPorts is a macro that will automatically configure switch port parameters and best-practice policies according to (CDP or LLDP discovered) device type, including Cisco IP phones, Digital Media Players, IP video surveillance cameras, access points, and even other Cisco switches. SmartPorts macros go well beyond QoS and configure security and other features as well. SmartPorts invokes AutoQoS for the QoS component of a port's configuration—which is essentially a macro (AutoQoS) being triggered within a macro (SmartPorts).

Control Plane Policing

Design recommendation:

> Understand the purpose and use of the control plane policing feature as an optional QoS-for-security tool to harden the network infrastructure.

Control plane policing (CoPP) is a security infrastructure feature available on Catalyst 4500 and 6500 series switches running Cisco IOS that allows the configuration of QoS policies that rate limit the traffic handled by the main CPU of the switch. This protects the control plane of the switch from direct denial-of-service (DoS) attacks, reconnaissance activity, or other network errors that may flood unwanted control traffic to the CPU (such as Internet Control Message Protocol [ICMP] error messages and so on).

CoPP protects switches by allowing the definition and enforcement of QoS policies that regulate the traffic processed by the main switch CPU (route or switch processor). With CoPP, these QoS policies are configured to permit, block, or rate limit the packets handled by the main CPU.

Packets handled by the main CPU, referred to as control plane traffic, typically include the following:

- Routing protocols.
- Packets destined to the local IP address of the router.

- Packets from network management protocols, such as Simple Network Messaging Protocol (SNMP).
- Interactive access protocols such as Secure Shell (SSH) and Telnet.
- Other protocols, such as ICMP, or IP options might also require handling by the switch CPU.
- Layer 2 packets such as bridge protocol data unit (BPDU), Cisco Discovery Protocol (CDP), DOT1X, and so on.

CoPP leverages MQC for its QoS policy configuration. MQC allows definition of class maps to divide the traffic destined to the CPU into multiple classes based on different criteria.

After the traffic classes are defined, a QoS policy map can be defined such that each class can be configured to permit all packets, drop all packets, or drop only those packets exceeding a specific rate limit.

Finally, the control plane is presented as a virtual interface within the configuration so that MQC service policy statement can be attached to it.

Additional details and design recommendations for control plane policing are discussed in Appendix B, “Control Plane Policing.”

Summary

This chapter discussed the considerations and recommendations for campus QoS designs. The primary role of QoS in rich-media campus networks was identified as managing packet loss due to instantaneous buffer overruns—which can occur within a few dozen milliseconds at GE/10GE campus network speeds. The secondary role of campus QoS was also called out as differentiating and managing application traffic at the access edge.

Several strategic QoS design principles that apply in the campus were also reviewed, including the following:

- Always perform QoS in hardware rather than software when a choice exists.
- Classify and mark applications as close to their sources as technically and administratively feasible.
- Police unwanted traffic flows as close to their sources as possible.
- Enable queuing policies at every node where the potential for congestion exists.
- (Optional) Protect the control plane and data plane by enabling control plane policing.

Following this, several campus-specific design considerations and recommendations were discussed, beginning with the syntactical and functional differences between MLS-based QoS and MQC-based QoS.

Next, the internal DSCP concept was presented, and how this value is derived by examining the operation of static and dynamic port trust states. Trust boundaries were then discussed, along with the need to separate endpoints into three main trust categories (untrusted, trusted, or conditionally trusted).

Following the trust discussion, an extended discussion of physical interface QoS versus logical interface QoS was presented. Port-based QoS was reviewed and shown to be the simplest and most modular QoS design option. VLAN-based QoS was then overviewed and illustrated as to how it can scale certain policies (such as trust, classification, and marking), but not others (such as aggregate policing). Also per-port/per-VLAN QoS was described as a tool for very fine and granular policy control. A similar physical versus logical discussion was then applied to EtherChannel bundles to identify—on a platform-by-platform basis—which QoS policy elements (if any) are to be applied to a logical port channel interface and which policy elements are to be applied to the physical member interfaces.

A four-step model for deploying QoS in the campus was then presented:

1. Globally enable QoS (on MLS QoS-based switches only).
2. Apply an ingress QoS model:
   a. Configure trust or explicitly classify, mark, and police flows.
   b. Configure ingress queuing (if supported and required).
3. Apply an egress QoS model to assign flows to transmit queues and to enable dropping policies.
4. (Optional) Enable control plane policing (on platforms that support this feature).

Following this, detailed ingress and egress port-QoS roles were defined for switch ports connecting to

- Untrusted endpoints
- Trusted endpoints
- Conditionally trusted endpoints
- Network infrastructure components.

Following this, the AutoQoS feature was overviewed. AutoQoS-VoIP was shown to be an effective tool for expediting QoS deployment for IP telephony deployments, while AutoQoS-SRND4 was introduced as a design option for rich-media campus requirements. Further, it was pointed out that AutoQoS was not an all-or-nothing design option, but rather could be leveraged as a design template that can be modified and tailored.

Finally, control plane policing was briefly discussed as an optional QoS-for-security tool to harden the network infrastructure.

Additional Reading

Cisco Enterprise Medianet Quality of Service Design 4.0—Overview: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSIntro_40.html

Cisco Enterprise Medianet Campus QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html

Medianet QoS Design Strategy At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qosmrn.pdf

Medianet Campus QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampusaag.html

Medianet Campus AutoQoS At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/autoqosmediacampus.pdf

Chapter 14

Campus Access (Cisco Catalyst 3750) QoS Design

The QoS role of the campus access switch is to differentiate and manage traffic at the edge and to manage packet loss. Figure 14-1 illustrates the port-specific quality of service (QoS) roles of a campus access switch.

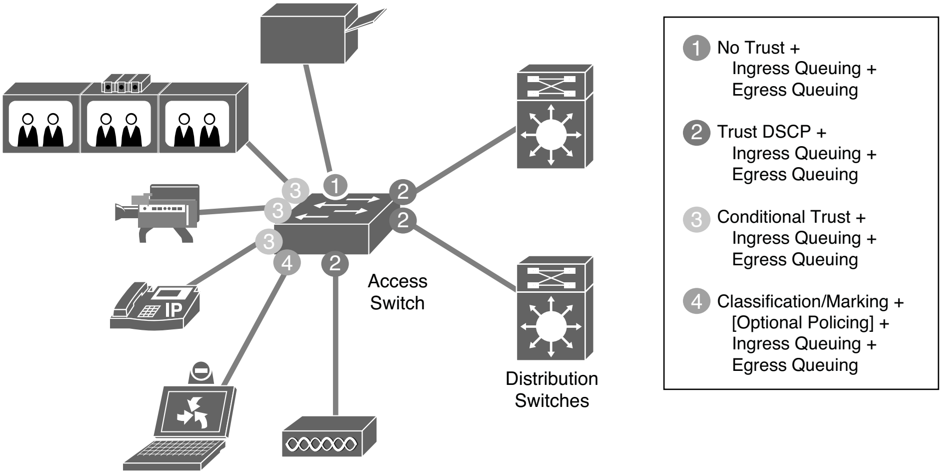

3
1
2
3
2
3
4
2
IP
Access Switch
Distribution Switches
1 No Trust + Ingress Queuing + Egress Queuing
2 Trust DSCP + Ingress Queuing + Egress Queuing
3 Conditional Trust + Ingress Queuing + Egress Queuing
4 Classification/Marking + [Optional Policing] + Ingress Queuing + Egress Queuing

Figure 14-1 *Campus Access Switch Port QoS Roles*

In the design chapters of this book (of which this is the first), only one platform will be chosen to represent a specific place in the network (PIN). This is simply to reduce the number of configuration permutations and to help bound this book in scope (and page count). For example, this design chapter focuses on the Cisco Catalyst 3750, which is well suited to perform the role of a campus access switch. Of course, this is not to say that it is the only platform to suit this role, nor that this is the only role that it can play in a network—just one that it is well-matched to.

To begin, let's review this platform's architecture, specifically as it relates to QoS.

Cisco Catalyst 3750 QoS Architecture

From a QoS perspective, the Cisco Catalyst 3750-X (hereafter referred to simply as the Catalyst 3750) is nearly identical to several other Catalyst 2K/3K platforms, including the Catalyst 2960-S and the 3650-X. Therefore, for the most part, the configurations presented in this design chapter will be equally applicable to these other platforms. However, a few key differences do exist between these platforms, as summarized in Table 14-1.

Table 14-1 *Cisco Catalyst 2960-S, 3560-G/E/X, and 3750-G/E/X: Major Feature and Functionality Matrix*

Switch	Layer 2-Only Switch	Layer 2/Layer 3 Multilayer Switch	Stackable?	Stacking Technology	Total Switching Capacity
Catalyst 2960-G	Yes				20 Gbps
Catalyst 2960-S	Yes		Some models	FlexStack	20 Gbps
Catalyst 3560-G		Yes			32 Gbps
Catalyst 3560-E		Yes			64 Gbps
Catalyst 3560-X		Yes			64 Gbps
Catalyst 3750-G		Yes	Yes	StackWise	32 Gbps
Catalyst 3750-E		Yes	Yes	StackWise Plus	64 Gbps
Catalyst 3750-X		Yes	Yes	StackWise Plus	64 Gbps

Figure 14-2 illustrates the QoS architecture for this platform.

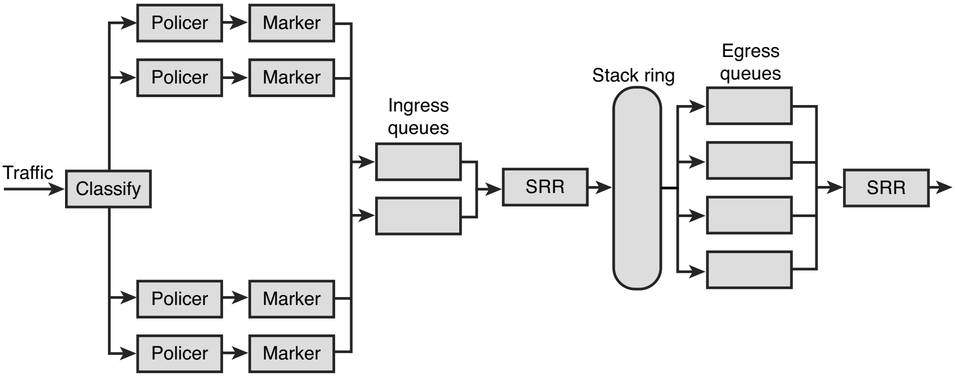


Figure 14-2 *Cisco Catalyst 3750 QoS Architectural Model*

Traffic is classified on ingress, based on trust states or class maps. Marking or policing policies can be applied to physical switch ports or (in the case of multilayer switch platforms) to switch virtual interfaces (SVIs), which allows for per-VLAN or per-port/per-VLAN policies.

Because the total inbound bandwidth of all ports can exceed the bandwidth of the stack or internal ring, ingress queues are located after the packet is classified, policed, and marked and before packets are forwarded into the switch fabric (that is, the internal or stack rings). Also, because multiple ingress ports can simultaneously send packets to an egress port (such as an uplink port) and cause congestion, outbound queues are located after the stack or internal rings. The queuing scheduler is shared round-robin (SRR), and the dropping algorithm is weighted tail drop (WTD), both of which are discussed in more detail later in this chapter.

Relating to QoS, these key switch-specific differences exist:

- The Catalyst 2960-G/S does not support multilayer switching and therefore does not correspondingly support per-VLAN or per-port/per-VLAN policies.
- The Catalyst 2960-G can police only to a minimum rate of 1 Mbps. All other platforms within this switch product family can police to a minimum rate of 8 Kbps (with the exception of the 2960-S, which, although it can be configured to police at 8 Kbps, can only police at a minimum rate of 16 Kbps).
- The Catalyst 2960-S does not support ingress queuing.
- The Catalyst 2960-S does not support a class-default class map.
- Only the Catalyst 3650-E/X and 3750-X support IPv6 QoS.
- Only the Catalyst 3650-E/X and 3750-X support policing on 10 Gigabit Ethernet interfaces.
- Only the Catalyst 3650-E/X and 3750-X support SRR shaping weights on 10 Gigabit Ethernet interfaces.

QoS Design Steps

There are four main steps to configure QoS on Cisco Catalyst 3750 series switches:

1. Enable QoS.
2. Configure ingress QoS models, including the following:
    - Trust DSCP Models
    - Conditional Trust Models
    - Service Policy Models
3. Configure ingress queuing.

4. Configure egress queuing.

Each of these design steps is covered in turn.

Enabling QoS

Because the Catalyst 3750 is an Multi-Layer Switch (MLS)-based QoS platform, QoS needs to be explicitly enabled globally, as it is disabled by default. This is a critical first step to deploying QoS on these platforms. If this small (but important) step is overlooked, it can lead to frustration in troubleshooting QoS problems; this is because the switch software accepts QoS commands and even displays these within the switch configuration, but none of the QoS commands become active until after the **mls qos** global command is enabled as demonstrated here:

```
C3750(config)# mls qos
```

You can verify this configuration with the **show mls qos** command as demonstrated in Example 14-1.

Note Only one example of a given verification command is usually presented (to minimize redundancy).

Example 14-1 *Verifying Global QoS on a Catalyst 3750:* ***show mls qos***

```
C3750# show mls qos
QoS is enabled
QoS ip packet dscp rewrite is enabled
```

Ingress QoS Models

Ingress QoS models include the following:

- Trust
- Classification and marking
- Optional policing
- Ingress queuing

Each of these ingress QoS models is discussed in turn.

Trust Models

There are four main trust states that a port can be configured with:

- Untrusted state
- Trust CoS state
- Trust DSCP state
- Conditional trust state

Of these trust states, all are static except the conditional trust state (which is dynamic). Each of these trust states is covered in turn.

Untrusted Model

After MLS QoS has been globally enabled on the Catalyst 3750, all ports will be automatically set to a default untrusted state. Therefore, no explicit command is needed to set a port to the untrusted state. However, if a port has previously been configured to one of the other trust states and is to be reverted to an untrusted state, the **no mls qos trust** interface command will be required.

Trust CoS Model

A Catalyst 3750 switch port can be configured to trust CoS by configuring the interface with the **mls qos trust cos** command. However, if an interface is set to trust CoS, it by default calculates a packet's internal differentiated services code point (DSCP) to be the incoming packet's (CoS value * 8). Although this might suitable for many markings, this default mapping might not be suitable for Voice over IP (VoIP), because VoIP is usually marked CoS 5, which would map by default to DSCP 40 (and not 46, which is the EF PHB as defined by RFC 3246). Therefore, if an interface is set to trust CoS, the default CoS-to-DSCP mapping table should be modified such that CoS 5 maps to DSCP 46, as shown in Example 14-2.

Example 14-2 *Configuring Trust CoS and CoS-to-DSCP Mapping Modification on a Catalyst 3750*

```
C3750(config)# mls qos map cos-dscp 0 8 16 24 32 46 48 56
 ! CoS 5 (the sixth CoS value, starting from 0) is mapped to 46
C3750(config)# interface GigabitEthernet 1/0/1
C3750(config-if)# mls qos trust cos
 ! The interface is set to statically trust CoS
```

You can verify the configuration in Example 14-2 with the following commands:

- **show mls qos map cos-dscp** (as shown in Example 14-3)
- **show mls qos interface** (as shown in Example 14-4)

Example 14-3 *Verifying Global CoS-to-DSCP Mapping Modifications on a Catalyst 3750:* ***show mls qos map cos-dscp***

```
C3750# show mls qos map cos-dscp
   Cos-dscp map:
        cos:   0  1  2  3  4  5  6  7
     --------------------------------
       dscp:   0  8 16 24 32 46 48 56
```

In Example 14-3, the CoS-to-DSCP mapping value for CoS 5 has been modified from the default mapping of 40 (CoS 5 * 8) to 46 (to match the recommendation from RFC 3246 that real-time applications be marked DSCP 46/EF).

Example 14-4 *Verifying Interface Trust Settings on a Catalyst 3750:* ***show mls qos interface***

```
C3750# show mls qos interface GigabitEthernet 1/0/1
GigabitEthernet1/0/1
trust state: trust cos
trust mode: trust cos
trust enabled flag: ena
COS override: dis
default COS: 0
DSCP Mutation Map: Default DSCP Mutation Map
Trust device: none
qos mode: port-based
```

Trust DSCP Model

Because of the additional granularity of DSCP versus QoS markings, it is generally recommended to trust DSCP rather than CoS (everything else being held equal). A Catalyst 3750 switch port can be configured to trust DSCP with the **mls qos trust dscp** interface command, as shown in Example 14-5.

Example 14-5 *Configuring Trust-DSCP on a Catalyst 3750-X*

```
C3750(config)# interface GigabitEthernet 1/0/1
C3750(config-if)# mls qos trust dscp
 ! The interface is set to statically trust DSCP
```

You can verify the configuration can in Example 14-5 with the **show mls qos interface** command.

Conditional Trust Models

In addition to configuring switch ports to statically trust endpoints, the Catalyst 3750 family also supports dynamic, conditional trust with the **mls qos trust device** interface command. This command supports the following keyword options:

- **cisco-phone** to conditionally trust Cisco IP phones
- **cts** to conditionally trust Cisco TelePresence Systems
- **ip-camera** to conditionally trust Cisco IP video surveillance cameras
- **media-player** to conditionally trust Cisco Digital Media Players

In addition to the type of device to be trusted, the type of trust to be extended may also be specified (either CoS or DSCP). For example, when configuring conditional trust to Cisco IP phones, it is recommended to dynamically extend CoS trust, because Cisco IP phones can only re-mark PC QoS markings at Layer 2 (CoS) and not at Layer 3 (DSCP). For other endpoints that do not have this re-marking limitation, it is recommended to dynamically extend DSCP trust (over CoS trust), not only because DSCP has greater marking granularity but also because the type of trust configured on the ingress switch port on a Catalyst 3750 switch ultimately determines the type of queuing policies that are applied on the egress switch port. Specifically, if an ingress switch port is configured to trust CoS—whether this is configured statically or dynamically (in conjunction with the **mls qos trust device** interface command)—a CoS-to-queue mapping determines the ingress and egress queuing policies. Conversely, if an ingress switch port is configured to trust-DSCP—whether this is configured statically or dynamically—a DSCP-to-queue mapping determines the ingress and egress queuing policies. Because DSCP-to-queue mapping has more granular policy options, it is the preferred way to assign packets to queues and therefore depends on the ingress switch port being set to trust DSCP.

An example of a dynamic, conditional trust policy that is set to extend CoS trust to CDP-verified Cisco IP phones is shown in Example 14-6.

Example 14-6 *Configuring (CoS Mode) Conditional Trust to a Cisco IP Phone on a Catalyst 3750*

```
C3750(config)# mls qos map cos-dscp 0 8 16 24 32 46 48 56
 ! CoS 5 (the sixth CoS value, starting from 0) is mapped to 46
C3750(config)# interface GigabitEthernet 1/0/1
C3750(config-if)# switchport access vlan 10
C3750(config-if)# switchport voice vlan 110
C3750(config-if)# spanning-tree portfast
C3750(config-if)# mls qos trust device cisco-phone
 ! The interface is set to conditionally trust Cisco IP phones
C3750(config-if)# mls qos trust cos
 ! CoS trust will be dynamically extended to Cisco IP phones
```

You can verify this configuration with the **show mls qos interface** command.

Classification and Marking Models

In many scenarios, trust models may not be available or sufficient to distinctly classify all types of traffic required by the end-to-end QoS strategic model. Therefore, explicit classification and marking policies may be needed at the access edge.

Even on MLS QoS-based switches, an MQC approach using class maps and policy maps is utilized for classification and marking policies, which allows for a nearly infinite variation of policy options. To simplify, Example 14-7 presents a configuration example based on Figure 11-5 (an eight-class QoS Model) from Chapter 11, "QoS Design Principles and Strategies."

Note As discussed in the previous chapter, not all application classes may be present at the access edge on ingress. For example, streaming video would likely not be present at the access edge on ingress (because these flows are not *sourced* from campus endpoints, but are likely *destined* to them), nor would network control flows be sourced from campus endpoints. Therefore, these classes would not need to be included in the access-edge classification and marking policy map.

Note Referenced access lists are omitted from the policy examples for brevity.

Example 14-7 *Classification and Marking Policy Example on a Catalyst 3750*

```
! This section configures the class maps
C3750(config-cmap)# class-map match-all VOICE
C3750(config-cmap)#  match ip dscp ef
 ! Voice is matched on DSCP EF
C3750(config-cmap)# class-map match-all INTERACTIVE-VIDEO
C3750(config-cmap)#  match access-group name INTERACTIVE-VIDEO
 ! Associates interactive video access-list with class map
C3750(config-cmap)# class-map match-all SIGNALING
C3750(config-cmap)#  match ip dscp cs3
 ! Signaling is matched on DSCP CS3
C3750(config-cmap)# class-map match-all TRANSACTIONAL-DATA
C3750(config-cmap)#  match access-group name TRANSACTIONAL-DATA
 ! Associates transactional data access-list with class map
C3750(config-cmap)# class-map match-all SCAVENGER
C3750(config-cmap)#  match access-group name SCAVENGER
 ! Associates scavenger access-list with class map
C3750(config-cmap)# class-map match-all DEFAULT
```

```
C3750(config-cmap)#  match access-group name DEFAULT
 ! Associates default access-list with class map

 ! This section configures the per-port ingress marking policy map
C3750(config-cmap)# policy-map PER-PORT-MARKING
C3750(config-pmap)# class VOICE
C3750(config-pmap-c)#  set dscp ef
 ! VoIP is marked EF (see note below)
C3750(config-pmap-c)# class SIGNALING
C3750(config-pmap-c)#  set dscp cs3
 ! Signaling (from the VVLAN) is marked CS3 (see note below)
C3750(config-pmap-c)# class INTERACTIVE-VIDEO
C3750(config-pmap-c)#  set dscp af41
 ! Interactive video is marked AF41
C3750(config-pmap-c)# class TRANSACTIONAL-DATA
C3750(config-pmap-c)#  set dscp af21
 ! Transactional data is marked AF21
C3750(config-pmap-c)# class SCAVENGER
C3750(config-pmap-c)#  set dscp cs1
 ! Scavenger traffic is marked CS1
C3750(config-pmap-c)# class DEFAULT
C3750(config-pmap-c)#  set dscp default
 ! An explicit class-default marks all other IP traffic to 0 (see note)

 ! This section attaches the service-policy to the interface(s)
C3750(config)# interface range GigabitEthernet 1/0/1-48
C3750(config-if-range)# switchport access vlan 10
C3750(config-if-range)# switchport voice vlan 110
C3750(config-if-range)# spanning-tree portfast
C3750(config-if-range)# service-policy input PER-PORT-MARKING
 ! Attaches the per-port marking policy to the interface(s)
```

Note While the Catalyst 3750 IOS Software includes an implicit class-default class, any policy actions assigned to this class are not enforced. Therefore, an explicit class DEFAULT is configured in Example 14-7 to enforce a marking/re-marking policy to DSCP 0 for all other IP traffic.

Note An explicit marking command (**set dscp**) is used even for trusted application classes (like Voice and Signaling) rather than a trust policy map action. This is because a **trust** statement in a policy map requires multiple hardware entries. The use of an explicit (but seemingly redundant) explicit marking command actually improves the policy efficiency from a hardware perspective.

Note Classification and marking models may be deployed on a per-port basis or on a per-VLAN basis (as discussed later in this chapter).

You can verify the configuration in Example 14-7 with the following commands:

- **show mls qos interface**
- **show class-map**
- **show policy-map**
- **show policy-map interface**

Note Other than the **show mls qos interface** verification command (which has already been presented in Example 14-5), these other verification commands are static and simply repeat back what has been configured without providing any dynamic verification information. Even the **show policy-map interface** command on the Catalyst 3750 does not show dynamic packet counts, byte counts, drops, and bits per second (bps) rates (like the corresponding command on other devices does). Therefore, there is little to be gleaned from including these verification command examples (other than parroting back a portion of the configuration), and therefore these are omitted for the sake of brevity.

Classification, Marking, and Policing Models

Optionally, policing policies can also be effectively deployed at the access edge of the campus to manage application traffic and to drop unwanted traffic.

In Example 14-8, a 12-class strategic policy (based on Figure 11-7) is being adapted to the campus access edge (where application classes that are not present, such as Network Control and Streaming Video, are pruned from the policy; also a distinction is made for some flows sourced from the voice VLAN (VVLAN) versus the dynamic VLAN (DVLAN). Specifically, in this example, VoIP and signaling traffic from the VVLAN can be policed to drop at 128 Kbps and 32 Kbps, respectively (because any excessive traffic matching this criteria would indicate of network abuse). Similarly, the multimedia conferencing, signaling, and scavenger traffic from the DVLAN can be policed to drop. However, data plane policing policies can be applied to transactional, bulk, and best effort data traffic, such that these flows are subject to being re-marked (but not dropped at the ingress edge) when severely out of profile. Re-marking is performed by configuring a policed DSCP map with the global configuration command **mls qos map policed-dscp**, which specifies which DSCP values are subject to re-marking if out-of-profile and what value these should be re-marked to (which in the case of data plane policing/Scavenger class QoS policies, this value is CS1/DSCP 8). Incidentally, the policy logic of this example was previously presented in Figure 13-8 in Chapter 13, “Campus QoS Design Considerations and Recommendations.”

Note Referenced class maps are omitted from the policy example for brevity and to minimize redundancy.

Example 14-8 *Classification, Marking, and Policing Policy Example on a Catalyst 3750*

```
! This section configures the global policed-DSCP markdown map
C3750(config)# mls qos map policed-dscp 0 10 18 to 8
 ! DSCP 0 (DF), 10 (AF11) and 18 (AF21) are marked down to 8 (CS1)
 ! if found to be in excess of their (respective) policing rates

! This section configures the per-port policing policy map
C3750(config)# policy-map PER-PORT-POLICING
C3750(config-pmap)# class VVLAN-VOIP
C3750(config-pmap-c)#  set dscp ef
C3750(config-pmap-c)#  police 128k 8000 exceed-action drop
 ! VoIP is marked EF and policed to drop at 128 Kbps
C3750(config-pmap-c)# class VVLAN-SIGNALING
C3750(config-pmap-c)#  set dscp cs3
C3750(config-pmap-c)#  police 32k 8000 exceed-action drop
 ! (VVLAN) Signaling is marked CS3 and policed to drop at 32 Kbps
C3750(config-pmap-c)# class MULTIMEDIA-CONFERENCING
C3750(config-pmap-c)#  set dscp af41
C3750(config-pmap-c)#  police 5m 8000 exceed-action drop
 ! Multimedia conferencing is marked AF41 and policed to drop at 5 Mbps
C3750(config-pmap-c)# class SIGNALING
C3750(config-pmap-c)#  set dscp cs3
C3750(config-pmap-c)#  police 32k 8000 exceed-action drop
 ! (DVLAN) Signaling is marked CS3 and policed to drop at 32 Kbps
C3750(config-pmap-c)# class TRANSACTIONAL-DATA
C3750(config-pmap-c)#  set dscp af21
C3750(config-pmap-c)#  police 10m 8000 exceed-action policed-dscp-transmit
 ! Trans-data is marked AF21 and policed to re-mark (to CS1) at 10 Mbps
C3750(config-pmap-c)# class BULK-DATA
C3750(config-pmap-c)#  set dscp af11
C3750(config-pmap-c)#  police 10m 8000 exceed-action policed-dscp-transmit
 ! Bulk data is marked AF11 and policed to re-mark (to CS1) at 10 Mbps
C3750(config-pmap-c)# class SCAVENGER
C3750(config-pmap-c)#  set dscp cs1
C3750(config-pmap-c)#  police 10m 8000 exceed-action drop
  ! Scavenger traffic is marked CS1 and policed to drop at 10 Mbps
```

```
C3750(config-pmap-c)# class DEFAULT
C3750(config-pmap-c)#  set dscp default
C3750(config-pmap-c)#  police 10m 8000 exceed-action policed-dscp-transmit
 ! An explicit default class marks all other IP traffic to DF
 ! and polices all other IP traffic to re-mark (to CS1) at 10 Mbps

 ! This section attaches the service policy to the interface(s)
C3750(config)# interface range GigabitEthernet 1/0/1-48
C3750(config-if-range)# switchport access vlan 10
C3750(config-if-range)# switchport voice vlan 110
C3750(config-if-range)# spanning-tree portfast
C3750(config-if-range)# service-policy input PER-PORT-POLICING
 ! Attaches the per-port policing policy to the interface(s)
```

Note The Catalyst 3750 IOS Software allows for policing rates to be entered using the postfixes **k** (for kilobits), **m** (for megabits), and **g** (for gigabits), as shown in Example 14-8. In addition, decimal points are allowed in conjunction with these postfixes. For example, a rate of 10.5 Mbps could be entered with the policy map command **police 10.5m.** Although these policing rates are converted to their full bits per second values within the configuration, it makes the entering of these rate more user friendly and less error prone (as could easily be the case when having to enter up to 10 zeros to define the policing rate).

Note Although technically it is possible to apply an aggregate policing policy such as this on a per-VLAN basis, it is not advisable to do so. This is because the number of endpoints in a given VLAN can dynamically vary, yet the policing rates are statically fixed at an aggregate level, resulting in unpredictable bandwidth allotments per endpoint.

You can verify the configuration in Example 14-8 with the following commands:

- **show mls qos interface**
- **show class-map**
- **show policy-map**
- **show policy-map interface**
- **show mls qos maps policed-dscp** (as shown in Example 14-9)
- **show mls qos interface interface x/y policers** (as shown in Example 14-10)

Example 14-9 *Verifying Global Policing Markdown Mappings on a Catalyst 3750:* ***show mls qos maps policed-dscp***

```
C3750# show mls qos maps policed-dscp
   Policed-dscp map:
     d1 :  d2 0  1  2  3  4  5  6  7  8  9
     ---------------------------------------
      0 :    08 01 02 03 04 05 06 07 08 09
      1 :    08 11 12 13 14 15 16 17 08 19
      2 :    20 21 22 23 24 25 26 27 28 29
      3 :    30 31 32 33 34 35 36 37 38 39
      4 :    40 41 42 43 44 45 46 47 48 49
      5 :    50 51 52 53 54 55 56 57 58 59
      6 :    60 61 62 63
```

Example 14-9 shows the policing DSCP-markdown mapping. The first digit of the DSCP value of a packet offered to a policer is shown along the Y-axis of the table. The second digit of the DSCP value of a packet offered to a policer is shown along the X-axis of the table. For example, the DSCP value for the transactional data application class (AF21/18) is found in the row d1=1 and column d2=8. And, as shown, packets with this offered DSCP value (along with DF/0 and AF11/10) are re-marked to CS1 (08) if found to be in excess of the policing rate.

Example 14-10 *Verifying Interface Policers on a Catalyst 3750:* ***show mls qos interface interface x/y policers***

```
C3750# show mls qos interface GigabitEthernet 1/0/1 policers
GigabitEthernet1/0/1
policymap=PER-PORT-POLICING
type=Single, id=1 rate=128000, qlimit=8000, drop=1
type=Single, id=2 rate=32000, qlimit=8000, drop=1
type=Single, id=3 rate=5000000, qlimit=8000, drop=1
type=Single, id=4 rate=32000, qlimit=8000, drop=1
type=Single, id=5 rate=10000000, qlimit=8000, drop=0
type=Single, id=6 rate=10000000, qlimit=8000, drop=0
type=Single, id=7 rate=10000000, qlimit=8000, drop=1
type=Single, id=8 rate=10000000, qlimit=8000, drop=0
```

Example 14-10 shows the interface policers for GigabitEthernet 1/0/1, including the policing rates, burst, and drop-function values (drop=1 means that exceeding-traffic is dropped, while drop=0 value means that exceeding-traffic is not dropped, but re-marked).

Queuing Models

Because the total inbound bandwidth of all ports on a Catalyst 3750 switch can exceed the bandwidth of the stack or internal ring, ingress queues are located after the packet is classified, policed, and marked and before packets are forwarded into the switch fabric (as shown in Figure 14-2). In addition, because multiple ingress ports can simultaneously send packets to an egress port and cause congestion, outbound queues are located after the stack or internal ring.

Both the ingress and egress queues on the Catalyst 3750 are serviced by a shaped round-robin (SRR) scheduling algorithm. SRR can be configured in two modes: shaped or sharing.

In shaped mode, the egress queues are guaranteed a percentage of the bandwidth and they are rate limited to that amount. Shaped traffic does not use more than the allocated bandwidth even if the link is idle. Shaping provides a more even flow of traffic over time and reduces the peaks and valleys of bursty traffic. With shaping, the absolute value of each weight is used to compute the bandwidth available for the queues. SRR shaping is configured with the **srr-queue bandwidth shape** interface command.

In shared mode, the ingress or egress queues share the bandwidth among them according to the configured weights. The bandwidth is guaranteed at this level but not limited to it. For example, if a queue is empty and no longer requires a share of the link, the remaining queues can expand into the unused bandwidth and share it among them. With sharing, the ratio of the weights controls the frequency of de-queuing; the absolute values are meaningless. SRR sharing is configured with the **srr-queue bandwidth share** interface command.

Furthermore, both the ingress and egress queuing structures support the enabling of a single priority queue, or expedite queue, as it corresponds to the EF PHB. An ingress or egress queue operating as an expedite queue is fully serviced ahead of all other queues until empty. After the priority queue has been fully serviced, the scheduler services the nonpriority queues, which are configured in either shaped or shared SRR modes. A strict-priority queue is enabled with the **priority-queue** interface command.

With respect to the scheduling hierarchy in the Catalyst 3750 switch, shaped mode overrides shared mode, and priority mode overrides both shaped and shared modes.

In addition, the Catalyst 3750 supports the weighted tail drop (WTD) congestion avoidance mechanism. WTD is implemented on queues to manage the queue lengths and to provide drop preferences for different traffic classifications. As a packet is enqueued to a particular ingress or egress queue, WTD uses the frame's assigned internal DSCP to subject it to different drop thresholds. If the threshold is exceeded for a given internal DSCP value (in other words, the space available in the destination queue is less than the size of the packet), the switch drops the packet. Each queue has three threshold values. The internal DSCP determines which of the three threshold values is subjected to the frame. Of the three thresholds, two are configurable (explicit) and one is not (implicit), because this last threshold corresponds to the tail of the queue (100 percent limit).

Packets are mapped to queues and thresholds on the Catalyst 3750 by either CoS-to-queue/threshold or DSCP-to-queue/threshold mappings. The mapping used directly corresponds to whether the packet was configured to trust CoS on ingress or to trust DSCP on ingress (untrusted packets are simply assigned to the default queue).

Ordinarily in each platform-specific design chapter, separate queuing models are presented for the 4-class, 8-class, and 12-class strategic end-to-end QoS models presented in Chapter 11. However, because the Catalyst 3750 has a fixed set of 4 hardware queues and because the configuration and functionality between 4-, 8-, and 12-class queuing models will be nearly identical, only a single ingress and egress queuing model is presented for this platform.

Ingress Queuing Model

Because the Catalyst 3750 switch platforms have architectures based on oversubscription, they have been engineered to guarantee QoS by protecting critical traffic trying to access the backplane/stack-ring via ingress queuing. Ingress queuing on this platform can be configured as 2Q3T or 1P1Q3T, with the latter being the recommended configuration (because it supports the RFC 3246 EF PHB).

1P1Q3T ingress queuing is configured by explicitly enabling Q2 as a priority queue and assigning it a bandwidth allocation, such as 30 percent. Next, an SRR weight can be assigned to the nonpriority queue, which in this case would be 70 percent. The buffer allocations can be tuned such that Q1 gets 90 percent of the buffers, while Q2 (the PQ) gets only 10 percent; because the PQ is serviced in real-time, it is generally more efficient to provision fewer buffers to it and more to the nonpriority queues. After this, WTD thresholds can be defined on Q1 to provide interqueue QoS; specifically, Q1T1 can be explicitly set at 80 percent queue depth, and Q1T2 can be explicitly set at 90 percent queue depth (while Q3 remains implicitly set at 100 percent queue depth).

With the queues and thresholds set, VoIP (EF), broadcast video (CS5), and real-time interactive (CS4) traffic can be mapped to the strict-priority ingress queue. All other traffic classes can be mapped to the default (nonpriority) ingress queue. However, drop preference can be given to control plane traffic, such that network control (CS7) and internetwork control (CS6) traffic is mapped to the highest WTD threshold (Q1T3). In addition, signaling (CS3) traffic can be mapped to the middle WTD threshold (Q1T2). All other flows would be mapped to Q1T1. Figure 14-3 shows these 1P1Q3T ingress queuing mappings for the Catalyst 3750.

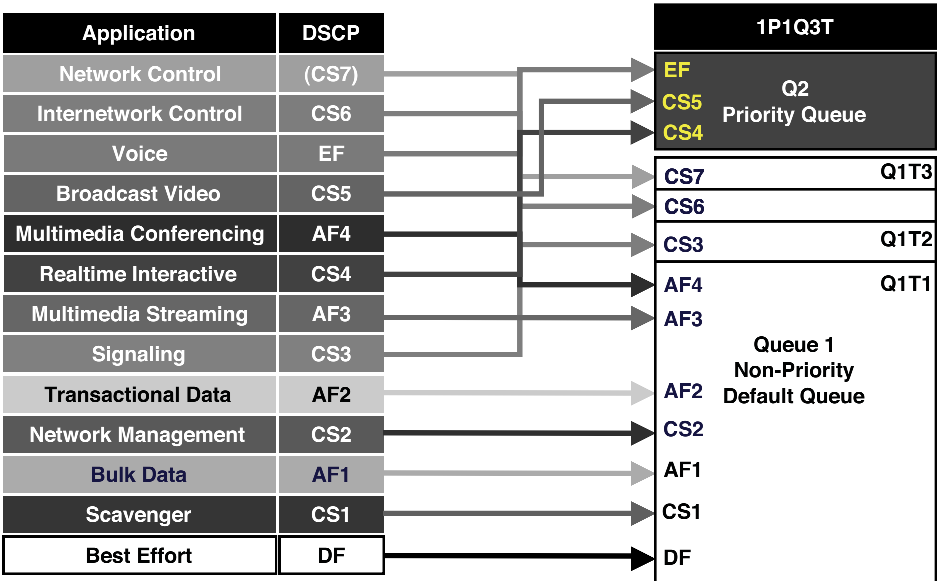


Figure 14-3 *Catalyst 3750 Ingress Queuing Model*

Example 14-11 shows the corresponding configuration for 1P1Q3T ingress queuing on the Catalyst 3750.

Example 14-11 *1P1Q3T Ingress Queuing Configuration Example on a Catalyst 3750*

```
 ! This section configures the ingress queues
C3750(config)# mls qos srr-queue input priority-queue 2 bandwidth 30
 ! Q2 is enabled as a strict-priority ingress queue with 30% BW
C3750(config)# mls qos srr-queue input bandwidth 70 30
 ! Q1 is assigned 70% BW via SRR shared weights
 ! Q2 SRR shared weight is ignored (as it has been configured as a PQ)
C3750(config)# mls qos srr-queue input buffers 90 10
 ! Q1 is assigned 90% of queuing buffers and Q2 (PQ) is assigned 10%
C3750(config)# mls qos srr-queue input threshold 1 80 90
 ! Q1 thresholds are configured at 80% (Q1T1) and 90% (Q1T2)
 ! Q1T3 is implicitly set at 100% (the tail of the queue)
 ! Q2 thresholds are all set (by default) to 100% (the tail of Q2)

 ! This section configures ingress CoS-to-queue mappings (if required)
C3750(config)# mls qos srr-queue input cos-map queue 1 threshold 1 0 1 2
 ! CoS values 0, 1 and 2 are mapped to Q1T1
C3750(config)# mls qos srr-queue input cos-map queue 1 threshold 2 3
 ! CoS value 3 is mapped to ingress Q1T2
C3750(config)# mls qos srr-queue input cos-map queue 1 threshold 3 6 7
```

```
 ! CoS values 6 and 7 are mapped to ingress Q1T3
C3750(config)# mls qos srr-queue input cos-map queue 2 threshold 1 4 5
 ! CoS values 4 and 5 are mapped to ingress Q2 (the PQ)

 ! This section configures ingress DSCP-to-queue Mappings
C3750(config)# mls qos srr-queue input dscp-map queue 1 threshold 1 0 8 10 12 14
 ! DSCP DF, CS1 and AF1 are mapped to ingress Q1T1
C3750(config)# mls qos srr-queue input dscp-map queue 1 threshold 1 16 18 20 22
 ! DSCP CS2 and AF2 are mapped to ingress Q1T1
C3750(config)# mls qos srr-queue input dscp-map queue 1 threshold 1 26 28 30 34 36
38
 ! DSCP AF3 and AF4 are mapped to ingress Q1T1
C3750(config)# mls qos srr-queue input dscp-map queue 1 threshold 2 24
 ! DSCP CS3 is mapped to ingress Q1T2
C3750(config)# mls qos srr-queue input dscp-map queue 1 threshold 3 48 56
 ! DSCP CS6 and CS7 are mapped to ingress Q1T3 (the tail of Q1)
C3750(config)# mls qos srr-queue input dscp-map queue 2 threshold 3 32 40 46
 ! DSCP CS4, CS5 and EF are mapped to ingress Q2T3 (the tail of the PQ)
```

Note CoS-to-queue mappings are required only if some switch ports are configured to trust CoS on ingress. In this case, the CoS-to-DSCP map should also be modified to map CoS 5 to DSCP EF (as shown in Example 14-2). In addition, note that because of the limited granularity of CoS-to-queue mapping, it is not possible to assign multimedia conferencing (AF4) and real-time interactive (CS4) traffic into separate queues (because both share the same CoS value of 4); nor is it possible to assign signaling (CS3) and multimedia streaming (AF3) traffic into separate queue thresholds (because both share the same CoS value of 3).

Note Nonstandard DSCP-to-queue mappings are not shown in the configurations in this chapter for the sake of simplicity and brevity.

You can verify the configuration in Example 14-11 with the following commands:

- **show mls qos input-queue** (shown in Example 14-12)
- **show mls qos maps cos-input-q** (shown in Example 14-13)
- **show mls qos maps dscp-input-q** (shown in Example 14-14)

Example 14-12 *Verifying Ingress Queuing on a Catalyst 3750:* ***show mls qos input-queue***

```
C3750# show mls qos input-queue
Queue     :       1       2
------------------------------------------
buffers   :      90      10
bandwidth :      70      30
priority  :       0      30
threshold1:      80     100
threshold2:      90     100
```

Example 14-12 shows that ingress queuing buffers and bandwidth have been allocated between Q1 and Q2 by a 70:30 split, respectively. Also, Q2 has been enabled as a strict-priority queue with a 30 percent maximum bandwidth guarantee. Q1T1 and Q1T2 thresholds have been set to 80 percent and 90 percent, but all Q2 thresholds are at 100 percent.

Example 14-13 *Verifying Ingress Queue Mapping on a Catalyst 3750:* ***show mls qos maps cos-input-q***

```
C3750# show mls qos maps cos-input-q
   Cos-inputq-threshold map:
              cos:  0   1   2   3   4   5   6   7
              ------------------------------------
  queue-threshold: 1-1 1-1 1-1 1-2 2-3 2-3 1-3 1-3
```

Example 14-13 shows the ingress CoS-to-queue mappings. Specifically, CoS values 1 and 2 have been mapped to Q1T1, CoS 3 has been mapped to Q1T2, CoS values 4 and 5 have been mapped to Q2T1 (the PQ), and CoS values 6 and 7 have been mapped to Q1T3.

Example 14-14 *Verifying Ingress Queue Mapping on a Catalyst 3750:* ***show mls qos maps dscp-input-q***

```
C3750# show mls qos maps dscp-input-q
   Dscp-inputq-threshold map:
    d1 :d2    0     1     2     3     4     5     6     7     8     9
     ------------------------------------------------------------
     0 :    01-01 01-01 01-01 01-01 01-01 01-01 01-01 01-01 01-01 01-01
     1 :    01-01 01-01 01-01 01-01 01-01 01-01 01-01 01-01 01-01 01-01
     2 :    01-01 01-01 01-01 01-01 01-02 01-01 01-01 01-01 01-01 01-01
     3 :    01-01 01-01 02-03 01-01 01-01 01-01 01-01 01-01 01-01 01-01
     4 :    02-03 02-01 02-01 02-01 02-01 02-01 02-03 02-01 01-03 01-01
     5 :    01-01 01-01 01-01 01-01 01-01 01-01 01-03 01-01 01-01 01-01
     6 :    01-01 01-01 01-01 01-01
```

Example 14-14 shows the ingress DSCP-to-queue mappings. The first digit of the DSCP value of a packet is shown along the Y-axis of the table; the second digit of the DSCP value of a packet is shown along the X-axis of the table. The mapping table corresponds to Figure 14-3. Note that CS4 (DSCP 32), CS5 (DSCP 40), and EF (DSCP 46) are all mapped to Q2 (the PQ). It should also be noted that internal DSCP values 40 through 47 are mapped to Q2T1 by default, which is why the table shows additional values being mapped to this queue.

Egress Queuing Models

Egress queuing on the Catalyst 3750 family of switches can be configured as 4Q3T or 1P3Q3T, with the latter being the recommended configuration (because it supports the RFC 3246 EF PHB).

Two different egress queuing sets can be configured on the Catalyst 3750. However, to maintain consistent per-hop behaviors, it is generally recommended to use only one.

A unique feature of the Catalyst 3750 is that it supports flexible buffer allocations to hardware queues, which may be dynamically loaned or borrowed against (as needed). Specifically, each queue can lend part of its buffering capacity, unless a specified minimum reserve threshold has been reached. In addition, each queue may borrow up to four times its capacity from a common pool of buffers (which are not allocated to any specific queue) should these be available for use. The recommended buffer allocations for queues 1 through 4 are 20 percent, 30 percent, 35 percent, and 15 percent, respectively. Correspondingly, the recommended parameters for reserve thresholds and maximum (overload) thresholds for nonpriority queues are 100 percent and 400 percent, respectively. For the priority queue, all thresholds should be set to 100 percent

Once the primary queuing set has been configured for 1P3Q3T egress queuing, WTD thresholds can be defined on Q2 and Q4 to provide intraqueue QoS. Specifically, Q2T1 can be explicitly set at 80 percent queue depth, and Q2T2 can be explicitly set at 90 percent queue depth (while Q3 remains implicitly set at 100 percent queue depth). Also, Q4T1 can be explicitly set at 60 percent queue depth, while the other thresholds for Q4 remain at their default values (of 100 percent queue depth), with the exception of the maximum (overload) threshold, which can be set to 400 percent. This last setting allows for even scavenger and bulk data traffic to benefit from the extended buffering capabilities of this platform, especially when considering that these are the least favored flows from a bandwidth perspective and thus will likely need the deepest queues.

With the queues and thresholds set, VoIP (EF), broadcast video (CS5), and real-time interactive (CS4) traffic can be mapped to the strict-priority egress queue (Q1). Network management (CS2), transactional data (AF2), multimedia streaming (AF3), and multimedia conferencing (AF4) traffic can be mapped to Q2T1. Signaling (CS3) traffic can be mapped to Q2T2. Network (CS7) and internetwork (CS6) traffic can be mapped to Q2T3. Default (DF) traffic can be mapped to Q3, the default queue. Scavenger (CS1) traffic can be mapped to Q4T1, while bulk data (AF1) is mapped to Q4T2. Figure 14-4 illustrates these 1P3Q3T egress queuing mappings for the Catalyst 3750.

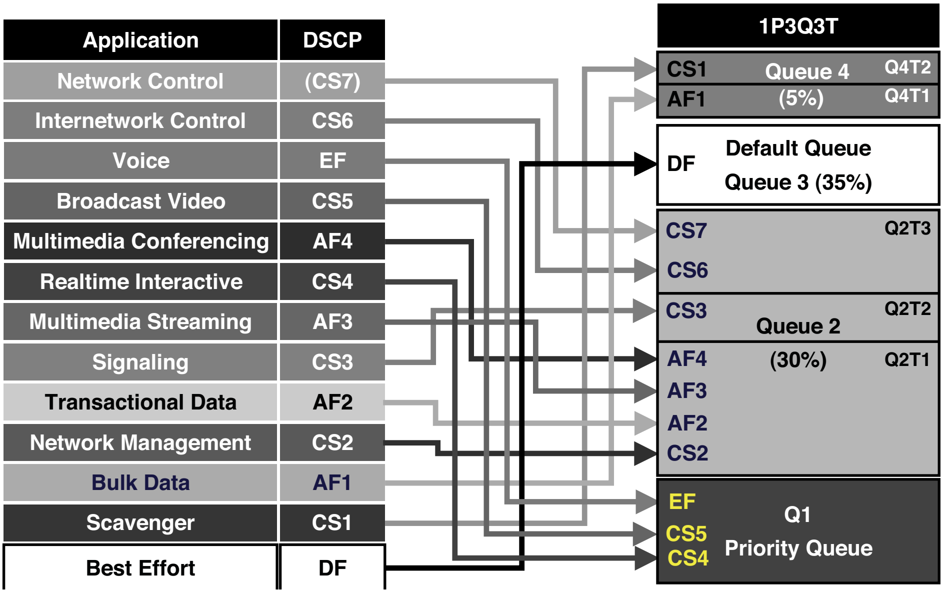


Figure 14-4 *Catalyst 3750 Egress Queuing Model*

Example 14-15 shows the corresponding configuration for 1P3Q3T egress queuing on the Catalyst 3750.

Example 14-15 *1P3Q3T Egress Queuing Configuration Example on a Catalyst 3750*

```
! This section configures buffers and thresholds on Q1 through Q4
C3750(config)# mls qos queue-set output 1 buffers 15 30 35 20
 ! Queue buffers are allocated
C3750(config)# mls qos queue-set output 1 threshold 1 100 100 100 100
 ! All Q1 (PQ) Thresholds are set to 100%
C3750(config)# mls qos queue-set output 1 threshold 2 80 90 100 400
 ! Q2T1 is set to 80%; Q2T2 is set to 90%;
 ! Q2 Reserve Threshold is set to 100%;
 ! Q2 Maximum (Overflow) Threshold is set to 400%
C3750(config)# mls qos queue-set output 1 threshold 3 100 100 100 400
 ! Q3T1 is set to 100%, as all packets are marked the same weight in Q3
 ! Q3 Reserve Threshold is set to 100%;
 ! Q3 Maximum (Overflow) Threshold is set to 400%
C3750(config)# mls qos queue-set output 1 threshold 4 60 100 100 400
 ! Q4T1 is set to 60%; Q4T2 is set to 100%
 ! Q4 Reserve Threshold is set to 100%;
 ! Q4 Maximum (Overflow) Threshold is set to 400%

 ! This section configures egress CoS-to-Queue mappings (if required)
```

```
C3750(config)# mls qos srr-queue output cos-map queue 1 threshold 3 4 5
 ! CoS 4 and 5 are mapped to egress Q1T3 (the tail of the PQ)
C3750(config)# mls qos srr-queue output cos-map queue 2 threshold 1 2
 ! CoS 2 is mapped to egress Q2T1
C3750(config)# mls qos srr-queue output cos-map queue 2 threshold 2 3
 ! CoS 3 is mapped to egress Q2T2
C3750(config)# mls qos srr-queue output cos-map queue 2 threshold 3 6 7
 ! CoS 6 and 7 are mapped to Q2T3
C3750(config)# mls qos srr-queue output cos-map queue 3 threshold 3 0
 ! CoS 0 is mapped to Q3T3 (the tail of the default queue)
C3750(config)# mls qos srr-queue output cos-map queue 4 threshold 3 1
 ! CoS 1 is mapped to Q4T3 (tail of the less-than-best-effort queue)

 ! This section configures egress DSCP-to-Queue mappings
C3750(config)# mls qos srr-queue output dscp-map queue 1 threshold 3 32 40 46
 ! DSCP CS4, CS5 and EF are mapped to egress Q1T3 (tail of the PQ)
C3750(config)# mls qos srr-queue output dscp-map queue 2 threshold 1 16 18 20 22
 ! DSCP CS2 and AF2 are mapped to egress Q2T1
C3750(config)# mls qos srr-queue output dscp-map queue 2 threshold 1 26 28 30 34 36
38
 ! DSCP AF3 and AF4 are mapped to egress Q2T1
C3750(config)# mls qos srr-queue output dscp-map queue 2 threshold 2 24
 ! DSCP CS3 is mapped to egress Q2T2
C3750(config)# mls qos srr-queue output dscp-map queue 2 threshold 3 48 56
 ! DSCP CS6 and CS7 are mapped to egress Q2T3
C3750(config)# mls qos srr-queue output dscp-map queue 3 threshold 3 0
 ! DSCP DF is mapped to egress Q3T3 (tail of the best effort queue)
C3750(config)# mls qos srr-queue output dscp-map queue 4 threshold 1 8
 ! DSCP CS1 is mapped to egress Q4T1
C3750(config)# mls qos srr-queue output dscp-map queue 4 threshold 2 10 12 14
 ! DSCP AF1 is mapped to Q4T3 (tail of the less-than-best-effort queue)

 ! This section configures interface egress queuing parameters
C3750(config)# interface range GigabitEthernet1/0/1-48
C3750(config-if-range)# queue-set 1
 ! The interface(s) is assigned to queue-set 1
C3750(config-if-range)# srr-queue bandwidth share 1 30 35 5
 ! The SRR sharing weights are set to allocate 30% BW to Q2
 ! 35% BW to Q3 and 5% BW to Q4
 ! Q1 SRR sharing weight is ignored, as it will be configured as a PQ
C3750(config-if-range)# priority-queue out
 ! Q1 is enabled as a strict-priority queue
```

> **Note** As previously mentioned, only a single queuing model is presented for the 4-class, 8-class, and 12-class strategic QoS models (presented in Chapter 11) for this platform, because the configuration and functionality between these queuing models is nearly identical. The only exception to this is four-class model, in which case it is recommended to make the following change to Example 14-15: **mls qos srr-queue output dscp-map queue 4 threshold 2 24**. This change will map signaling traffic (marked CS3/DSCP 24) to Q4T2 (instead of Q2T2), which better aligns to the four-class strategic model.

> **Note** CoS-to-queue mappings are only required if some switch ports are configured to trust CoS on ingress. In which case, the CoS-to-DSCP map should also be modified to map CoS 5 to DSCP EF (as shown in Example 14-2). In addition, note that because of the limited granularity of CoS-to-queue mapping, it is not possible to assign multimedia conferencing (AF4) and real-time interactive (CS4) traffic into separate queues (because both share the same CoS value of 4). Nor is it possible to assign signaling (CS3) and multimedia streaming (AF3) traffic into separate queue thresholds (because both share the same CoS value of 3). Nor is it possible to assign scavenger (CS1) and bulk data (AF1) traffic into separate queue thresholds (because both share the same CoS value of 1).

You can verify the configuration in Example 14-15 with the following commands:

- **show mls qos queue-set** (shown in Example 14-16)
- **show mls qos maps cos-output-q**
- **show mls qos maps dscp-output-q**
- **show mls qos interface interface x/y queueing** (shown in Example 14-17)
- **show mls qos interface interface x/y statistics** (shown in Example 14-18)

Example 14-16 *Verifying Egress Queuing on a Catalyst 3750:* ***show mls qos queue-set***

```
C3750# show mls qos queue-set 1
Queueset: 1
Queue     :       1       2       3       4
----------------------------------------------
buffers   :      15      30      35      20
threshold1:     100      80     100      60
threshold2:     100      90     100     100
reserved  :     100     100     100     100
maximum   :     100     400     400     400
```

Example 14-16 shows that the queuing buffers, drop thresholds, reserve thresholds, and maximum (overload) thresholds have been configured correctly on a per-queue-set basis.

Example 14-17 *Verifying Egress Queuing on a Catalyst 3750:* ***show mls qos interface interface x/y queueing***

```
C3750# show mls qos interface GigabitEthernet 1/0/1 queueing
GigabitEthernet1/0/1
Egress Priority Queue : enabled
Shaped queue weights (absolute) :  25 0 0 0
Shared queue weights  :  1 30 35 5
The port bandwidth limit : 100  (Operational Bandwidth:100.0)
The port is mapped to qset : 1
```

Example 14-17 shows that strict-priority queuing has been enabled on the interface, and that the queues Q2, Q3, and Q4 receive 30 percent, 35 percent, and 5 percent of the remaining bandwidth, respectively.

Example 14-18 *Verifying Egress Queuing on a Catalyst 3750:* ***show mls qos interface interface x/y statistics***

```
C3750# show mls qos interface GigabitEthernet 1/0/49 statistics
GigabitEthernet1/0/49 (All statistics are in packets)
  dscp: incoming
-------------------------------
 0 -  4 :        1729            0            0            0         0
 5 -  9 :           0            0            0            0         0
10 - 14 :           0            0            0            0         0
15 - 19 :           0            0            0            0         0
20 - 24 :           0            0            0            0         0
25 - 29 :           0            0            0            0         0
30 - 34 :           0            0            0            0         0
35 - 39 :           0            0            0            0         0
40 - 44 :           0            0            0            0         0
45 - 49 :           0       127292            0         1263         0
50 - 54 :           0            0            0            0         0
55 - 59 :           0            0            0            0         0
60 - 64 :           0            0            0            0

 dscp: outgoing
-------------------------------
 0 -  4 :      947678            0            0            0         0
 5 -  9 :           0            0            0     23842155         0
10 - 14 :     1190043            0            0            0         0
```

```
15 - 19 :           0            0            0      1061726         0
20 - 24 :           0            0            0            0     10372
25 - 29 :           0            0            0            0         0
30 - 34 :           0            0            0            0   8320623
35 - 39 :           0            0            0            0         0
40 - 44 :           0            0            0            0         0
45 - 49 :           0       127291            0          784         0
50 - 54 :           0            0            0            0         0
55 - 59 :           0            0            0            0         0
60 - 64 :           0            0            0            0
  cos: incoming
------------------------------
 0 -  4 :      130653            0            0          998         0
 5 -  7 :      127599          613         3156
  cos: outgoing
------------------------------
 0 -  4 :      947754     25032199      1061726        10372   8320623
 5 -  7 :      127291          784         3462
  output queues enqueued:
queue: threshold1 threshold2 threshold3
----------------------------------------
 queue 0:           0           0      127291
 queue 1:     9382416       10396        4246
 queue 2:           0           0      947611
 queue 3:    23842155     1190043           0
  output queues dropped:
queue: threshold1 threshold2 threshold3
----------------------------------------
 queue 0:            0            0            0
 queue 1:            0            0            0
 queue 2:            0            0            0
 queue 3:          892            0            0
Policer: Inprofile:            0 OutofProfile:           0
```

Example 14-18 shows a set of dynamically updated packet statistic tables for an uplink port on an access layer Catalyst 3750 switch that is primarily congested in the access-to-distribution direction. The first table shows the incoming DSCP values (from the distribution layer). DSCP values are broken into groups of five. For example, incoming packets marked DSCP EF/46 are listed in the DSCP 45–49 row in the second column (in this case, 127,291 packets). The second table shows the outgoing packets (to the distribution layer) in a similar format. For example, DSCP CS1/8 is listed in the DSCP 5–9 row in the third column (23,842,155 packets). The third table shows incoming packets (from the distribution layer) by CoS values (again grouped in sets of four). Similarly, the fourth table shows outgoing packets (to the distribution layer) by CoS values. The fifth and sixth tables are particularly interesting in terms of queuing statistics: The fifth table shows the number of packets assigned to each queue/threshold combination.

Note The queue numbers in this verification command output are one less than the queue numbers used in the actual configuration syntax. For example, in the configuration, the queues are numbered 1 through 4 (and there is no such thing as a queue 0). However, in this verification output, the statistics for queue 1 are presented as for queue 0. Similarly, Q2 is shown here as Q1, Q3 is shown here as Q2, and Q4 is shown here as Q3. This is simply an engineering oversight.

For example, from the fifth table, it can be seen that 127,291 packets were sent to the (tail of the) PQ (shown here as Q0); similarly, 23,842,155 packets were sent to the scavenger/bulk queue first threshold (shown here as Q3T1). Finally, the sixth table shows any drops that have occurred on a per-queue/per-threshold basis. From this table, you can see that 892 drops occurred in the scavenger/bulk queue first threshold (Scavenger class drops).

Additional Platform-Specific QoS Design Options

These designs represent some generic building blocks for Catalyst 3750 QoS in a campus access-edge context, but they are by no means the only design options available to you. Additional options and considerations include the following:

- Per-VLAN QoS
- Per-port/per-VLAN QoS
- EtherChannel QoS
- AutoQoS SRND4
- Control plane policing

Each of these additional QoS design options and considerations is briefly discussed in turn.

Per-VLAN QoS Design

An alternative approach for deploying marking policies on the Catalyst 3750 platforms is to deploy these on a per-VLAN basis. To do so, you need to configure the interfaces belonging to the VLANs with the **mls qos vlan-based** interface command. In addition, the policy map can be simplified/broken apart, as applicable to each VLAN. Example 14-19 shows a per-VLAN marking model.

Example 14-19 *Per-VLAN Policy Example on a Catalyst 3750*

```
! This section configures the interface(s) for VLAN-based QoS
C3750(config)# interface range GigabitEthernet 1/0/1-48
C3750(config-if-range)# switchport access vlan 10
C3750(config-if-range)# switchport voice vlan 110
C3750(config-if-range)# spanning-tree portfast
C3750(config-if-range)# mls qos vlan-based
! Enables VLAN-based QoS on the interface(s)
 ! This section attaches the DVLAN policy to the DVLAN interface
C3750(config)# interface Vlan 10
C3750(config-if)# description DVLAN
C3750(config-if)# service-policy input DVLAN-MARKING
 ! Attaches the DVLAN Per-VLAN Marking policy to the DVLAN interface

 ! This section attaches the VVLAN policy to the VVLAN interface
C3750(config)# interface Vlan 110
C3750(config-if)# description VVLAN
C3750(config-if)# service-policy input VVLAN-MARKING
 ! Attaches the VVLAN Per-VLAN Marking policy to the VVLAN interface
```

You can verify the configuration in Example 14-19 with the following commands:

- **show mls qos interface**
- **show class-map**
- **show policy-map**
- **show policy-map interface**

Per-Port/Per-VLAN QoS

Technically speaking, the Catalyst 3750 supports per-port/per-VLAN QoS. However, the implementation of this feature on this platform is fairly convoluted, requiring hierarchical QoS policies.

The first step is to configure a class map that defines the switch ports to which the policers are attached. Then one or more per-port policers need to be defined (according to the various levels of policing rates or exceeding actions required). These policers reference the previously defined class map that specifies the switch ports are policed. These per-port policers comprise the “child” policy maps in the hierarchy.

Following this, “parent” policy maps are configured that combine the various per-port policers for the various classes of traffic for a given VLAN. Incidentally, a nested/child policy map can be referenced by only one parent service policy. Each of these parent policy maps reference child policies that implement the per-port policing functions. Finally, these parent policy maps are applied to the VLAN interfaces.

Furthermore, note that on Catalyst 3750, when you enable VLAN-based QoS and configure a hierarchical policy map in a switch stack (to enable per-port/per-VLAN QoS), these automatic actions occur when the stack configuration changes:

- When a new stack master is selected, the stack master reenables and reconfigures these features on all applicable interfaces on the stack master.
- When a stack member is added, the stack master reenables and reconfigures these features on all applicable ports on the stack member.
- When you merge switch stacks, the new stack master reenables and reconfigures these features on the switches in the new stack.
- When the switch stack divides into two or more switch stacks, the stack master in each switch stack reenables and reconfigures these features on all applicable interfaces on the stack members, including the stack master.

Therefore, although the Catalyst 3750 supports per-port/per-VLAN QoS, because it is so excessively complex (as compared to the same feature implementation on other platforms, such as the Catalyst 4500) and entails these additional caveats, it is recommended for advanced administrators only.

Note You can find an example per-port/per-VLAN policy at http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html#wp1099172.

EtherChannel QoS Design

QoS policies applied to EtherChannel links on the Catalyst 3750 are required to be identically configured on each and every EtherChannel physical port member interface; these include both ingress trust/classification/marking/policing policies, and egress queuing policies. (Ingress queuing policies are globally defined and therefore are not bound by this requirement.) If the policies are not identically configured, even though they may appear in the configuration, these will not take effect. No policy configuration needs to be configured on the logical port channel interface.

AutoQoS SRND4

As of August 2010, an updated version of AutoQoS was released for the Catalyst 2K and 3K family of switches with IOS Release 12.2(55)SE. This release was directly based on the recommendations put forward in latest QoS SRND to support Medianet applications. In fact, the new keyword and name for this version of AutoQoS is AutoQoS-SRND4 (taken from *Solution Reference Network Design Guide Version 4*, which is the Cisco name for this design chapter). Appendix A, "AutoQoS for Medianet," discusses AutoQoS-SRND4 in detail.

Control Plane Policing

At the time of this writing, control plane policing is not supported on the Catalyst 3750.

Summary

This design chapter detailed best-practice recommendations for QoS design on the Cisco Catalyst 3750 series switch in the role of a campus access layer switch. (In addition, the majority of these designs are recommendations are also compatible across the Catalyst 2K/3K switch family.)

The first step to deploying QoS on this MLS-QoS platform is to globally enable QoS. If this small, but important, step is not performed, any and all QoS commands configured will remain inactive.

The next step is to define ingress QoS policies, which may include trust (of CoS or DSCP or conditional trust of Cisco devices) and classification and marking (with optional policing policies). Ingress queuing is also recommended to be configured on this platform, because the sum of all ingress bandwidth potential exceeds the backplane/ring capacity. The recommended ingress queuing model is 1P1Q3T.

Egress queuing policies were also detailed, showing how a 4-class, 8-class, and 12-class strategic model can all be provisioned on this platform, despite a hardware limitation of four queues (via a 1P3Q3T egress queuing model).

Additional platform-specific design options and considerations were discussed, including how to configure per-VLAN QoS and QoS for EtherChannels. The AutoQoS SRND4 feature is an option on this platform also, but is discussed in Appendix A.

Additional Reading

Cisco Enterprise Medianet Campus QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html

Medianet Campus QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampusaag.html

Medianet Catalyst 3560/3750 QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampuscat3xxxaag.html

Medianet Campus AutoQoS At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/autoqosmediacampus.pdf

Cisco Catalyst 3750-X and Catalyst 3560-X Switch Software Configuration Guide, Cisco IOS Release 15.0(2)SE—Configuring QoS http://www.cisco.com/en/US/docs/switches/lan/catalyst3750x_3560x/software/release/15.0_2_se/configuration/guide/swqos.html

Chapter 15

Campus Distribution (Cisco Catalyst 4500) QoS Design

The primary role of quality of service (QoS) in the campus distribution switch is to manage packet loss. Therefore, the distribution switch should trust differentiated services code point (DSCP) markings on ingress (as these have been previously set by access-edge switches) and perform both ingress (if required and supported) and egress queuing, as illustrated in Figure 15-1.

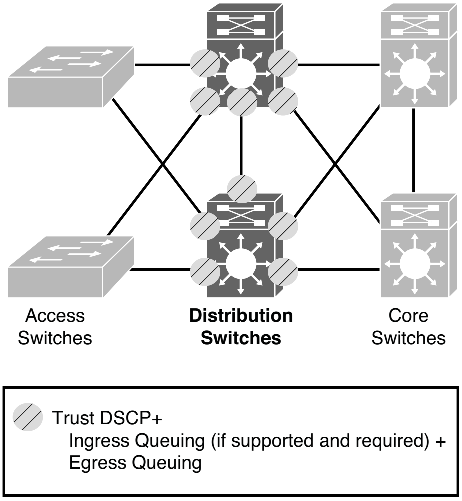


Figure 15-1 *Campus Distribution Switch Port QoS Roles*

The Cisco Catalyst 4500E Supervisor 7-E is a platform well suited to the role of a campus distribution switch and therefore is featured in this design chapter.

Incidentally, the QoS design requirements of a Catalyst 4500E Supervisor 7-E in the role of a distribution switch are generally equivalent to the requirements of a campus core switch.

Cisco Catalyst 4500 QoS Architecture

From a QoS perspective, the Cisco Catalyst 4500-E Supervisor 7-E is nearly identical to the Supervisor 6-E platform and the Catalyst 4500-X, because all of these platforms are Modular QoS command-line interface (MQC) based. However, earlier Catalyst 4500 platforms (such as the Supervisor II-Plus through Supervisor V-10GE) are Multi-Layer Switch (MLS)-QoS-based platforms and are referred to as *Classic Supervisors*.

Note QoS design for these older Classic Supervisors is beyond the scope for this design chapter. However, you can find design guidance for these platforms at http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html#wp1099634.

Figure 15-2 illustrates the QoS architecture for this Catalyst 4500E Supervisor 7-E (hereafter referred to simply as the Catalyst 4500) platform.

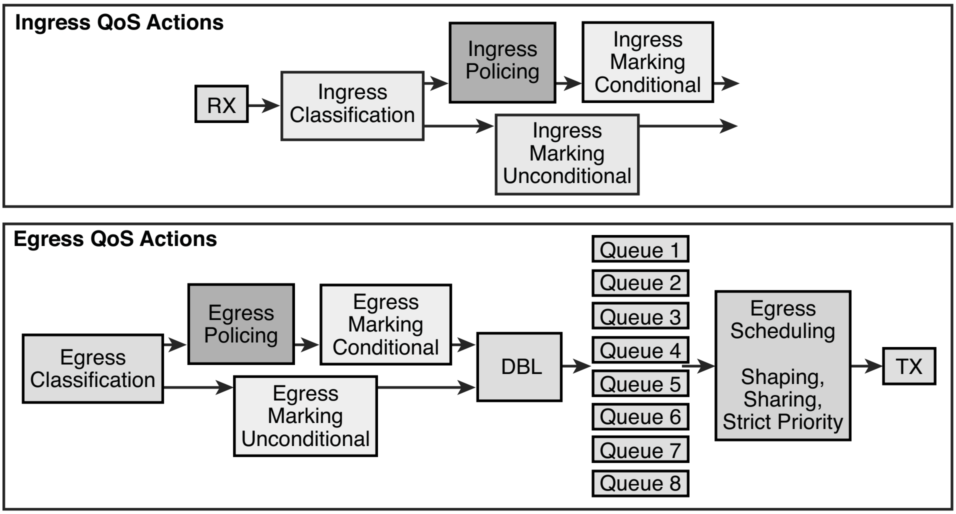


Figure 15-2 *Cisco Catalyst 4500 QoS Architectural Model*

QoS is enabled by default on all MQC-based platforms, which includes the Catalyst 4500. In addition, by default, all ports are set to a trust-DSCP/trust-CoS state.

In the MQC-based Catalyst 4500, QoS policies are applied as follows:

1. The incoming packet is classified (based on different packet fields, receive port, or VLAN) to belong to a traffic class.
2. Depending on the traffic class and configured polices, the packet is policed, which may result in the packet being dropped or re-marked.

3. After the packet has been marked/re-marked, it is looked up for forwarding. This action obtains the transmit port and VLAN to transmit the packet.
4. The packet is classified in the output direction based on the transmit port or VLAN/marking.
5. Depending on the output policies, the packet is policed, and may be dropped or re-marked.
6. The transmit queue for the packet is determined based on the traffic class and the configured egress queuing policies.
7. The transmit queue state is dynamically monitored via Dynamic Buffer Limiting (DBL) and drop threshold configuration to determine whether the packet should be dropped or queued for transmission.
8. If eligible for transmission, the packet is assigned to a transmit queue.

Based on these QoS operations, the design steps for configuring QoS on the Catalyst 4500 in the role of a distribution switch are discussed next.

QoS Design Steps

While there are two explicit QoS policy requirements of a distribution switch (namely to trust DSCP on ingress and queuing policies), because of the default QoS settings on MQC-based platforms there is effectively only a single step to configuring QoS on a Catalyst 4500 in this role:

1. Configure the ingress QoS model—which is recommended to be DSCP trust (and which is enabled by default on all MQC-based platforms).

Note This step may include ingress queuing policies on platforms which support this feature (however, the Catalyst 4500 does not support ingress queuing).

2. Configure egress queuing.

Queuing Models

Ingress queuing is not supported on the Catalyst 4500; only egress queuing is supported.

Note Other ingress QoS policies (including trust, classification, marking, and policing) are all supported; only ingress *queuing* is not supported on this platform.

The Catalyst 4500 supports a strict-priority hardware queue with (up to) seven additional nonpriority hardware queues. In addition, the Catalyst 4500 supports DSCP-to-queue mapping.

At the time of this writing, DSCP-based weighted random early detection (WRED) is not supported on the Catalyst 4500 platform. However, the Catalyst 4500 family uses a platform-specific congestion avoidance algorithm to provide active queue management (AQM), namely Dynamic Buffer Limiting (DBL). DBL tracks the queue length for each traffic flow in the switch. When the queue length of a flow exceeds its limit, DBL drop packets or sets the Explicit Congestion Notification (ECN) bits in the packet headers. The DBL algorithm can identify belligerent flows (that is, unchecked/nonadaptive/inelastic flows) and drop these more aggressively. Belligerent flows can use excessive bandwidth and switch buffers, resulting in poor application performance for well-behaved flows. Therefore, DBL can induce not only random "probabilistic drops" (in a manner similar to WRED), but also "belligerent flow drops," both of which are counted and displayed via the **show policy-map interface** command output on classes where DBL has been enabled (as demonstrated later in Example 15-4).

Therefore, the egress queuing model for the Catalyst 4500 platform can be expressed as 1P7Q1T+DBL.

Note DBL is unique to the Catalyst 4500 platforms. At the time of this writing, there are no tuning options for DBL.

The Catalyst 4500 can be configured to support 4-class, 8-class, or 12-class queuing models, as discussed in the following sections.

Four-Class Egress Queuing Model

In the four-class model (illustrated in Figures 11-3 and 11-4 in Chapter 11, "QoS Design Principles and Strategies"), the application class to queue mappings are as follows:

- Real-time traffic (marked EF) is assigned to the priority queue (which may be optionally policed to 30 percent bandwidth).
- Control traffic (marked CS3) is assigned to a dedicated nonpriority queue with a 10 percent bandwidth allocation.
- Transactional data (marked AF2) is assigned to another dedicated nonpriority queue with a 35 percent bandwidth allocation with DBL enabled.
- Best-effort traffic (marked DF) is assigned to a default queue with 25 percent bandwidth allocation with DBL enabled.

Note DBL is enabled *only* on the transactional data queue and the default queue (because real-time traffic and control traffic should never be early dropped).

Note When the priority queue is configured on one class of a policy map *without* a policer, only **bandwidth remaining percent** is accepted on other classes (guaranteeing a minimum bandwidth for other classes from the remaining bandwidth of what is left after using the priority queue). However, when the priority queue is configured *with* a policer, either **bandwidth percent** or **bandwidth remaining percent** is accepted on the other queuing classes.

Note If queuing policies are to be applied to EtherChannel interfaces, it is recommended not to police the priority queue. This is because two policy maps would be needed in this case: One policy map would be needed to police the priority queue (which would have to be applied to the logical EtherChannel interface in the egress direction), and a second policy map would be needed to define the queuing policy (using bandwidth remaining percent), which would be applied to all EtherChannel physical port-member interfaces in the egress direction. Therefore, to simplify the queuing policy and to increase its portability and modularity, the priority queue is not policed in the queuing design examples in this chapter (which necessitates the use of **bandwidth remaining percent** on nonpriority queues).

Note Although it is true that there will be fractional differences in bandwidth allotments to an application class depending on whether **bandwidth percent** or **bandwidth remaining percent** is used. However, because these differences are relatively minor, the same numeric values are used in these examples for the sake of consistency.

Figure 15-3 illustrates the resulting four-class (1P3Q1T+DBL) egress queuing model for the Catalyst 4500.

Example 15-1 shows the corresponding configuration for four-class (1P3Q1T+DBL) egress queuing on the Catalyst 4500.

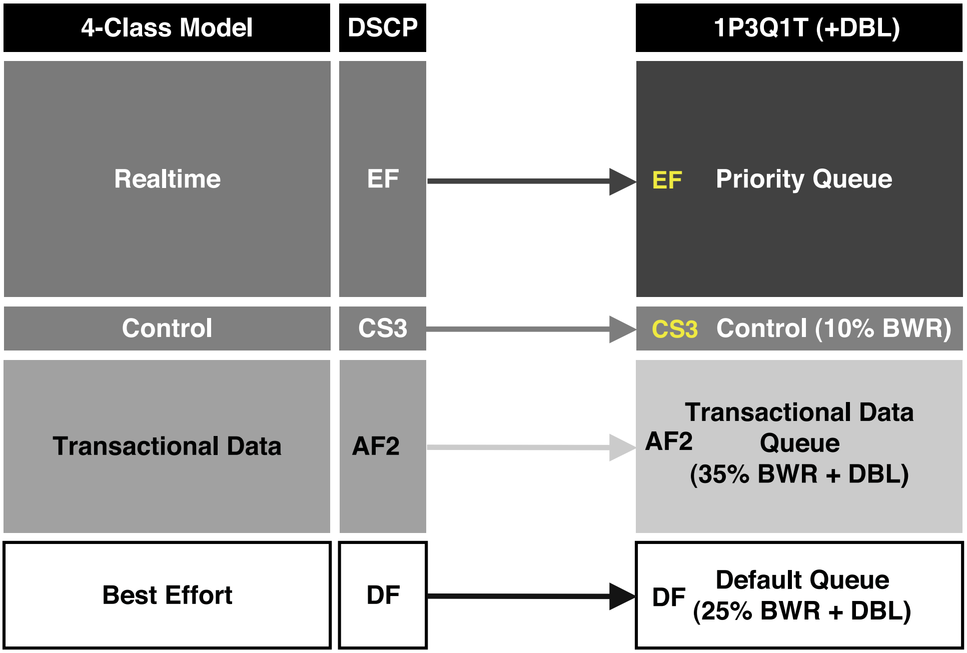


Figure 15-3 *Catalyst 4500 Four-Class Egress Queuing Model*

Example 15-1 *Four-Class (1P3Q1T+DBL) Egress Queuing Configuration Example on a Catalyst 4500*

```
 ! This section configures the class maps for the egress queuing policy
C4500(config)# class-map match-all PRIORITY-QUEUE
C4500(config-cmap)# match dscp ef
 ! VoIP (EF) is mapped to the PQ
C4500(config)# class-map match-all CONTROL-QUEUE
C4500(config-cmap)# match dscp cs3
 ! Signaling (CS3) is mapped to a dedicated queue
C4500(config)# class-map match-all TRANSACTIONAL-DATA-QUEUE
C4500(config-cmap)# match dscp af21 af22 af23
 ! Transactional Data (AF2) is mapped to a dedicated queue


 ! This section configures the four-class egress queuing policy map
C4500(config)# policy-map 1P3Q1T
C4500(config-pmap-c)# class PRIORITY-QUEUE
C4500(config-pmap-c)#  priority
 ! Enables the priority queue
C4500(config-pmap-c)# class CONTROL-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 10
```

```
 ! Defines the control queue with 10% BW remaining
C4500(config-pmap-c)# class TRANSACTIONAL-DATA-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 35
C4500(config-pmap-c)#  dbl
 ! Defines a transactional data queue with 35% BW remaining + DBL
C4500(config-pmap-c)# class class-default
C4500(config-pmap-c)#  bandwidth remaining percent 25
C4500(config-pmap-c)#  dbl
 ! Provisions the default/Best Effort queue with 25% BW remaining + DBL


 ! This section attaches the egress queuing policy to the interface(s)
C4500(config)# interface range TenGigabitEthernet 1/1-2
C4500(config-if-range)# service-policy output 1P3Q1T
```

Note Class maps defined for egress-queuing policies require unique names from any ingress-policy class maps; otherwise, classification errors can occur due to overlapping classification logic

You can verify the configuration in Example 15-1 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Eight-Class Egress Queuing Model

In the eight-class model (illustrated in Figures 11-5 and 11-6), the application class to queue mappings are as follows:

- Real-time traffic (marked EF) is assigned to the priority queue (which may be optionally policed to 10 percent bandwidth).
- Interactive video (marked AF4) is assigned to a dedicated nonpriority queue with a 23 percent bandwidth allocation with DBL enabled.
- Streaming video (marked AF3) is assigned to a dedicated nonpriority queue with a 10 percent bandwidth allocation with DBL enabled.
- Network control traffic (marked CS6) is assigned to a dedicated nonpriority queue with a 5 percent bandwidth allocation.
- Signaling traffic (marked CS3) is assigned to a dedicated nonpriority queue with a 2 percent bandwidth allocation.

- Transactional data (marked AF2) is assigned to dedicated nonpriority queue with a 24 percent bandwidth allocation with DBL enabled.
- Scavenger traffic (marked CS1) is constrained within a dedicated nonpriority queue with a 1 percent bandwidth allocation.
- Best-effort traffic (marked DF) is assigned to a default queue with 25 percent bandwidth allocation with DBL enabled.

Note As before, DBL is not enabled on the real-time or control traffic classes (because real-time traffic and control traffic should never be early dropped); nor would DBL be required on the scavenger class, because traffic in this class has no "good-faith" guarantee of service to begin with. Enabling DBL on the Interactive Video and Streaming Video classes assumes that the video codecs used for these flows are adaptive/elastic and therefore will adjust transmission rates in the event of congestion.

Figure 15-4 illustrates the resulting eight-class (1P7Q1T+DBL) egress queuing model for the Catalyst 4500.

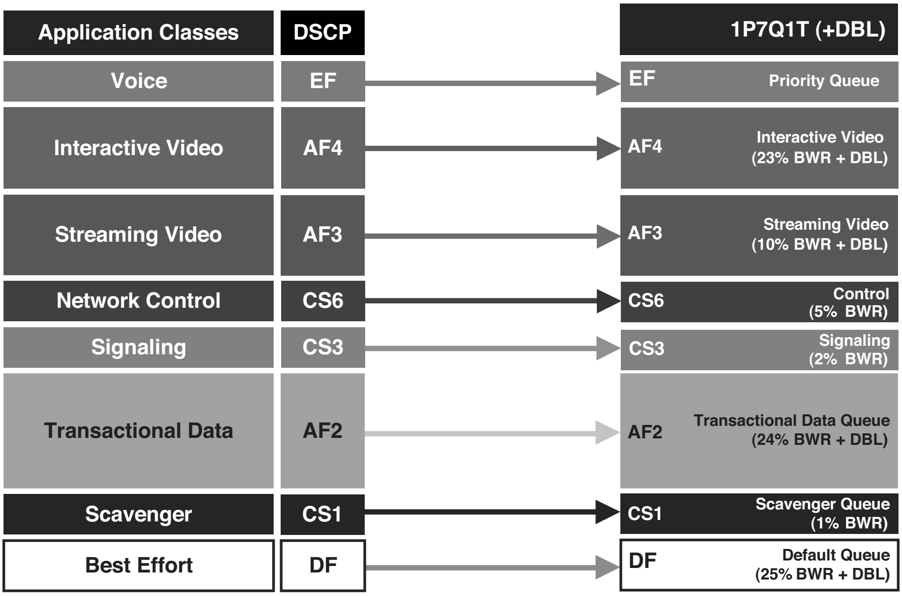


Figure 15-4 *Catalyst 4500 Eight-Class (1P7Q1T+DBL) Egress Queuing Model*

Example 15-2 shows the corresponding configuration for eight-class (1P7Q1T+DBL) egress queuing on the Catalyst 4500.

Example 15-2 *Eight-Class (1P7Q1T+DBL) Egress Queuing Configuration Example on a Catalyst 4500*

```
 ! This section configures the class maps for the egress queuing policy
C4500(config)# class-map match-all PRIORITY-QUEUE
C4500(config-cmap)# match dscp ef
 ! VoIP (EF) is mapped to the PQ
C4500(config)# class-map match-all INTERACTIVE-VIDEO-QUEUE
C4500(config-cmap)# match dscp af41 af42 af43
 ! Interactive-Video (AF4) is assigned a dedicated queue
C4500(config)# class-map match-all STREAMING-VIDEO-QUEUE
C4500(config-cmap)# match dscp af31 af32 af33
 ! Streaming-Video (AF3) is assigned a dedicated queue
C4500(config)# class-map match-all CONTROL-QUEUE
C4500(config-cmap)# match dscp cs6
 ! Network Control (CS6) is mapped to a dedicated queue
C4500(config)# class-map match-all SIGNALING-QUEUE
C4500(config-cmap)# match dscp cs3
 ! Signaling (CS3) is mapped to a dedicated queue
C4500(config)# class-map match-all TRANSACTIONAL-DATA-QUEUE
C4500(config-cmap)# match dscp af21 af22 af23
 ! Transactional Data (AF2) is assigned a dedicated queue
C4500(config)# class-map match-all SCAVENGER-QUEUE
C4500(config-cmap)# match dscp cs1
 ! Scavenger (CS1) is assigned a dedicated queue

 ! This section configures the 1P7Q1T+DBL egress queuing policy map
C4500(config)# policy-map 1P7Q1T
C4500(config-pmap-c)# class PRIORITY-QUEUE
C4500(config-pmap-c)#  priority
 ! Defines a priority queue
C4500(config-pmap-c)# class INTERACTIVE-VIDEO-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 23
C4500(config-pmap-c)#  dbl
 ! Defines a interactive-video queue with 23% BW remaining + DBL
C4500(config-pmap-c)# class STREAMING-VIDEO-QUEUE
C4500(config-pmap-c)# bandwidth remaining percent 10
C4500(config-pmap-c)#  dbl
 ! Defines a streaming-video queue with 10% BW remaining + DBL
C4500(config-pmap-c)# class CONTROL-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 5
 ! Defines a control/management queue with 5% BW remaining
C4500(config-pmap-c)# class SIGNALING-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 2
 ! Defines a signaling queue with 2% BW remaining
```

```
C4500(config-pmap-c)# class TRANSACTIONAL-DATA-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 24
C4500(config-pmap-c)#  dbl
 ! Defines a transactional data queue with 24% BW remaining + DBL
C4500(config-pmap-c)# class SCAVENGER-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 1
 ! Defines a (minimal) scavenger queue with 1% BW remaining/limit
C4500(config-pmap-c)# class class-default
C4500(config-pmap-c)#  bandwidth remaining percent 25
C4500(config-pmap-c)#  dbl
 ! Provisions the default/Best Effort queue with 25% BW remaining + DBL


 ! This section attaches the egress queuing policy to the interface(s)
C4500(config)# interface range TenGigabitEthernet 1/1-2
C4500(config-if-range)# service-policy output 1P7Q1T
```

You can verify the configuration in Example 15-2 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Twelve-Class Egress Queuing Model

In the 12-class model (illustrated in Figures 11-7 and 11-8), the application class to queue mappings are as follows:

- Voice (marked EF), broadcast video (marked CS5), and real-time interactive traffic (marked CS4) is all assigned to the priority queue (which may be optionally policed to 30 percent bandwidth).
- Multimedia-conferencing traffic (marked AF4) is assigned to a dedicated nonpriority queue with a 10 percent bandwidth allocation with DBL enabled.
- Multimedia-streaming traffic (marked AF3) is assigned to a dedicated nonpriority queue with a 10 percent bandwidth allocation with DBL enabled.
- Network control traffic (marked CS6), signaling traffic (marked CS3) and network management traffic (marked CS2) is all assigned to a dedicated nonpriority queue with a 10 percent bandwidth allocation; optionally, CS7 traffic may also be mapped to this queue.
- Transactional data traffic (marked AF2) is assigned to dedicated nonpriority queue with a 10 percent bandwidth allocation with DBL enabled.

- Bulk data traffic (marked AF1) is assigned to a dedicated nonpriority queue with 4 percent bandwidth allocation with DBL enabled.
- Scavenger traffic (marked CS1) is constrained within a dedicated nonpriority queue with a 1 percent bandwidth allocation.
- Best-effort traffic (marked DF) is assigned to a default queue with 25 percent bandwidth allocation with DBL enabled.

Figure 15-5 illustrates the resulting 12-class (1P7Q1T+DBL) egress queuing model for the Catalyst 4500.

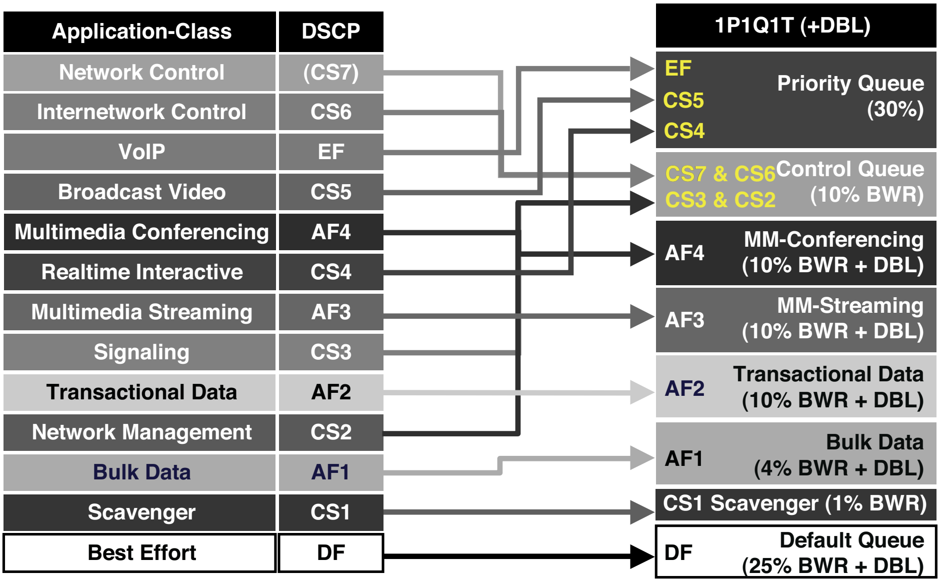


Figure 15-5 *Catalyst 4500 12-Class (1P7Q1T+DBL) Egress Queuing Model*

Example 15-3 shows the corresponding configuration for 12-class (1P7Q1T+DBL) egress queuing on the Catalyst 4500.

Example 15-3 *Twelve-Class (1P7Q1T+DBL) Egress Queuing Configuration Example on a Catalyst 4500*

```
 ! This section configures the class maps for the egress queuing policy
C4500(config)# class-map match-any PRIORITY-QUEUE
C4500(config-cmap)# match dscp ef
C4500(config-cmap)# match dscp cs5
C4500(config-cmap)# match dscp cs4
 ! VoIP (EF), Broadcast Video (CS5) and Realtime Interactive (CS4)
 ! are all mapped to the PQ
```

```
C4500(config)# class-map match-any CONTROL-MGMT-QUEUE
C4500(config-cmap)# match dscp cs7
C4500(config-cmap)# match dscp cs6
C4500(config-cmap)# match dscp cs3
C4500(config-cmap)# match dscp cs2
 ! Network Control (CS7), Internetwork Control (CS6),
 ! Signaling (CS3) and Management (CS2) are mapped
 ! to a Control/Management Queue
C4500(config)# class-map match-all MULTIMEDIA-CONFERENCING-QUEUE
C4500(config-cmap)# match dscp af41 af42 af43
 ! Multimedia Conferencing (AF4) is assigned a dedicated queue
C4500(config)# class-map match-all MULTIMEDIA-STREAMING-QUEUE
C4500(config-cmap)# match dscp af31 af32 af33
 ! Multimedia Streaming (AF3) is assigned a dedicated queue
C4500(config)# class-map match-all TRANSACTIONAL-DATA-QUEUE
C4500(config-cmap)# match dscp af21 af22 af23
 ! Transactional Data (AF2) is assigned a dedicated queue
C4500(config)# class-map match-all BULK-DATA-QUEUE
C4500(config-cmap)# match dscp af11 af12 af13
 ! Bulk Data (AF1) is assigned a dedicated queue
C4500(config)# class-map match-all SCAVENGER-QUEUE
C4500(config-cmap)# match dscp cs1
 ! Scavenger (CS1) is assigned a dedicated queue


 ! This section configures the 1P7Q1T+DBL egress queuing policy map
C4500(config)# policy-map 1P7Q1T
C4500(config-pmap-c)# class PRIORITY-QUEUE
C4500(config-pmap-c)#  priority
 ! Defines a priority queue
C4500(config-pmap-c)# class CONTROL-MGMT-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 10
 ! Defines a control/management queue with 10% BW remaining
C4500(config-pmap-c)# class MULTIMEDIA-CONFERENCING-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 10
C4500(config-pmap-c)#  dbl
 ! Defines a multimedia conferencing queue with 10% BW remaining + DBL
C4500(config-pmap-c)# class MULTIMEDIA-STREAMING-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 10
C4500(config-pmap-c)#  dbl
 ! Defines a multimedia streaming queue with 10% BW remaining + DBL
C4500(config-pmap-c)# class TRANSACTIONAL-DATA-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 10
C4500(config-pmap-c)#  dbl
```

```
! Defines a transactional data queue with 10% BW remaining + DBL
C4500(config-pmap-c)# class BULK-DATA-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 4
C4500(config-pmap-c)#  dbl
 ! Defines a bulk data queue with 10% BW remaining + DBL
C4500(config-pmap-c)# class SCAVENGER-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 1
 ! Defines a (minimal) scavenger queue with 1% BW remaining/limit
C4500(config-pmap-c)# class class-default
C4500(config-pmap-c)#  bandwidth remaining percent 25
C4500(config-pmap-c)#  dbl
 ! Provisions the default/Best Effort queue with 25% BW remaining + DBL


 ! This section attaches the egress queuing policy to the interface(s)
C4500(config)# interface range TenGigabitEthernet 1/1-2
C4500(config-if-range)# service-policy output 1P7Q1T
```

You can verify the configuration in Example 15-3 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface** (as shown in Example 15-4)

Example 15-4 *Verifying Queuing Policies on a Catalyst 4500:* ***show policy-map interface***

```
C4500# show policy-map interface TenGigabitEthernet 1/1
 TenGigabitEthernet1/1
  Service-policy output: 1P7Q1T
    Class-map: PRIORITY-QUEUE (match-any)
      102598 packets
      Match:  dscp ef (46)
        102598 packets
      Match:  dscp cs5 (40)
        0 packets
      Match:  dscp cs4 (32)
        0 packets
      priority queue:
          Transmit: 22782306 Bytes, Queue Full Drops: 0 Packets

    Class-map: CONTROL-MGMT-QUEUE (match-any)
      24847 packets
      Match:  dscp cs7 (56)
        0 packets
      Match:  dscp cs6 (48)
        0 packets
      Match:  dscp cs3 (24)
        24847 packets
      Match:  dscp cs2 (16)
        0 packets
      bandwidth remaining 10 (%)
          Transmit: 24909844 Bytes, Queue Full Drops: 0 Packets


    Class-map: MULTIMEDIA-CONFERENCING-QUEUE (match-all)
      22280511 packets
      Match:  dscp af41 (34) af42 (36) af43 (38)
      bandwidth remaining 10 (%)
          Transmit: 4002626800 Bytes, Queue Full Drops: 0 Packets
      dbl
          Probabilistic Drops: 0 Packets
          Belligerent Flow Drops: 0 Packets


    Class-map: MULTIMEDIA-STREAMING-QUEUE (match-all)
      0 packets
      Match:  dscp af31 (26) af32 (28) af33 (30)
      bandwidth remaining 10 (%)
          Transmit: 0 Bytes, Queue Full Drops: 0 Packets
      dbl
          Probabilistic Drops: 0 Packets
          Belligerent Flow Drops: 0 Packets


    Class-map: TRANSACTIONAL-DATA-QUEUE (match-all)
      235852 packets
      Match:  dscp af21 (18) af22 (20) af23 (22)
      bandwidth remaining 10 (%)
          Transmit: 247591260 Bytes, Queue Full Drops: 0 Packets
      dbl
          Probabilistic Drops: 0 Packets
          Belligerent Flow Drops: 0 Packets


    Class-map: BULK-DATA-QUEUE (match-all)
      2359020 packets
      Match:  dscp af11 (10) af12 (12) af13 (14)
      bandwidth remaining 4 (%)
          Transmit: 2476460700 Bytes, Queue Full Drops: 0 Packets
      dbl
          Probabilistic Drops: 0 Packets
          Belligerent Flow Drops: 0 Packets


    Class-map: SCAVENGER-QUEUE (match-all)
      78607323 packets
      Match:  dscp cs1 (8)
      bandwidth remaining 1 (%)
          Transmit: 98144078642 Bytes, Queue Full Drops: 26268 Packets


    Class-map: class-default (match-any)
      12388183 packets
      Match: any
        12388183 packets
      bandwidth remaining 25 (%)
          Transmit: 13001465825 Bytes, Queue Full Drops: 0 Packets
      dbl
          Probabilistic Drops: 0 Packets
          Belligerent Flow Drops: 0 Packets
C4500#
```

Example 15-4 shows various queuing classes and their associated packet and byte counts, including 26,268 queuing drops noted on the scavenger queue.

Additional Platform-Specific QoS Design Options

These designs represent a generic building block for Catalyst 4500 QoS in a campus distribution switch role, but they are by no means the only design options available to you. Additional options and considerations include the following:

- Access-edge design options
- Per-VLAN QoS design
- Per-port/per-VLAN QoS design
- EtherChannel QoS design
- AutoQoS SRND4
- Control plane policing

Each of these additional QoS design options is discussed in turn.

Access-Edge Design Options

This chapter has focused on QoS designs for the Catalyst 4500 in the role of a campus distribution switch (which are generally equivalent to the QoS designs required were it serving in the role of a campus core switch). However, the Catalyst 4500 can also be deployed as a campus access switch. Therefore, a few additional design options would apply in such a role, including the following access-edge models:

- Conditional Trust Model
- Classification and Marking Model
- Classification, Marking, and Policing Model

Each of these access-edge design options will be discussed in turn.

Conditional Trust Model

As previously mentioned, MQC-based platforms trust at Layer 2 and Layer 3 by default and therefore do not require any explicit commands to perform such functions. Therefore, there are no equivalent commands to **mls qos trust cos** or **mls qos trust dscp (**nor are any required).

However, there is a need to provide conditional trust functionality for all switch platforms that may be deployed in the role of an access switch. Hence, there is a corresponding command for conditional trust on the Catalyst 4500 (namely, **qos trust device**).

At the time of this writing, the Catalyst 4500 supports conditional trust for the following devices:

- Cisco IP phone via the **cisco-phone** keyword option
- Cisco TelePresence systems via the **cts** keyword option
- Cisco IP video surveillance cameras systems via the **ip-camera** keyword option
- Cisco Digital Media Players via the **media-player** keyword option

When extending conditional trust to Cisco IP phones, it is important to remember that these can only re-mark class of service (CoS) bits (on PC-generated traffic). Therefore, the Conditional Trust Model on the Catalyst 4500 requires a dynamic conditional trust policy applied to the port in conjunction with a simple MQC policy that explicitly matches CoS 5 (for voice) and CoS 3 (for signaling) and marks the DSCP values of these packets to EF and CS3, respectively (essentially performing a CoS-to-DSCP mapping). Example 15-5 shows this conditional trust model for the Catalyst 4500.

Example 15-5 *Configuring (CoS-Based) Conditional Trust to a Cisco IP Phone on a Catalyst 4500*

```
 ! This section defines the class maps to match Voice and Signaling
C4500(config-cmap)# class-map match-all VOICE
C4500(config-cmap)#  match cos 5
C4500(config-cmap)# class-map match-all SIGNALING
C4500(config-cmap)#  match cos 3


 ! This section defines the CoS-to-DSCP re-marking policy map
C4500(config-cmap)# policy-map CISCO-IPPHONE
C4500(config-pmap)#  class VOICE
C4500(config-pmap-c)# set dscp ef
 ! Maps CoS 5 to DSCP EF
C4500(config-pmap-c)# class SIGNALING
C4500(config-pmap-c)#  set dscp cs3
 ! Maps CoS 3 to DSCP CS3
C4500(config-pmap-c)# class class-default
C4500(config-pmap-c)#  set dscp default
 ! All other traffic is set to DSCP DF


 ! This section applies conditional trust and policy map to the int(s)
C4500(config)# interface GigabitEthernet 3/1
C4500(config-if)# switchport access vlan 10
C4500(config-if)# switchport voice vlan 110
C4500(config-if)# spanning-tree portfast
C4500(config-if)# qos trust device cisco-phone
 ! Applies conditional-trust to the switch port
C4500(config-if)# service-policy input CISCO-IPPHONE
 ! Attaches the CoS-to-DSCP mapping policy map
```

You can verify the configuration in Example 15-5 with the following commands:

- show qos interface
- show class-map
- show policy-map
- show policy-map interface

Medianet Metadata Classification Model

Beginning with Cisco IOS Release IOS XE 3.3.0SG and IOS 15.1(1)SG, you can configure a class map with metadata filters. A QoS policy that includes such classes is termed a metadata-based QoS policy. It allows you to classify flows based on user-friendly metadata attributes rather than on access control list (ACL)-based classification criteria (such as source/destination addresses/ports, and so on).

The following restrictions apply to using a metadata-based QoS policy on a Catalyst 4500 series switch:

- They can only be attached to target in input direction.
- They can only be attached to physical ports and EtherChannel port channel interfaces; they cannot be attached to VLANs, port VLANs, and switch virtual interfaces (SVIs).
- A policy can have multiple metadata-based classifiers.
- A class map can have one or more metadata filters with **match-any** or **match-all** semantics.
- Policy actions corresponding to metadata class are applied on aggregate traffic; however, if the metadata filter is configured along with Flexible NetFlow record filter, the policy action (like policer) applies on individual flows.

Note Flow-based QoS policies and Flexible NetFlow (FNF) are discussed further in a following section.

Example 15-6 illustrates a metadata-based QoS policy with two classes using metadata filters.

Example 15-6 *Medianet Metadata Classification Policy Example on a Catalyst 4500*

```
 ! This section configures the medianet metadata class maps
C4500(config-cmap)# class-map match-all REALTIME-INTERACTIVE
C4500(config-cmap)#  match application telepresence-media
 ! Identifies TelePresence media flows via metadata
C4500(config-cmap)# class-map match-any MULTIMEDIA-CONFERENCING
C4500(config-cmap)#  match application webex-video
 ! Identifies WebEx video flows via metadata
C4500(config-cmap)#  match application webex-voice
 ! Identifies WebEx voice flows via metadata
```

You can verify the configuration in Example 15-6 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Classification and Marking Models

In many scenarios, trust models may not be available or sufficient to distinctly classify all types of traffic required by the end-to-end QoS strategic model. Therefore, explicit classification and marking policies may be needed at the access edge.

Example 15-7 shows a configuration example based on Figure 11-5 (An eight-class QoS model).

Note As previously discussed, not all application classes may be present at the access edge on ingress. For example, streaming video would likely not be present at the access edge on ingress (as these flows are not *sourced* from campus endpoints, but are likely *destined* to them), nor would network control flows be sourced from campus endpoints. Therefore, these classes would not need to be included in the access-edge classification and marking policy map.

Note Referenced access lists are omitted from the policy examples for brevity.

Example 15-7 *Classification and Marking Policy Example on a Catalyst 4500*

```
 ! This section configures the class maps
C4500(config-cmap)# class-map match-all VOICE
C4500(config-cmap)#  match dscp ef
 ! Voice is matched on DSCP EF
C4500(config-cmap)# class-map match-all INTERACTIVE-VIDEO
C4500(config-cmap)#  match access-group name INTERACTIVE-VIDEO
 ! Associates INTERACTIVE-VIDEO access-list with class map
C4500(config-cmap)# class-map match-all SIGNALING
C4500(config-cmap)#  match cs3
 ! Signaling is matched on DSCP CS3
C4500(config-cmap)# class-map match-all TRANSACTIONAL-DATA
C4500(config-cmap)#  match access-group name TRANSACTIONAL-DATA
 ! Associates TRANSACTIONAL-DATA access-list with class map
C4500(config-cmap)# class-map match-all SCAVENGER
C4500(config-cmap)#  match access-group name SCAVENGER
 ! Associates SCAVENGER access-list with class map
 ! This section configures the Per-Port ingress marking policy map
C4500(config-cmap)# policy-map PER-PORT-MARKING
C4500(config-pmap)#  class VOICE
C4500(config-pmap-c)# set dscp ef
 ! VoIP is marked EF
C4500(config-pmap-c)# class INTERACTIVE-VIDEO
C4500(config-pmap-c)#  set dscp af41
 ! Interactive-Video is marked AF41
C4500(config-pmap-c)# class SIGNALING
C4500(config-pmap-c)#  set dscp cs3
 ! Signaling is marked CS3
C4500(config-pmap-c)# class TRANSACTIONAL-DATA
C4500(config-pmap-c)#  set dscp af21
 ! Transactional Data is marked AF21
C4500(config-pmap-c)# class SCAVENGER
C4500(config-pmap-c)#  set dscp cs1
 ! Scavenger traffic is marked CS1
C4500(config-pmap-c)# class class-default
C4500(config-pmap-c)#  set dscp default
 ! All other traffic is marked DF


 ! This section attaches the service-policy to the interface(s)
C4500(config)# interface range GigabitEthernet 2/1-48
C4500(config-if-range)# switchport access vlan 10
C4500(config-if-range)# switchport voice vlan 110
C4500(config-if-range)# spanning-tree portfast
C4500(config-if-range)# qos trust device cisco-phone
 ! The interface is set to conditionally trust Cisco IP Phones
C4500(config-if-range)# service-policy input PER-PORT-MARKING
 ! Attaches the Per-Port Marking policy to the interface(s)
```

You can verify the configuration in Example 15-7 with the following commands:

- **show qos interface**
- **show class-map**
- **show policy-map**
- **show policy-map interface**

Classification, Marking, and Policing Model

In addition to classification and marking, policing might also be required at the access edge. The Catalyst 4500 can perform single-rate (two-color) policing and three-color policing—via either the RFC 2697 single-rate three-color marker (srTCM) or the RFC 2698 two-rate three-color marker (trTCM). Example 15-8 shows a per-port single-rate policing example for the Catalyst 4500 (based on Figure 13-8), and Example 15-9 shows policy amendments to support a RFC 2698 two-rate three-color marker.

Example 15-8 *(Single-Rate Two-Color) Per-Port Policing Configuration Example on a Catalyst 4500*

```
 ! This section configures the single-rate per-port policing policy map
C4500(config)# policy-map PER-PORT-POLICING
C4500(config-pmap)# class VVLAN-VOIP
C4500(config-pmap-c)#  set dscp ef
C4500(config-pmap-c)#  police 128k bc 8000
C4500(config-pmap-c-police)# conform-action transmit
C4500(config-pmap-c-police)# exceed-action drop
 ! VoIP is marked EF and policed to drop at 128 kbps
C4500(config-pmap)# class VVLAN-SIGNALING
C4500(config-pmap-c)#  set dscp cs3
C4500(config-pmap-c)#  police 32k bc 8000
C4500(config-pmap-c-police)# conform-action transmit
C4500(config-pmap-c-police)# exceed-action drop
 ! (VVLAN) Signaling is marked CS3 and policed to drop at 32 Kbps
C4500(config-pmap)# class MULTIMEDIA-CONFERENCING
C4500(config-pmap-c)#  set dscp af41
C4500(config-pmap-c)#  police 5m bc 8000
C4500(config-pmap-c-police)# conform-action transmit
C4500(config-pmap-c-police)# exceed-action drop
 ! Multimedia-conferencing is marked AF41 and policed to drop at 5 Mbps
C4500(config-pmap)# class SIGNALING
C4500(config-pmap-c)#  set dscp cs3
C4500(config-pmap-c)#  police 32k bc 8000
C4500(config-pmap-c-police)# conform-action transmit
C4500(config-pmap-c-police)# exceed-action drop
 ! (DVLAN) Signaling is marked CS3 and policed to drop at 32 Kbps
C4500(config-pmap)# class TRANSACTIONAL-DATA
C4500(config-pmap-c)#  set dscp af21
C4500(config-pmap-c)#  police 10m bc 8000
C4500(config-pmap-c-police)# conform-action transmit
C4500(config-pmap-c-police)# exceed-action set-dscp-transmit af22
 ! Trans-data is marked AF21 and policed to re-mark (to AF22) at 10 Mbps
C4500(config-pmap)# class BULK-DATA
C4500(config-pmap-c)#  set dscp af11
C4500(config-pmap-c)#  police 10m bc 8000
C4500(config-pmap-c-police)# conform-action transmit
C4500(config-pmap-c-police)# exceed-action set-dscp-transmit af12
```

```
 ! Bulk-data is marked AF11 and policed to re-mark (to AF12) at 10 Mbps
C4500(config-pmap)# class SCAVENGER
C4500(config-pmap-c)#  set dscp cs1
C4500(config-pmap-c)#  police 10m bc 8000
C4500(config-pmap-c-police)# conform-action transmit
C4500(config-pmap-c-police)# exceed-action drop
 ! Scavenger traffic is marked CS1 and policed to drop at 10 Mbps
C4500(config-pmap)# class class-default
C4500(config-pmap-c)#  set dscp default
C4500(config-pmap-c)#  police 10m bc 8000
C4500(config-pmap-c-police)# conform-action transmit
C4500(config-pmap-c-police)# exceed-action set-dscp-transmit cs1
 ! The implicit default class marks all other traffic to DF
 ! and polices all other traffic to re-mark (to CS1) at 10 Mbps


 ! This section attaches the service-policy to the interface(s)
C4500(config)# interface range GigabitEthernet 2/1-48
C4500(config-if-range)# switchport access vlan 10
C4500(config-if-range)# switchport voice vlan 110
C4500(config-if-range)# spanning-tree portfast
C4500(config-if-range)# qos trust device cisco-phone
 ! The interface is set to conditionally trust Cisco IP phones
C4500(config-if-range)# service-policy input PER-PORT-POLICING
 ! Attaches the Per-Port Policing policy to the interface(s)
```

Note The Catalyst 4500 IOS Software allows for policing rates to be entered using the postfixes **k** (for kilobits), **m** (for megabits), and **g** (for gigabits), as shown in Example 15-8. In addition, decimal points are allowed in conjunction with these postfixes. For example, a rate of 10.5 Mbps could be entered with the policy map command **police 10.5m**. These policing rates are converted to their full bits-per-second values within the configuration, but it makes the entering of these rate more user friendly and less error prone (as could easily be the case when having to enter up to 10 zeros to define the policing rate).

You can verify the configuration in Example 15-8 with the following commands:

- **show qos interface**
- **show class-map**
- **show policy-map**
- **show policy-map interface**

To avoid excessive repetition, Example 15-9 amends and expands the policer from a single-rate two-color marker to a two-rate three-color marker only on a single class (the Bulk Data class). However, similar amendments can be made on any Assured Forwarding (AF) class of traffic.

Example 15-9 *(Two-Rate Three-Color) Per-Port Policing Configuration Amendment Example on a Catalyst 4500*

```
 ! This section configures a dual-rate per-port policing policy map
C4500(config)# policy-map TWO-RATE-POLICER

<snip>

C4500(config-pmap)# class BULK-DATA
C4500(config-pmap-c)# set dscp af11
C4500(config-pmap-c)# police 10m bc 8000 pir 15m
  ! Bulk-data is policed to 10 Mbps rate and 15 Mbps peak rate
C4500(config-pmap-c-police)# conform-action set-dscp-transmit af11
 ! Bulk data under 10 Mbps will be marked AF11
C4500(config-pmap-c-police)# exceed-action set-dscp-transmit af12
 ! Bulk data traffic between 10 Mbps and 15 Mbps will be marked AF12
C4500(config-pmap-c-police)# violate-action set-dscp-transmit af13
 ! Bulk data traffic over 15Mbps will be marked AF13
```

You can verify the configuration in Example 15-9 with the following commands:

- **show qos interface**
- **show class-map**
- **show policy-map**
- **show policy-map interface**

Per-VLAN QoS Design

The Catalyst 4500 supports VLAN-based QoS. However, unlike the Catalyst 3750, the Catalyst 4500 does not support the **mls qos vlan-based** interface command. Furthermore, service policies are attached to VLANs via the VLAN configuration mode (instead of the interface configuration mode), as shown in Example 15-10.

Example 15-10 *Per-VLAN Marking Configuration Example on a Catalyst 4500*

```
 ! This section configures the interface(s) for conditional trust,
C4500(config)# interface range GigabitEthernet 2/1-48
C4500(config-if-range)# switchport access vlan 10
```

```
C4500(config-if-range)# switchport voice vlan 110
C4500(config-if-range)# spanning-tree portfast
C4500(config-if-range)# qos trust device cisco-phone
 ! The interface is set to conditionally trust Cisco IP phones


 ! This section attaches a marking policy to the DVLAN
C4500(config)# vlan config 10
C4500(config-vlan-config)# service-policy input DVLAN-MARKING


 ! This section attaches a marking policy to the VVLAN
C4500(config)# vlan config  110
C4500(config-vlan-config)# service-policy input VVLAN-MARKING
```

You can verify the configuration in Example 15-10 with the following commands:

- **show qos interface**
- **show class-map**
- **show policy-map**
- **show policy-map vlan** *vlan-number* (This command is nearly identical to show policy map interface, except that it references a VLAN directly, rather than a VLAN interface.)

Note It is not recommended to deploy policing policies on a per-VLAN basis, as discussed further in the next section.

Per-Port/Per-VLAN QoS

Although it is technically possible to apply a (aggregate) policing policy on a per-VLAN basis, it is not advisable to do so. This is because the number of endpoints in a given VLAN can dynamically vary, yet the policing rates are statically fixed at an aggregate level, resulting in unpredictable bandwidth allotments per endpoint.

However, a more flexible and discrete approach for deploying policing policies exists on the Catalyst 4500 platforms—namely, to deploy these on a per-port/per-VLAN basis. The Catalyst 4500 has a very elegant syntax for deploying per-port/per-VLAN policies, as follows: Within a (trunked) switch port's interface configuration, each VLAN carried over that trunked port can have a separate policy applied to it via an **interface-vlan** configuration mode, as shown in Example 15-11.

Example 15-11 *Per-Port/Per-VLAN Policing Configuration Example on a Catalyst 4500*

```
! This section attaches the policy to the VLANs on a per-port basis
C4500(config)# interface range GigabitEthernet 2/1-48
C4500(config-if-range)# switchport access vlan 10
C4500(config-if-range)# switchport voice vlan 110
C4500(config-if-range)# spanning-tree portfast
C4500(config-if-range)# qos trust device cisco-phone
 ! The interface is set to conditionally trust Cisco IP phones
C4500(config-if-range)# vlan 10
C4500(config-if-vlan-range)# service-policy input DVLAN-POLICERS
 ! Attaches the per-port/per-VLAN DVLAN policing policy to the
 ! DVLAN of the trunked switch port(s)
C4500(config-if-range)# vlan 110
C4500(config-if-vlan-range)# service-policy input VVLAN-POLICERS
 ! Attaches the per-port/per-VLAN VVLAN policing policy to the
 ! VVLAN of the trunked switch port(s)
```

You can verify the configuration in Example 15-11 with the following commands:

- **show qos interface**
- **show class-map**
- **show policy-map**
- **show policy-map interface**
- **show policy-map interface** *interface x/y* **vlan** *vlan-number*

EtherChannel QoS Design

The following rules apply when deploying QoS service policies on Catalyst 4500 EtherChannels:

- Classification, marking, and policing policies (whether ingress or egress) are applied to the logical port channel interfaces.
- Queuing policies are applied to the physical port-member interfaces.

For EtherChannel interfaces configured on Catalyst 4500 switches, *the ingress QoS policies* (including classification, marking, and policing policies) are applied via MQC **service-policy** statements (in the ingress direction using the **input** keyword) configured on the *logical port channel interface*. Trust statements are not required because this MQC-based platform trusts by default.

In addition, the Catalyst 4500 supports *egress QoS policies* (including marking/policing policies) to be similarly applied via MQC **service-policy** statements (in the egress direction using the **output** keyword) on the *logical port channel interface*.

Egress queuing policies, however, are applied via MQC **service-policy** statements (in the egress direction using the **output** keyword) on the *physical port-member interfaces*, as shown in Example 15-12.

Example 15-12 *EtherChannel QoS Design on a Catalyst 4500*

```
! This section configures the logical port channel interface
C4500(config)# interface Port-channel1
C4500(config-if)# description ETHERCHANNEL-LOGICAL-INTERFACE
C4500(config-if)# switchport mode trunk
C4500(config-if)# switchport trunk encapsulation dot1q
C4500(config-if)# switchport trunk allowed vlan 10,110
C4500(config-if)# service-policy input MARKING


 ! This section configures 1P3Q1T+DBL queuing on physical port-member interfaces
C4500(config)# interface range TenGigabitEthernet1/1-2
C4500(config-if-range)# description PORT-CHANNEL1-PORT-MEMBER
C4500(config-if-range)# switchport mode trunk
C4500(config-if-range)# switchport trunk encapsulation dot1q
C4500(config-if-range)# switchport trunk allowed vlan 10,110
C4500(config-if-range)# channel-group 1 mode auto
C4500(config-if-range)# service-policy output 1P7Q1T-QUEUING
 ! Applies 1P7Q1T+DBL-QUEUING queuing policy to physical port member
```

You can verify the configuration in Example 15-12 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Note As previously stated, the queuing policies will only attach to EtherChannel port-member physical interfaces if the priority queue is not explicitly policed. If policing the priority queue is desired, a separate policy map needs to be constructed to do so and attached to the logical EtherChannel interface in the *egress* direction.

Flow-Based QoS

Flow-based QoS enables microflow policing and marking capability to dynamically learn traffic flows, providing the capability to police every unique flow to an individual rate. Flow-based QoS is available on a Catalyst 4500 series switch with the built-in NetFlow hardware support. It can be applied to ingress traffic on both switched and routed interfaces with flow masks defined using Flexible NetFlow (FNF). Flow-based QoS is typically used in environments where per-user, granular rate limiting is required. Flow-based QoS is also referred to as user-based rate limiting (UBRL).

A *flow* is defined as a stream of packets having the same properties as those defined by the key fields in the FNF flow record. A new flow is created when the value of data in packet's key fields is unique with respect to the flows that already exist.

A flow-based QoS policy is possesses one or more class maps matching on a FNF flow record. Such a class map must be configured as **match-all** to match all the match criteria specified in the class map. When a flow-based QoS policy is attached to a QoS target, ingress traffic on the target is first classified based on the classification rules specified in the class map. If the classifier has an FNF flow record, the key fields specified in the FNF flow record are applied on the classified traffic to create flows provided the flow does not already exist. The corresponding policy actions (policing and marking) are then applied to these individual flows. Flow-based policers (termed microflow policers) rate limit each unique flow. Flows are dynamically created and inactive flows are periodically aged out.

Flow-based QoS policy can be applied on a per-port basis, per-port/per-VLAN basis, or on an EtherChannel port channel interface (but only in the ingress direction). Therefore, flow-based QoS may be deployed at either the access layer or distribution layer (wherever UBRL may be of value).

Note that flow-based policies will apply to all flows matched within a given class. For example, if a flow-based policer is applied to the default class and attached to port or VLAN, *all* flows originating from that port or VLAN (respectively) will be subject to the policer. If this is not to be the intent, additional classification is recommended and the flow-based policer should be more selectively applied.

Example 15-13 shows how to configure a flow-based QoS policy that uses microflow policing in the context of user-based rate limiting. Any and all flows sourced from the subnet 192.168.10.* are microflow policed to 1 Mbps.

Example 15-13 *Configuring Flow-Based QoS (UBRL) on Catalyst 4500*

```
 ! This section defines an ACL to match traffic from subnet
C4500(config)# ip access-list extended USERGROUP-1
C4500(config-ext-nacl)# permit ip 192.168.10.0 0.0.0.255 any
 ! Traffic sourced from the 1922.168.10.x subnet is matched
 ! This section defines a flow record with source address as key
C4500(config)# flow record FLOW-RECORD-1
C4500(config-flow-record)# match ipv4 source address
 ! Source address is defined as the key tuple


 ! This section defines the class map to match on USERGROUP-1 ACL
 ! and specify FLOW-RECORD-1 definition for flow creation
C4500(config)# class-map match-all USER-GROUP-1
C4500(config-cmap)# match access-group name USERGROUP-1
C4500(config-cmap)# match flow record FLOW-RECORD-1
 ! A "match-all" class map binds the ACL and flow-record
 ! to identify unique flows


 ! This section defines the microflow policer policy map
C4500(config)# policy-map 1MBS-MICROFLOW-POLICER
C4500(config-pmap)# class USER-GROUP-1
C4500(config-pmap-c)# police cir 1m
C4500(config-pmap-c-police)# conform-action transmit
C4500(config-pmap-c-police)# exceed-action drop
 ! Specifies each discrete microflow is to be limited to 1Mbs


 ! This section applies the microflow policer to the interface
C4500(config)# interface gigabitEthernet3/1
C4500(config-if)# service-policy input 1MBS-MICROFLOW-POLICER
```

You can verify the configuration in Example 15-13 with the following commands:

- **show flow record** (demonstrated in Example 15-14)
- **show class-map**
- **show policy-map**
- **show policy-map interface**

Example 15-14 *Verifying Flow-Based QoS Policies on a Catalyst 4500:* ***show flow record***

```
C4500# show flow record
flow record FLOW-RECORD-1:
  Description:        User defined
  No. of users:       1
```

```
Total field space:  4 bytes
Fields:
  match ipv4 source address
```

AutoQoS SRND4

AutoQoS SRND4 is supported on the Cisco Catalyst 4500 beginning with Cisco IOS Release IOS XE 3.3.0SG and IOS 15.1(1)SG and is detailed in Appendix A, "AutoQoS for Medianet."

Control Plane Policing

Control plane policing (CPP) is supported on the Catalyst 4500 and is detailed in Appendix B, "Control Plane Policing."

Summary

This design chapter primarily discussed the best-practice QoS design recommendations for the Cisco Catalyst 4500 (Supervisor 6-E/7-E) series switch in the role of a campus distribution layer switch. (which, incidentally are equivalent to the QoS designs required were it serving in the role of a campus core switch).

Because the Catalyst 4500 is an MQC-based QoS platform, QoS is enabled by default, as is DSCP trust, on all ports. Therefore, there is effectively only a single step to configuring QoS on a Catalyst 4500 performing the role of a distribution switch: to configure an egress queuing policy.

To this end, 4-class, 8-class, and 12-class queuing policies were detailed, along with corresponding configurations and verification examples, leveraging the Catalyst 4500's flexible 1P7Q1T+DBL hardware queuing capabilities.

Additional platform-specific design options and considerations were discussed, including how the Catalyst 4500 could be deployed as an access-edge switch, and how to configure per-VLAN QoS, per-port/per-VLAN QoS, and EtherChannel QoS designs.

AutoQoS SRND4 is supported on the Catalyst 4500 and is covered in Appendix A; similarly, CPP is also supported and is covered in Appendix B.

Further Reading

Cisco Enterprise Medianet Campus QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html

Medianet Campus QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampusaag.html

Medianet Catalyst 4500 QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampuscat4500aag.html

Cisco Catalyst 4500 Series Switch Software Configuration Guide, Release IOS XE 3.3.0SG and IOS 15.1(1)SG—QoS Configuration Guide: http://www.cisco.com/en/US/docs/switches/lan/catalyst4500/15.1/XE_330SG/configuration/guide/qos_mrg.html

Chapter 16

Campus Core (Cisco Catalyst 6500) QoS Design

The primary role of quality of service (QoS) in the campus core switch is to manage packet loss. Therefore, the core switch should trust differentiated services code point (DSCP) markings on ingress (because these have been previously set by access-edge switches) and perform both ingress and egress queuing, as illustrated in Figure 16-1.

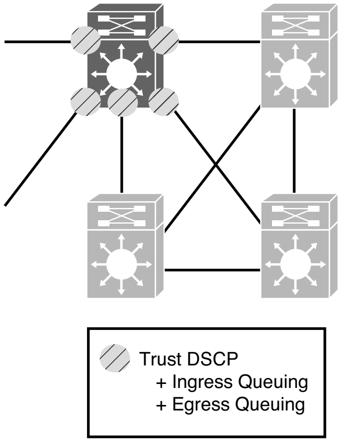


Figure 16-1 *Campus Core Switch Port QoS Roles*

The Cisco Catalyst 6500 Supervisor 2T is a platform well suited to the role of a campus core switch and is featured in this design chapter.

Note The design recommendations in this chapter apply to Supervisor 2T systems, whether these are Catalyst 6500 series chassis or the newer Catalyst 6800 series chassis.

Incidentally, the QoS design requirements of a Catalyst 6500 Supervisor 2T (hereafter referred to simply as Catalyst 6500) in the role of a core switch are generally equivalent to the requirements of a campus distribution switch.

Cisco Catalyst 6500 QoS Architecture

QoS on the Catalyst 6500 is performed both in a centralized Policy Feature Card (PFC) and on the individual modular linecards. The Catalyst 6500 Supervisor 2T uses the fourth-generation PFC (PFC4). QoS processing for the PFC4-based linecards can be split into three processing steps, with each step occurring at a different point in the system. Figure 16-2 illustrates these three steps.

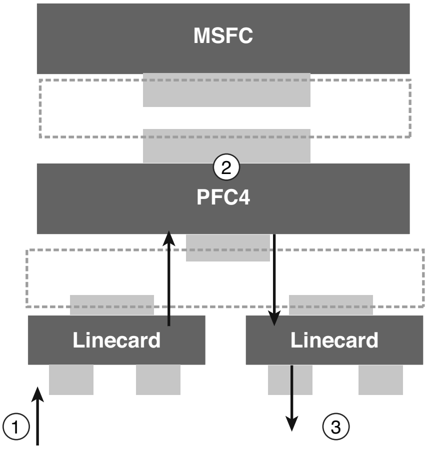


Figure 16-2 *Cisco Catalyst 6500 QoS Processing in a PFC4-Based Linecard*

1. Ingress QoS is performed on the ingress linecard port; ingress QoS functions include input queue scheduling and congestion avoidance.
2. Centralized QoS is performed on the PFC4; centralized QoS functions include port trust, marking, classification, QoS access control lists (ACLs), and policing.
3. Egress QoS is performed on the egress linecard port; egress QoS functions include queue scheduling, congestion avoidance, and, in some linecards, shaping.

In addition, one of the limitations of previous PFCs is that decisions are made in multiple cycles, thereby adding latency to the whole forwarding process. PFC4 makes many of these decisions in a single pass, albeit by going through the Layer 2 and Layer 3 components in a step-by-step process. The component that performs Layer 3 and QoS functionalities is implemented in a pipeline mode, with each stage in the pipeline performing a specific task. The two logical pipelines that make up the physical pipeline are the Input Forwarding Engine (IFE) and Output Forwarding Engine (OFE), as illustrated in Figure16-3.

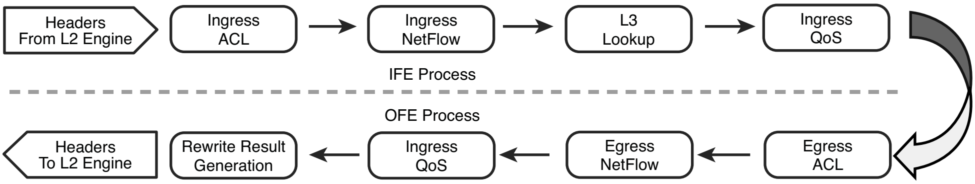


Figure 16-3 *Cisco Catalyst 6500 IFE and OFE Processes*

Beyond extended functional capabilities and operational efficiency, a key advantage that the PFC4 includes over its predecessors is that it supports configuration in the Cisco Common Classification Policy Language (C3PL), which is a platform-independent syntax that essentially makes Multi-Layer Switch (MLS) QoS commands appear more like Modular QoS command-line interface (MQC) commands.

For instance, because of the differences in queuing implementation in port application-specific integrated circuits (ASICs), MQC compliance in PFC3-based Catalyst 6500 systems (such as Supervisor 720) was limited to marking and policing. Queuing configurations, however, remained platform specific and interface specific. However, the PFC4 removes these limitations by supporting C3PL, which is a policy-driven CLI syntax, for marking, policing, *and* queuing. Therefore, C3PL serves to significantly simplify resulting QoS configuration, particularly when consistent queuing policies are applied to many ports/interfaces (because each interface then requires simply a pair of **service-policy** statements to enable ingress and egress queuing policies, rather than dozens of lines of queuing policies *per interface*).

This change to C3PL also affects the default QoS behavior for the PFC4. The main changes to default QoS behavior on the PFC4 (as compared to previous generations of PFCs) can be broadly summarized as follows:

- QoS is enabled by default (so no need to explicitly enable QoS with the **mls qos** global command, as with previous supervisors).
- QoS for an interface is always defined by the attached service policies.
- By default, packets are passed through the switch without a change in differentiated services code point (DSCP), expedited (EXP), or class of service (CoS) for Layer 2 (L2) packets or L2-classified Layer 3 (L3) packets (as compared with all ports being set to an untrusted state, which was the case with previous supervisors once QoS was globally enabled).
- Service policy marking does not depend on port trust.
- By default, the port-level QoS is disabled, and the port-level ingress queue scheduling and congestion avoidance is CoS based.

Note The PFC3-based **mls qos** global command is replaced with the **auto qos default** global command, which is used for enabling QoS just at the port level and not at the PFC level. This command—incidentally—is not be confused with the AutoQoS feature commands discussed in previous chapters; AutoQoS for Medianet is not supported on the Catalyst 6500 (at the time of this writing).

In addition, in the Catalyst 6500 Supervisor 2T, port trust is now defined in the PFC4/DFC4, instead of being taken from the port ASIC. The Layer 3 forwarding logic will assign a 6-bit Discard Class value (which is another name for the internal DSCP discussed in Chapter 13, "Campus QoS Design Considerations and Recommendations"). This 6-bit Discard Class value is an internal tag used only for QoS processing within the switch.

Specific to marking, the following important changes for PFC4 behavior also apply:

- QoS global maps defined using C3PL table map syntax.
- DSCP is preserved by default, independent of port state.
- CoS is preserved by default for Layer 2 packets, independent of port state.
- Port **trust dscp** command is eliminated.

Therefore, the PFC4 provides the ability to prioritize a packet without rewriting the IP packet, which is referred to as DSCP transparency, which can be controlled on a per-policy class basis.

QoS Design Steps

As shown in Figure 16-1, there are three QoS policy requirements of a core switch:

- Trust DSCP on ingress
- Queue on ingress
- Queue on egress

However, because of the default C3PL behavior of the PFC4 (which trusts DSCP by default), there are effectively only two steps to configuring QoS on a Catalyst 6500 in the role of a campus core switch:

1. Configure ingress queuing.
2. Configure egress queuing.

Queuing Models

The Catalyst 6500 supports both ingress and egress queuing models, which vary by Supervisor and linecards, as summarized in Table 16-1 through Table 16-4.

Table 16-1 *Cisco 6500 Supervisor Engine Module QoS Queue Structures*

Supervisor Engines	Ingress Queue and Drop Thresholds	Ingress Queue Scheduler	Egress Queue and Drop Thresholds	Egress Queue Scheduler	Total Buffer Size	Ingress Buffer Size	Egress Buffer Size
VS-S2T-10G-XL, VS-S2T-10G							
With Gigabit Ethernet ports enabled	2q4t	WRR	1p3q4t	DWRR or SRR	240 MB	128 MB	112 MB
	Does not support DSCP-based queuing						
With Gigabit Ethernet ports disabled	8q4t	WRR	1p7q4t	DWRR or SRR	240 MB	128 MB	112 MB
	Supports DSCP-based queuing.						

Table 16-2 *Cisco 6500 40-Gigabit Ethernet Modules QoS Queue Structures*

Modules	Ingress Queue and Drop Thresholds	Ingress Queue Scheduler	Egress Queue and Drop Thresholds	Egress Queue Scheduler	Total Buffer Size	Ingress Buffer Size	Egress Buffer Size
WS-X6904-40G-2TXL, WS-X6904-40G-2T (supports DSCP-based queuing)	1p7q4t or 2p6q4t	DWRR	1p7q4t or 2p6q4t	DWRR	Port groups 1and 2 have a combined *ingress* packet buffer of 10 MB, the same as the port groups 3 and 4. Port groups 1 and 2 have a combined *egress* packet buffer of 176 MB, the same as port groups 3 and 4. Distribution of packet buffers among the ports in Performance or Oversubscribed mode is explained in more detail here. The *ingress* buffer for port groups 1 and 2 or port groups 3 and 4 is as follows: In Performance mode, 1ms of storage = 10 MB or 10 MB / 4 for each of the 4 × 10-GE ports. In Oversubscribed mode = 10 MB / 2 for each of the 2 × 40-GE ports or (10 MB / 4) / 2) for each of the 8 × 10-GE ports. The *egress* buffer for port groups 1 and 2 or port groups 3 and 4 is as follows: In Performance mode, 10 ms of storage equals 176 MB or 176 MB / 4 for each of the 4 × 10-GE ports, In Oversubscribed mode, 10 ms of storage equals 176 MB / 2 for each of 2 × 40 GE ports and/or (176 / 4) / 2) for each of 8 × 10-GE ports.		

Table 16-3 *Cisco 6500 10-Gigabit Ethernet Modules QoS Queue Structures*

Modules	Ingress Queue and Drop Thresholds	Ingress Queue Scheduler	Egress Queue and Drop Thresholds	Egress Queue Scheduler	Total Buffer Size	Ingress Buffer Size	Egress Buffer Size
WS-X6908-10GE (supports DSCP-based queuing)	8q4t	DWRR	1p7q4t	DWRR SRR	200 MB	108 MB	90 MB
WS-X6816-10T-2T, WS-X6716-10T, WS-X6816-10G-2T, WS-X6716-10GE (supports DSCP-based queuing)							
Performance mode	8q4t	DWRR	1p7q4t	DWRR SRR	198 MB	108 MB per port	90 MB per port
Oversubscription mode	1P7Q4T	DWRR	1p7q4t	DWRR SRR	91 MB	90 MB per port	1 MB per port group
WS-X6704-10GE	8q8t	WRR	1p7q8t	DWRR	16 MB	2 MB	14 MB

Table 16-4 *Cisco 6500 Gigabit and 10/100/1000 Ethernet Modules QoS Queue Structures*

Modules	Ingress Queue and Drop Thresholds	Ingress Queue Scheduler	Egress Queue and Drop Thresholds	Egress Queue Scheduler	Total Buffer Size	Ingress Buffer Size	Egress Buffer Size
WS-X6848-TX-2T, WS-X6748-GE-TX, WS-X6848-SFP-2T, WS-X6748-SFP, WS-X6824-SFP-2T, WS-X6724-SFP							
	2q8t	WRR	1p3q8t	DWRR	1.3 MB	166 KB	1.2 MB

The most common queuing structures across these Supervisors and linecards (excluding the GE and 10/100/1000 linecards, which are rarely used in a campus core context anyways) is 8Q4T for ingress and 1P7Q4T for egress. The subsequent sections examine how to adapt these common queuing structures for the 4-class, 8-class, and 12-class QoS models that have been presented throughout this book.

Because the queuing configuration syntax of C3PL is similar to MQC, not every queue needs to be defined and provisioned. In other words, even though eight hardware queues are available, a four-class QoS model only needs four to be defined.

In addition, because the number of ingress queues exactly matches the number of egress queues, it is efficient to configure ingress and egress queuing policies at the same time, because the class maps can be used/reused for each queuing policy map, as shown in the configuration examples that follow.

Four-Class (4Q4T Ingress and 1P3Q4T Egress) Queuing Models

In the four-class model (illustrated in Figures 11-3 and 11-4), the application class to queue mappings are as follows:

- Real-time traffic (marked EF) is assigned to the real-time queue, which is allocated 30 percent bandwidth in the ingress 4Q4T model, but is enabled with strict-priority queuing in the egress 1P3Q4T model.
- Control traffic (marked CS3) is assigned to a dedicated nonpriority queue with a 10 percent bandwidth/bandwidth-remaining allocation.
- Transactional data (marked AF2) is assigned to another dedicated nonpriority queue with a 35 percent bandwidth/bandwidth-remaining allocation with DSCP-based weighted random early detection (WRED) enabled.
- Best effort traffic (marked DF) is assigned to the default queue (which will have at least 25 percent bandwidth implicitly allotted to it) with WRED enabled.

Note WRED is enabled *only* on the transactional data queue and the default queue (because real-time traffic and control traffic should never be early dropped).

Note When the priority queue is configured on one class of a policy map, only **bandwidth remaining percent** is accepted on other classes (guaranteeing a minimum bandwidth for other classes from the remaining bandwidth of what is left after using the priority queue).

Note Although it is true that there will be fractional differences in bandwidth allotments to an application class depending on whether **bandwidth percent** or **bandwidth remaining percent** is used, these differences are relatively minor. Therefore, for the sake of consistency, the same numeric values are used for both variations in the following examples. However, you can always change these values such that the allotted bandwidth matches exactly.

Figure 16-4 illustrates the resulting four-class (4Q4T ingress and 1P3Q4T egress) queuing models for the Catalyst 6500.

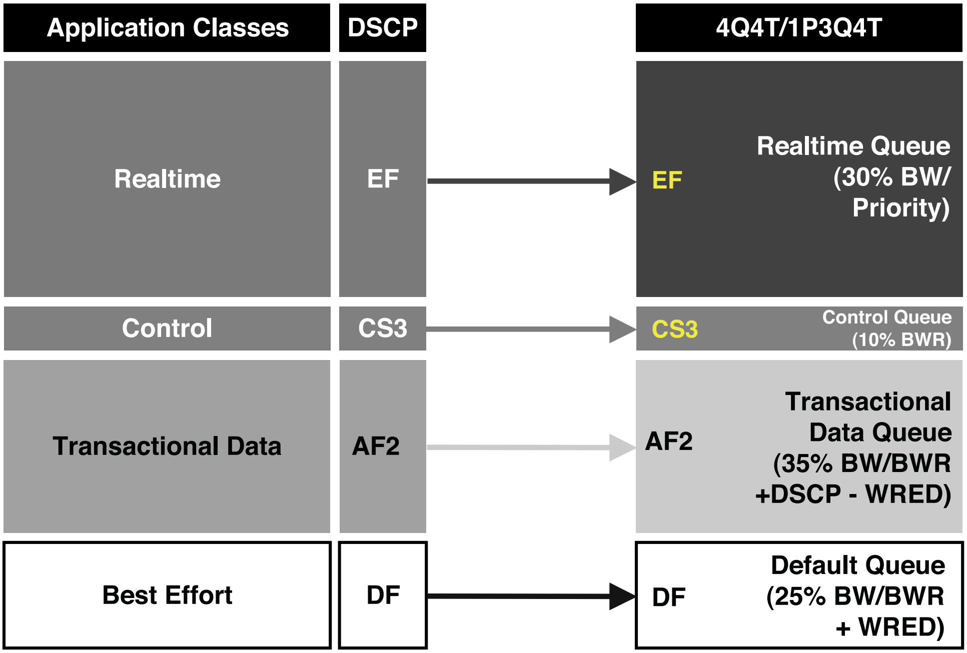


Figure 16-4 *Catalyst 6500 Four-Class (4Q4T Ingress and 1P3Q4T Egress) Queuing Model Examples*

Example 16-1 shows the corresponding configuration for four-class (4Q4T ingress and 1P3Q4T egress) queuing models for the Catalyst 6500.

Example 16-1 *Four-Class (4Q4T Ingress and 1P3Q4T Egress) Queuing Configuration Example on a Catalyst 6500*

```
 ! This section configures the class maps for the queuing policy maps
 ! Note both ingress and egress policy maps share these common class maps
C6500(config-cmap)# class-map type lan-queuing REALTIME-QUEUE
C6500(config-cmap)#  match dscp ef
 ! Realtime traffic (EF) is mapped to the REALTIME-QUEUE
C6500(config-cmap)# class-map type lan-queuing CONTROL-QUEUE
C6500(config-cmap)#  match dscp cs3
 ! Signaling (CS3) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing TRANSACTIONAL-DATA-QUEUE
C6500(config-cmap)#  match  dscp af21  af22  af23
 ! Transactional data (AF2) is mapped to a dedicated nonpriority queue


 ! This section configures the four-class ingress (4Q4T) policy map
C6500(config-pmap)# policy-map type lan-queuing INGRESS-4Q4T
```

```
C6500(config-pmap)#  class REALTIME-QUEUE
C6500(config-pmap-c)#  bandwidth percent 30
 ! Defines the REALTIME-QUEUE with 30% BW
C6500(config-pmap-c)# class CONTROL-QUEUE
C6500(config-pmap-c)#  bandwidth percent 10
 ! Defines the control queue with 10% BW
C6500(config-pmap-c)# class TRANSACTIONAL-DATA-QUEUE
C6500(config-pmap-c)#  bandwidth percent 35
 ! Defines the transactional data queue with 35% BW
C6500(config-pmap-c)#  random-detect dscp-based
C6500(config-pmap-c)#  random-detect dscp af21 percent 80 100
C6500(config-pmap-c)#  random-detect dscp af22 percent 70 100
C6500(config-pmap-c)#  random-detect dscp af23 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF2 PHB
C6500(config-pmap-c)# class class-default
C6500(config-pmap-c)#  random-detect dscp-based
C6500(config-pmap-c)#  random-detect dscp default percent 80 100
 ! DSCP-based WRED is enabled and tuned for default DSCP (0)


 ! This section configures the four-class egress (1P3Q4T) policy map
C6500(config-pmap)# policy-map type lan-queuing EGRESS-1P3Q4T
C6500(config-pmap)#  class REALTIME-QUEUE
C6500(config-pmap-c)#  priority
 ! Enables strict-priority queuing for the real-time queue
C6500(config-pmap-c)# class CONTROL-QUEUE
C6500(config-pmap-c)#  bandwidth remaining percent 10
 ! Defines the control queue with 10% BW remaining
C6500(config-pmap-c)# class TRANSACTIONAL-DATA-QUEUE
C6500(config-pmap-c)#  bandwidth remaining percent 35
 ! Defines the transactional data queue with 35% BW remaining
C6500(config-pmap-c)#  random-detect dscp-based
C6500(config-pmap-c)#  random-detect dscp af21 percent 80 100
C6500(config-pmap-c)#  random-detect dscp af22 percent 70 100
C6500(config-pmap-c)#  random-detect dscp af23 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF2 PHB
C6500(config-pmap-c)# class class-default
C6500(config-pmap-c)#  random-detect dscp-based
C6500(config-pmap-c)#  random-detect dscp default percent 80 100
 ! DSCP-based WRED is enabled and tuned for default DSCP (0)


 ! This section attaches the ingress and egress queuing policies
 ! to the interface(s)
C6500(config-pmap-c)# interface range TenGigabitEthernet 3/1-8
```

```
C6500(config-if-range)# service-policy type lan-queuing input INGRESS-4Q4T
 ! Attaches the INGRESS-4Q4T queuing policy to the interfaces
C6500(config-if-range)# service-policy type lan-queuing output EGRESS-1P3Q4T
 ! Attaches the EGRESS-1P3Q4T queuing policy to the interfaces
```

Note Class maps and policy maps defined for queuing policies need to be explicitly defined as **type lan-queuing**.

Note No explicit bandwidth percentage is needed to be defined on class default.

You can verify the configuration in Example 16-1 with the following commands:

- **show class-map**
- **show policy-map**
- **show queueing interface**

Eight-Class (8Q4T Ingress and 1P7Q4T Egress) Queuing Models

In the eight-class model (illustrated in Figures 11-5 and 11-6), the application class to queue mappings are as follows:

- Real-time traffic (marked EF)) is assigned to the real-time queue, which is allocated 10 percent bandwidth in the ingress 8Q4T model, but is enabled with strict-priority queuing in the egress 1P7Q4T model.
- Network control traffic (marked CS6) is assigned to a dedicated nonpriority queue with a 5 percent bandwidth/bandwidth-remaining allocation.
- Signaling traffic (marked CS3) is assigned to a dedicated nonpriority queue with a 2 percent bandwidth/bandwidth-remaining allocation.
- Interactive video (marked AF4) is assigned to a dedicated nonpriority queue with a 23 percent bandwidth/bandwidth-remaining allocation with DSCP-based WRED enabled.
- Streaming video (marked AF3) is assigned to a dedicated nonpriority queue with a 10 percent bandwidth/bandwidth-remaining allocation with DSCP-based WRED enabled.
- Transactional data (marked AF2) is assigned to dedicated nonpriority queue with a 24 percent bandwidth/bandwidth-remaining allocation with DSCP-based WRED enabled.

- Scavenger traffic (marked CS1) is constrained within a dedicated nonpriority queue with a 1 percent bandwidth/bandwidth-remaining allocation.
- Best effort traffic (marked DF) is assigned to the default queue (which will have at least 25 percent bandwidth implicitly allotted to it) with WRED enabled.

Figure 16-5 illustrates the resulting eight-class (8Q4T ingress and 1P7Q4T egress) queuing models for the Catalyst 6500.

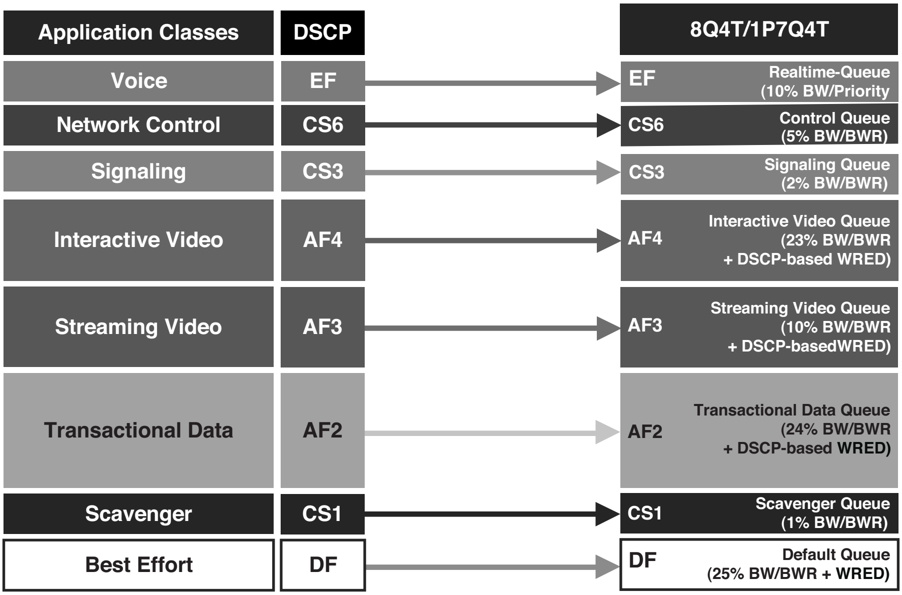

Application Classes
DSCP
8Q4T/1P7Q4T
Voice
EF
EF
Realtime-Queue (10% BW/Priority
Network Control
CS6
CS6
Control Queue (5% BW/BWR)
Signaling
CS3
CS3
Signaling Queue (2% BW/BWR)
Interactive Video
AF4
AF4
Interactive Video Queue (23% BW/BWR + DSCP-based WRED)
Streaming Video
AF3
AF3
Streaming Video Queue (10% BW/BWR + DSCP-basedWRED)
Transactional Data
AF2
AF2
Transactional Data Queue (24% BW/BWR + DSCP-based WRED)
Scavenger
CS1
CS1
Scavenger Queue (1% BW/BWR)
Best Effort
DF
DF
Default Queue (25% BW/BWR + WRED)

Figure 16-5 *Catalyst 6500 Eight-Class (8Q4T Ingress and 1P7Q4T Egress) Queuing Model Examples*

Example 16-2 shows the corresponding configuration for eight-class (8Q4T ingress and 1P7Q4T egress) queuing models for the Catalyst 6500.

Example 16-2 *Eight-Class (8Q4T-Ingress and 1P7Q4T-Egress) Queuing Configuration Example on a Catalyst 6500*

```
 ! This section configures the class maps for the queuing policy maps
 ! Note both ingress and egress policy maps share these common class maps
C6500(config-cmap)# class-map type lan-queuing REALTIME-QUEUE
C6500(config-cmap)#  match dscp ef
 ! Realtime traffic (EF) is mapped to the REALTIME-QUEUE
C6500(config-cmap)# class-map type lan-queuing CONTROL-QUEUE
C6500(config-cmap)#  match dscp cs6
 ! Network Control (CS6) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing SIGNALING-QUEUE
```

```
C6500(config-cmap)#  match dscp cs3
 ! Signaling (CS3) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing INTERACTIVE-VIDEO-QUEUE
C6500(config-cmap)#  match dscp af41 af42 af43
 ! Interactive-Video (AF4) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing STREAMING-VIDEO-QUEUE
C6500(config-cmap)#  match dscp af31 af32 af33
 ! Streaming-Video (AF3) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing TRANSACTIONAL-DATA-QUEUE
C6500(config-cmap)#  match  dscp af21  af22  af23
 ! Transactional Data (AF2) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing SCAVENGER-QUEUE
C6500(config-cmap)#  match dscp cs1
 ! Scavenger (CS1) is mapped to a bandwidth-constrained queue


 ! This section configures the eight-class ingress (8Q4T) policy map
C6500(config-pmap)# policy-map type lan-queuing INGRESS-8Q4T
C6500(config-pmap)#  class REALTIME-QUEUE
C6500(config-pmap-c)#   bandwidth percent 10
 ! Defines the real-time queue with 10% BW
C6500(config-pmap-c)#  class CONTROL-QUEUE
C6500(config-pmap-c)#   bandwidth percent 5
 ! Defines the control queue with 5% BW
C6500(config-pmap-c)#  class SIGNALING-QUEUE
C6500(config-pmap-c)#   bandwidth percent 2
 ! Defines the signaling queue with 2% BW
C6500(config-pmap-c)#  class INTERACTIVE-VIDEO-QUEUE
C6500(config-pmap-c)#   bandwidth percent 23
 ! Defines the interactive video queue with 23% BW
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af41 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af42 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af43 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF4 PHB
C6500(config-pmap-c)#  class STREAMING-VIDEO-QUEUE
C6500(config-pmap-c)#   bandwidth percent 10
 ! Defines the streaming video queue with 10% BW
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af31 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af32 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af33 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF3 PHB
C6500(config-pmap-c)#  class TRANSACTIONAL-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth percent 24
```

```
 ! Defines the transactional data queue with 24% BW
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af21 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af22 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af23 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF2 PHB
C6500(config-pmap-c)#  class SCAVENGER-QUEUE
C6500(config-pmap-c)#   bandwidth percent 1
 ! Defines the scavenger queue with 1% BW
C6500(config-pmap-c)#  class class-default
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp default percent 80 100
 ! DSCP-based WRED is enabled and tuned for default DSCP (0)


 ! This section configures the eight-class egress (1P7Q4T) policy map
C6500(config-pmap)# policy-map type lan-queuing EGRESS-1P7Q4T
C6500(config-pmap)#  class REALTIME-QUEUE
C6500(config-pmap-c)#   priority
 ! Enables strict-priority queuing on the real-time-queue
C6500(config-pmap-c)#  class CONTROL-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 5
 ! Defines the control queue with 5% BW remaining
C6500(config-pmap-c)#  class SIGNALING-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 2
 ! Defines the signaling queue with 2% BW remaining
C6500(config-pmap-c)#  class INTERACTIVE-VIDEO-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 23
 ! Defines the interactive video queue with 23% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af41 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af42 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af43 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF4 PHB
C6500(config-pmap-c)#  class STREAMING-VIDEO-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 10
 ! Defines the streaming video queue with 10% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af31 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af32 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af33 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF3 PHB
C6500(config-pmap-c)#  class TRANSACTIONAL-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 24
 ! Defines the transactional data queue with 24% BW remaining
```

```
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af21 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af22 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af23 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF2 PHB
C6500(config-pmap-c)#  class SCAVENGER-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 1
 ! Defines the scavenger queue with 1% BW remaining
C6500(config-pmap-c)#  class class-default
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp default percent 80 100
 ! DSCP-based WRED is enabled and tuned for default DSCP (0)



 ! This section attaches the ingress and egress queuing policies
 ! to the interface(s)
C6500(config)# interface range TenGigabitEthernet 3/1-8
C6500(config-if-range)# service-policy type lan-queuing input INGRESS-8Q4T
 ! Attaches the INGRESS-8Q4T queuing policy to the interfaces
C6500(config-if-range)# service-policy type lan-queuing output EGRESS-1P7Q4T
 ! Attaches the EGRESS-1P7Q4T queuing policy to the interfaces
```

You can verify the configuration in Example 16-2 with the following commands:

- **show class-map**
- **show policy-map**
- **show queueing interface**

Twelve-Class (8Q4T Ingress and 1P7Q4T Egress) Queuing Models

In the 12-class model (illustrated in Figures 11-7 and 11-8), the application class to queue mappings are as follows:

- Voice (marked EF), Broadcast video (marked CS5), and real-time interactive (marked CS4) are all assigned to the real-time queue, which is allocated 10 percent bandwidth in the ingress 8Q4T model, but is enabled with strict-priority queuing in the egress 1P7Q4T model.
- Network control traffic (marked CS6), signaling traffic (marked CS3), and network management traffic (marked CS2) are all assigned to a dedicated nonpriority queue with a 10 percent bandwidth/bandwidth-remaining allocation; optionally, CS7 traffic may also be mapped to this queue.

- Multimedia conferencing (marked AF4) is assigned to a dedicated nonpriority queue with a 10 percent bandwidth/bandwidth-remaining allocation with DSCP-based WRED enabled.
- Multimedia streaming (marked AF3) is assigned to a dedicated nonpriority queue with a 10 percent bandwidth/bandwidth-remaining allocation with DSCP-based WRED enabled.
- Transactional data (marked AF2) is assigned to dedicated nonpriority queue with a 10 percent bandwidth/bandwidth-remaining allocation with DSCP-based WRED enabled.
- Bulk data (marked AF1) is assigned to a dedicated nonpriority queue with 4 percent bandwidth/bandwidth-remaining allocation with DSCP-based WRED enabled.
- Scavenger traffic (marked CS1) is constrained within a dedicated nonpriority queue with a 1 percent bandwidth allocation.
- Best effort traffic (marked DF) is assigned to the default queue (which will have at least 25 percent bandwidth implicitly allotted to it) with WRED enabled.

Figure 16-6 illustrates the resulting 12-class (8Q4T ingress and 1P7Q4T egress) queuing models for the Catalyst 6500.

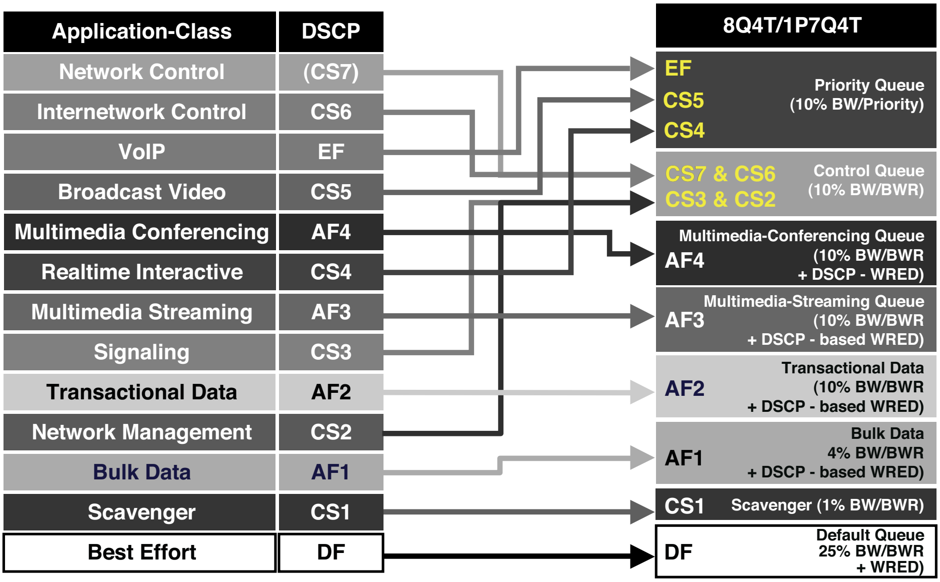


Figure 16-6 *Catalyst 6500 12-Class (8Q4T Ingress and 1P7Q4T Egress) Queuing Model Examples*

Example 16-3 shows the corresponding configuration for 12-class (8Q4T ingress and 1P7Q4T egress) queuing models for the Catalyst 6500.

Example 16-3 *Twelve-Class (8Q4T Ingress and 1P7Q4T Egress) Queuing Configuration Example on a Catalyst 6500*

```
 ! This section configures the class maps for the queuing policy maps
 ! Note both ingress and egress policy maps share these common class maps
C6500(config-cmap)# class-map type lan-queuing REALTIME-QUEUE
C6500(config-cmap)#  match dscp cs4 cs5 ef
 ! Real-time interactive (CS4), broadcast video (CS5) and voice (EF)
 ! are all mapped to the real-time-queue
C6500(config-cmap)# class-map type lan-queuing CONTROL-QUEUE
C6500(config-cmap)#  match dscp cs2 cs3 cs6 cs7
 ! Network management (CS2), signaling (CS3),
 ! Internetwork control/routing (CS6) and network control (CS7)
 ! are all mapped to the control queue
C6500(config-cmap)# class-map type lan-queuing MULTIMEDIA-CONFERENCING-QUEUE
C6500(config-cmap)#  match dscp af41 af42 af43
 ! Multimedia conferencing (AF4) is mapped to a nonpriority queue
C6500(config-cmap)# class-map type lan-queuing MULTIMEDIA-STREAMING-QUEUE
C6500(config-cmap)#  match dscp af31 af32 af33
 ! Multimedia streaming (AF3) is mapped to a nonpriority queue
C6500(config-cmap)# class-map type lan-queuing TRANSACTIONAL-DATA-QUEUE
C6500(config-cmap)#  match  dscp af21  af22  af23
 ! Transactional data (AF2) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing BULK-DATA-QUEUE
C6500(config-cmap)#  match  dscp af11  af12  af13
 ! Bulk Data (AFa) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing SCAVENGER-QUEUE
C6500(config-cmap)#  match dscp cs1
 ! Scavenger (CS1) is mapped to a bandwidth-constrained queue


 ! This section configures the 12-class ingress (8Q4T) policy map
C6500(config-pmap)# policy-map type lan-queuing INGRESS-8Q4T
C6500(config-pmap)#  class REALTIME-QUEUE
C6500(config-pmap-c)#   bandwidth percent 10
 ! Defines the real-time queue with 10% BW
C6500(config-pmap-c)#  class CONTROL-QUEUE
C6500(config-pmap-c)#   bandwidth percent 10
 ! Defines the control queue with 10% BW
C6500(config-pmap-c)#  class MULTIMEDIA-CONFERENCING-QUEUE
C6500(config-pmap-c)#   bandwidth percent 10
 ! Defines the multimedia conferencing queue with 10% BW
```

```
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af41 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af42 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af43 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF4 PHB
C6500(config-pmap-c)#  class MULTIMEDIA-STREAMING-QUEUE
C6500(config-pmap-c)#   bandwidth percent 10
 ! Defines the multimedia streaming queue with 10% BW
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af31 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af32 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af33 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF3 PHB
C6500(config-pmap-c)#  class TRANSACTIONAL-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth percent 10
 ! Defines the transactional data queue with 10% BW
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af21 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af22 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af23 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF2 PHB
C6500(config-pmap-c)#  class BULK-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth percent 4
 ! Defines the bulk data queue with 4% BW
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af11 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af12 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af13 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF1 PHB
C6500(config-pmap-c)#  class SCAVENGER-QUEUE
C6500(config-pmap-c)#   bandwidth percent 1
 ! Defines the scavenger queue with 1% BW
C6500(config-pmap-c)#  class class-default
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp default percent 80 100
 ! DSCP-based WRED is enabled and tuned for default DSCP (0)


 ! This section configures the 12-class egress (1P7Q4T) policy map
C6500(config-pmap)# policy-map type lan-queuing EGRESS-1P7Q4T
C6500(config-pmap)#  class REALTIME-QUEUE
C6500(config-pmap-c)#   priority
 ! Enables strict-priority queuing on the real-time queue
C6500(config-pmap-c)#  class CONTROL-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 10
```

```
 ! Defines the control queue with 10% BW remaining
C6500(config-pmap-c)#  class MULTIMEDIA-CONFERENCING-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 10
 ! Defines the multimedia conferencing queue with 10% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af41 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af42 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af43 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF4 PHB
C6500(config-pmap-c)#  class MULTIMEDIA-STREAMING-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 10
 ! Defines the multimedia streaming queue with 10% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af31 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af32 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af33 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF3 PHB
C6500(config-pmap-c)#  class TRANSACTIONAL-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 10
 ! Defines the transactional data queue with 10% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af21 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af22 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af23 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF2 PHB
C6500(config-pmap-c)#  class BULK-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 4
 ! Defines the bulk data queue with 4% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af11 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af12 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af13 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF1 PHB
C6500(config-pmap-c)#  class SCAVENGER-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 1
 ! Defines the scavenger queue with 1% BW remaining
C6500(config-pmap-c)#  class class-default
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp default percent 80 100



 ! This section attaches the ingress and egress queuing policies
 ! to the interface(s)
C6500(config)# interface range TenGigabitEthernet 3/1-8
C6500(config-if-range)# service-policy type lan-queuing input INGRESS-8Q4T
```

```
 ! Attaches the INGRESS-8Q4T queuing policy to the interfaces
C6500(config-if-range)# service-policy type lan-queuing output EGRESS-1P7Q4T
 ! Attaches the EGRESS-1P7Q4T queuing policy to the interfaces
```

You can verify the configuration in Example 16-3 with the following commands:

- **show class-map**
- **show policy-map**
- **show queueing interface** (as shown in Example 16-4)

Example 16-4 *Verifying Queuing Policies on a Catalyst 6500:* ***show queueing interface***

```
C6500# show queueing interface TenGigabitEthernet 3/8
Interface TenGigabitEthernet3/8 queueing strategy:  Weighted Round-Robin

  Port QoS is enabled globally
  Queueing on Te3/8: Tx Enabled Rx Enabled

Trust boundary disabled

  Trust state: trust DSCP
  Trust state in queueing: trust DSCP
  Extend trust state: not trusted [COS = 0]
  Default COS is 0
    Class-map to Queue in Tx direction
    Class-map           Queue Id
    ----------------------------
    REALTIME-QUEUE            8
    CONTROL-QUEUE             7
    MULTIMEDIA-CONFERENC      6
    MULTIMEDIA-STREAMING      5
    TRANSACTIONAL-DATA-Q      4
    BULK-DATA-QUEUE           3
    SCAVENGER-QUEUE           2
    class-default             1

    Queueing Mode In Tx direction: mode-dscp
    Transmit queues [type = 1p7q4t]:
    Queue Id    Scheduling  Num of thresholds
    -----------------------------------------
       01         WRR                 04
       02         WRR                 04
       03         WRR                 04
```

```
        04          WRR                 04
        05          WRR                 04
        06          WRR                 04
        07          WRR                 04
        08          Priority            01

    WRR bandwidth ratios:   55[queue 1]   1[queue 2]   4[queue 3]  10[queue 4]
10[queue 5]  10[queue 6]  10[queue 7]
    queue-limit ratios:    100[queue 1] 100[queue 2] 100[queue 3] 100[queue 4]
100[queue 5] 100[queue 6] 100[queue 7]  15[Pri Queue]

<snip>

    queue random-detect-min-thresholds
    ----------------------------------
      1    80[1] 70[2] 70[3] 100[4]
      2    40[1] 70[2] 70[3] 70[4]
      3    80[1] 70[2] 60[3] 70[4]
      4    80[1] 70[2] 60[3] 100[4]
      5    80[1] 70[2] 60[3] 100[4]
      6    80[1] 70[2] 60[3] 100[4]
      7    100[1] 100[2] 100[3] 100[4]

    queue random-detect-max-thresholds
    ----------------------------------
      1    100[1] 100[2] 100[3] 100[4]
      2    70[1] 100[2] 100[3] 100[4]
      3    100[1] 100[2] 100[3] 100[4]
      4    100[1] 100[2] 100[3] 100[4]
      5    100[1] 100[2] 100[3] 100[4]
      6    100[1] 100[2] 100[3] 100[4]
      7    100[1] 100[2] 100[3] 100[4]

    WRED disabled queues:      2  7

<snip>


    queue thresh dscp-map
    ---------------------------------------
    1     1      0
    1     2
    1     3
    1     4      1 2 3 4 5 6 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41
42 43 44 45 47 49 50 51 52 53 54 55 57 58 59 60 61 62 63
```

```
2     1      8
2     2
2     3
2     4
3     1      10
3     2      12
3     3      14
3     4
4     1      18
4     2      20
4     3      22
4     4
5     1      26
5     2      28
5     3      30
5     4
6     1      34
6     2      36
6     3      38
6     4
7     1      16 24 48 56
7     2
7     3
7     4
8     1      32 40 46



Class-map to Queue in Rx direction
Class-map           Queue Id
----------------------------
REALTIME-QUEUE            8
CONTROL-QUEUE             7
MULTIMEDIA-CONFERENC      6
MULTIMEDIA-STREAMING      5
TRANSACTIONAL-DATA-Q      4
BULK-DATA-QUEUE           3
SCAVENGER-QUEUE           2
class-default             1

Queueing Mode In Rx direction: mode-dscp
Receive queues [type = 8q4t]:
Queue Id    Scheduling  Num of thresholds
-----------------------------------------
   01          WRR                 04
```

```
       02          WRR                 04
       03          WRR                 04
       04          WRR                 04
       05          WRR                 04
       06          WRR                 04
       07          WRR                 04
       08          WRR                 04

    WRR bandwidth ratios:   45[queue 1]   1[queue 2]   4[queue 3]  10[queue 4]
10[queue 5]  10[queue 6]  10[queue 7]  10[queue 8]
    queue-limit ratios:    100[queue 1] 100[queue 2] 100[queue 3] 100[queue 4]
100[queue 5] 100[queue 6] 100[queue 7] 100[queue 8]

<snip>

    queue random-detect-min-thresholds
    ----------------------------------
      1    80[1] 40[2] 50[3] 100[4]
      2    100[1] 100[2] 100[3] 100[4]
      3    80[1] 70[2] 60[3] 100[4]
      4    80[1] 70[2] 60[3] 100[4]
      5    80[1] 70[2] 60[3] 100[4]
      6    80[1] 70[2] 60[3] 100[4]
      7    100[1] 100[2] 100[3] 100[4]
      8    100[1] 100[2] 100[3] 100[4]

    queue random-detect-max-thresholds
    ----------------------------------
      1    100[1] 80[2] 90[3] 100[4]
      2    100[1] 100[2] 100[3] 100[4]
      3    100[1] 100[2] 100[3] 100[4]
      4    100[1] 100[2] 100[3] 100[4]
      5    100[1] 100[2] 100[3] 100[4]
      6    100[1] 100[2] 100[3] 100[4]
      7    100[1] 100[2] 100[3] 100[4]
      8    100[1] 100[2] 100[3] 100[4]

    WRED disabled queues:      2  7  8

<snip>

    queue thresh dscp-map
    ---------------------------------------
    1     1      0
    1     2
    1     3
    1     4      1 2 3 4 5 6 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 42
 43 44 45 47 49 50 51 52 53 54 55 57 58 59 60 61 62 63
    2     1      8
    2     2
    2     3
    2     4
    3     1      10
    3     2      12
    3     3      14
    3     4
    4     1      18
    4     2      20
    4     3      22
    4     4
    5     1      26
    5     2      28
    5     3      30
    5     4
    6     1      34
    6     2      36
    6     3      38
    6     4
    7     1      16 24 48 56
    7     2
    7     3
    7     4
    8     1      32 40 46
    8     2
    8     3
    8     4


  Packets dropped on Transmit:
    BPDU packets:  0

    queue              dropped  [dscp-map]
    ---------------------------------------------

    1                        0  [0 1 2 3 4 5 6 7 9 11 13 15 17 19 21 23 25 27 29
 31 33 35 37 39 41 42 43 44 45 47 49 50 51 52 53 54 55 57 58 59 60 61 62 63 ]
    2                   213432   [8 ]
    3                        0  [10 12 14 ]
    4                        0  [18 20 22 ]
    5                        0  [26 28 30 ]
    6                        0   [34 36 38 ]
    7                        0   [16 24 48 56 ]
    8                        0   [32 40 46 ]

  Packets dropped on Receive:
    BPDU packets:  0

    queue              dropped   [dscp-map]
    ---------------------------------------------
    1                        0   [0 1 2 3 4 5 6 7 9 11 13 15 17 19 21 23 25 27 29
31 33 35 37 39 41 42 43 44 45 47 49 50 51 52 53 54 55 57 58 59 60 61 62 63 ]
    2                        0   [8 ]
    3                        0   [10 12 14 ]
    4                        0   [18 20 22 ]
    5                        0   [26 28 30 ]
    6                        0   [34 36 38 ]
    7                        0   [16 24 48 56 ]
    8                        0   [32 40 46 ]
C6500#
```

Example 16-4 shows all the static configuration details of the 12-class 8Q4T ingress and 1P7Q4T egress queuing policies (all nondefault configuration settings have been highlighted) and dynamic per-queue dropping details, including 213,432 queuing drops noted on the (transmitting) scavenger queue.

Note how even though the policies were configured via C3PL, they have been translated into hardware weighted round-robin (WRR) queuing parameters. (For example, queuing class names have been assigned discrete hardware-queue numbers, and DSCP-based WRED thresholds have been assigned to specific queue/threshold combinations to implement their functionality.)

2P6Q4T Ingress and Egress Queuing Models

As shown in Table 16-2, the Catalyst 6904 40 Gigabit Ethernet module may be configured with either a 1P7Q4T or with a dual-PQ design of 2P6Q4T.

The priority queues have a hierarchy, as specified by their priority levels:

- **Priority level 1:** Will be fully serviced before servicing any other queue.
- **Priority level 2:** Will only be serviced after the priority level 1 queue is fully serviced, and will have its service interrupted if traffic arrives in the priority level 1 queue.

Only after both priority queues have been fully serviced will the nonpriority queues be serviced (in a WRR fashion).

The preceding queuing models may be amended fairly easily to take advantage of this feature, as shown in Example 16-5. The same policy can be applied in both the ingress and egress directions on these linecards.

Example 16-5 *2P6Q4T Ingress and Egress Queuing Policy Amendment Configuration Example on a Catalyst 6500*

```
! This section configures a dual-PQ (2P6Q4T) ingress/egress policy map
C6500(config-pmap)# policy-map type lan-queuing 2P6Q4T
C6500(config-pmap)#  class REALTIME-VOICE-QUEUE
C6500(config-pmap-c)#   priority level 1
 ! Enables strict-priority queuing on the real-time voice queue
C6500(config-pmap)#  class REALTIME-VIDEO-QUEUE
C6500(config-pmap-c)#   priority level 2
 ! Enables level-2 priority queuing on the real-time video queue
...

! This section attaches the ingress and egress queuing policies
 ! to the interface(s)
C6500(config)# interface FortyGigabitEthernet 1/1
C6500(config-if)# service-policy type lan-queuing input 2P6Q4T
 ! Attaches the 2P6Q4T queuing policy in the ingress direction
C6500(config-if)# service-policy type lan-queuing output 2P6Q4T
 ! Attaches the 2P6Q4T queuing policy in the egress direction
```

You can verify the configuration in Example 16-5 with the following commands:

- **show class-map**
- **show policy-map**
- **show queueing interface**

Additional Platform-Specific QoS Design Options

These designs represent a generic building block for Catalyst 6500 QoS in a campus core switch role, but they are by no means the only design options available to you. Additional options and considerations include the following:

- Access-edge design options
- Microflow policing
- Per-VLAN QoS design
- EtherChannel QoS design

- AutoQoS SRND4
- Control plane policing

Each of these additional QoS design options is discussed in turn.

Access-Edge Design Options

This chapter has focused on QoS designs for the Catalyst 6500 in the role of a campus core switch (which are generally equivalent to the QoS designs required were it serving in the role of a campus distribution switch). However, the Catalyst 6500 may also be deployed as a campus access switch. Therefore, a few additional design options would apply in such a role, including the following access-edge models:

- Conditional Trust Model
- Classification and Marking Model
- Classification, Marking, and Policing Model

Each of these access-edge design options is discussed in turn.

Conditional Trust Model

As previously mentioned, the C3PL-based Catalyst 6500 trusts at Layer 2 and Layer 3 by default and therefore does not require any explicit commands to perform such functions.

However, there is a need to provide conditional trust functionality for all switching platforms when they are deployed as an access switch. Hence, there is a corresponding command for conditional trust on the Catalyst 6500, namely **platform qos trust device**.

Note C3PL includes some platform-specific syntax (such as the conditional **trust** command), which is very similar to MLS QoS commands. However, C3PL replaces the **mls** keyword with **platform**.

Note At the time of this writing, the only device that the Catalyst 6500 supports conditional trust for is the Cisco IP phone via the **cisco-phone** keyword option.

It is important to remember that Cisco IP phones can only re-mark CoS bits (on PC-generated traffic). Therefore, the Conditional Trust Model on the Catalyst 6500 requires a dynamic conditional trust policy applied to the port in conjunction with a simple C3PL policy that explicitly matches CoS 5 (for voice) and CoS 3 (for signaling) and marks the DSCP values of these packets to EF and CS3, respectively (essentially performing a CoS-to-DSCP mapping). Example 16-6 shows this conditional trust model for the Catalyst 6500.

Example 16-6 *Configuring (CoS-Based) Conditional Trust to a Cisco IP Phone on a Catalyst 6500*

```
 ! This section defines the class maps to match voice and signaling
C6500(config-cmap)# class-map match-all VOICE
C6500(config-cmap)#  match cos 5
 ! Matches VoIP traffic on CoS 5
C6500(config-cmap)# class-map match-all SIGNALING
C6500(config-cmap)#  match cos 3
 ! Matches signaling traffic on CoS 3


 ! This section defines the CoS-to-DSCP re-marking policy map
C6500(config-cmap)# policy-map CISCO-IPPHONE
C6500(config-pmap)#  class VOICE
C6500(config-pmap-c)# set dscp ef
 ! Maps CoS 5 to DSCP EF
C6500(config-pmap-c)# class SIGNALING
C6500(config-pmap-c)#  set dscp cs3
 ! Maps CoS 3 to DSCP CS3
C6500(config-pmap-c)# class class-default
C6500(config-pmap-c)#  set dscp default
 ! All other traffic is set to DSCP DF


 ! This section applies conditional trust and policy map to the int(s)
C6500(config)# interface range GigabitEthernet 1/1-48
C6500(config-if-range)# switchport
C6500(config-if-range)# switchport access vlan 10
C6500(config-if-range)# switchport voice vlan 110
C6500(config-if-range)# spanning-tree portfast edge
C6500(config-if-range)# platform qos trust device cisco-phone
 ! Applies conditional-trust/trust-boundary to the switch port
C6500(config-if-range)# service-policy input CISCO-IPPHONE
 ! Attaches the CoS-to-DSCP mapping policy map
```

You can verify the configuration in Example 16-6 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**
- **show queueing interface** (as shown in Example 16-7)

Example 16-7 *Verifying Conditional-Trust Policies on a Catalyst 6500:* ***show queueing interface***

```
C6500# show queueing interface GigabitEthernet 1/48
Interface GigabitEthernet1/40 queueing strategy:  Weighted Round-Robin

  Port QoS is enabled globally
  Queueing on Gi1/40: Tx Enabled Rx Enabled

Trust boundary enabled

  Port is trusted
  Trust state in queueing: trust COS
  Extend trust state: trust CoS

<snip>
```

Example 16-7 shows that the conditional trust statement (which enables a conditional trust boundary) has been applied to the switch port and, with a Cisco IP phone attached to it, the port trust state is set to trust CoS (because this is the default extended trust state for a switch port operating at Layer 2).

Classification and Marking Models

In many scenarios, trust models may not be available or sufficient to distinctly classify all types of traffic required by the end-to-end QoS strategic model. Therefore, explicit classification and marking policies may be needed at the access edge.

Example 16-8 shows a configuration example based on Figure 11-5 (an eight-class QoS model).

Note As previously discussed, not all application classes may be present at the access edge on ingress. For example, streaming video would likely not be present at the access edge on ingress (because these flows are not *sourced* from campus endpoints, but are likely *destined* to them), nor would network control flows be sourced from campus endpoints. Therefore, these classes would not need to be included in the access-edge classification and marking policy map.

Note Referenced access lists are omitted from the policy examples for brevity.

Example 16-8 *Classification and Marking Policy Example on a Catalyst 6500*

```
 ! This section configures the class maps
C6500(config-cmap)# class-map match-all VOICE
C6500(config-cmap)#  match dscp ef
 ! Voice is matched on DSCP EF
C6500(config-cmap)# class-map match-all INTERACTIVE-VIDEO
C6500(config-cmap)#  match access-group name INTERACTIVE-VIDEO
 ! Associates interactive-video access list with class map
C6500(config-cmap)# class-map match-all SIGNALING
C6500(config-cmap)#  match dscp cs3
 ! Signaling is matched on DSCP CS3
C6500(config-cmap)# class-map match-all TRANSACTIONAL-DATA
C6500(config-cmap)#  match access-group name TRANSACTIONAL-DATA
 ! Associates transactional data access list with class map
C6500(config-cmap)# class-map match-all SCAVENGER
C6500(config-cmap)#  match access-group name SCAVENGER
 ! Associates scavenger access list with class map


 ! This section configures the per-port ingress marking policy map
C6500(config-cmap)# policy-map PER-PORT-MARKING
C6500(config-pmap)#  class VOICE
C6500(config-pmap-c)# set dscp ef
 ! VoIP is marked EF
C6500(config-pmap-c)# class INTERACTIVE-VIDEO
C6500(config-pmap-c)#  set dscp af41
 ! Interactive-Video is marked AF41
C6500(config-pmap-c)# class SIGNALING
C6500(config-pmap-c)#  set dscp cs3
 ! Signaling is marked CS3
C6500(config-pmap-c)# class TRANSACTIONAL-DATA
C6500(config-pmap-c)#  set dscp af21
 ! Transactional data is marked AF21
C6500(config-pmap-c)# class SCAVENGER
C6500(config-pmap-c)#  set dscp cs1
 ! Scavenger traffic is marked CS1
C6500(config-pmap-c)# class class-default
C6500(config-pmap-c)#  set dscp default
 ! All other traffic is marked DF


 ! This section attaches the service policy to the interface(s)
C6500(config)# interface range GigabitEthernet 1/1-48
C6500(config-if-range)# switchport
```

```
C6500(config-if-range)# switchport access vlan 10
C6500(config-if-range)# switchport voice vlan 110
C6500(config-if-range)# spanning-tree portfast edge
C6500(config-if-range)# platform qos trust device cisco-phone
 ! The interface is set to conditionally trust Cisco IP phones
C6500(config-if-range)# service-policy input PER-PORT-MARKING
 ! Attaches the per-port marking policy to the interface(s)
```

You can verify the configuration in Example 16-8 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface** (as shown in Example 16-9)
- **show queueing interface**

Example 16-9 *Verifying Service Policies on a Catalyst 6500:* ***show policy-map interface***

```
C6500# show policy-map interface GigabitEthernet 1/48

 GigabitEthernet1/48

  Service-policy input: PER-PORT-MARKING

    class-map: VOICE (match-all)
      Match: ip dscp ef (46)
      set dscp 46:
      Earl in slot 5 :
      0 bytes
      30 second offered rate 0 bps
      aggregate-forwarded 0 bytes

    class-map: INTERACTIVE-VIDEO (match-all)
      Match: access-group name INTERACTIVE-VIDEO
      set dscp 34:

    class-map: SIGNALING (match-all)
      Match: ip dscp cs3 (24)
      set dscp 24:
      Earl in slot 5 :
      0 bytes
      30 second offered rate 0 bps
      aggregate-forwarded 0 bytes
    class-map: TRANSACTIONAL-DATA (match-all)
      Match: access-group name TRANSACTIONAL-DATA
      set dscp 18:

    class-map: SCAVENGER (match-all)
      Match: access-group name SCAVENGER
      set dscp 8:

    class-map: class-default (match-any)
      Match: any
      set dscp 0:
      Earl in slot 5 :
      3081 bytes
      30 second offered rate 720 bps
      aggregate-forwarded 3081 bytes
C6500#
```

Classification, Marking, and Policing Model

In addition to classification and marking, policing may also be required at the access edge. The Catalyst 6500 can perform single-rate (two-color) policing and three-color policing—via either the RFC 2697 single-rate three-color marker (srTCM) or the RFC 2698 two-rate three-color marker (trTCM). Example 16-10 shows a per-port single-rate policing example for the Catalyst 6500 (based on Figure 12-8), and Example 16-11 shows policy amendments to support an RFC 2698 two-rate three-color marker.

Example 16-10 *(Single-Rate Two-Color) Per-Port Policing Configuration Example on a Catalyst 6500*

```
 ! This section configures the discard-class/dscp markdown map
 ! for traffic exceeding normal burst
C6500(config)# table-map policed-discard-class-normal-burst-map
C6500(config-tablemap)#  map from 18 to 20
 ! AF21 (18) is mapped to AF22 (20) if exceeding normal burst
C6500(config-tablemap)#  map from 10 to 12
 ! AF11 (10) is mapped to AF12 (12) if exceeding normal burst
C6500(config-tablemap)#  map from 0 to 8
 ! DSCP 0 is mapped to CS1 (Scavenger) if exceeding normal burst



 ! This section configures the class maps
C6500(config-cmap)# class-map VVLAN-VOIP
C6500(config-cmap)#  match dscp ef
 ! Voice from the VVLAN is matched on DSCP EF
```

```
C6500(config-cmap)# class-map VVLAN-SIGNALING
C6500(config-cmap)#  match dscp cs3
 ! Signaling from the VVLAN is matched on DSCP CS3
C6500(config-cmap)# class-map MULTIMEDIA-CONFERENCING
C6500(config-cmap)#  match access-group name MULTIMEDIA-CONFERENCING
 ! Multimedia conferencing traffic is matched by an ACL
C6500(config-cmap)# class-map SIGNALING
C6500(config-cmap)#  match access-group name SIGNALING
 ! Signaling from the DVLAN is matched by an ACL
C6500(config-cmap)# class-map TRANSACTIONAL-DATA
C6500(config-cmap)#  match access-group name TRANSACTIONAL-DATA
 ! Transactional data traffic is matched by an ACL
C6500(config-cmap)# class-map BULK-DATA
C6500(config-cmap)#  match access-group name BULK-DATA
 ! Bulk data traffic is matched by an ACL
C6500(config-cmap)# class-map SCAVENGER
C6500(config-cmap)#  match access-group name SCAVENGER
 ! Scavenger traffic is matched by an ACL


 ! This section configures the single-rate per-port policer policy map
C6500(config-pmap)# policy-map PER-PORT-POLICING
C6500(config-pmap-c)# class VVLAN-VOIP
C6500(config-pmap-c)#  police 128k
C6500(config-pmap-c-police)# conform-action transmit
C6500(config-pmap-c-police)# exceed-action drop
 ! Voice from the VVLAN is policed to drop at 128 kbps
C6500(config-pmap-c)# class VVLAN-SIGNALING
C6500(config-pmap-c)#  police 32k
C6500(config-pmap-c-police)# conform-action transmit
C6500(config-pmap-c-police)# exceed-action drop
 ! Signaling from the VVLAN is policed to drop at 32 kbps
C6500(config-pmap-c)# class MULTIMEDIA-CONFERENCING
C6500(config-pmap-c)#  police 5m
C6500(config-pmap-c-police)# conform-action set-dscp-transmit af41
C6500(config-pmap-c-police)# exceed-action drop
 ! Multimedia conferencing is policed to drop at 5 Mbps
C6500(config-pmap-c)# class SIGNALING
C6500(config-pmap-c)#  police 32k
C6500(config-pmap-c-police)# conform-action set-dscp-transmit cs3
C6500(config-pmap-c-police)# exceed-action drop
 ! Signaling from the DVLAN is policed to drop at 32 Kbps
C6500(config-pmap-c)# class TRANSACTIONAL-DATA
C6500(config-pmap-c)#  police 10m
C6500(config-pmap-c-police)# conform-action set-dscp-transmit af21
```

```
C6500(config-pmap-c-police)# exceed-action policed-dscp-transmit
 ! Transactional data is policed to re-mark from AF21 to AF22 at 10 Mbps
C6500(config-pmap-c)# class BULK-DATA
C6500(config-pmap-c)#  police 10m
C6500(config-pmap-c-police)# conform-action set-dscp-transmit af11
C6500(config-pmap-c-police)# exceed-action policed-dscp-transmit
 ! Bulk data is policed to re-mark from AF11 to AF12 at 10 Mbps
C6500(config-pmap-c)# class SCAVENGER
C6500(config-pmap-c)#  police 10m
C6500(config-pmap-c-police)# conform-action set-dscp-transmit cs1
C6500(config-pmap-c-police)# exceed-action drop
 ! Scavenger is policed to drop at 10 Mbps
C6500(config-pmap-c)# class class-default
C6500(config-pmap-c)#  police 10m
C6500(config-pmap-c-police)# conform-action set-dscp-transmit default
C6500(config-pmap-c-police)# exceed-action policed-dscp-transmit
 ! Best effort is policed to re-mark to Scavenger (CS1) at 10 Mbps


 ! This section applies the service-policy to the interface(s)
C6500(config)# interface range GigabitEthernet 1/1-48
C6500(config-if-range)# switchport
C6500(config-if-range)# switchport access vlan 10
C6500(config-if-range)# switchport voice vlan 110
C6500(config-if-range)# spanning-tree portfast
C6500(config-if-range)# platform qos trust device cisco-phone
 ! The interface is set to conditionally trust Cisco IP phones
C6500(config-if-range)# service-policy input PER-PORT-POLICING
 ! Attaches the Per-Port Policing policy to the interface(s)
```

Note DSCP values to be re-marked cannot be set directly on the Catalyst 6500, but must be set via markdown table maps, as shown in the preceding example.

Note The Catalyst 6500 IOS Software allows for policing rates to be entered using the postfixes **k** (for kilobits), **m** (for megabits), and **g** (for gigabits), as shown in Example 16-8. In addition, decimal points are allowed in conjunction with these postfixes. For example, a rate of 10.5 Mbps could be entered with the policy map command **police 10.5m**. Although these policing rates are converted to their full bits per second values within the configuration, it makes the entering of these rate more user friendly and less error prone (as could easily be the case when having to enter up to 10 zeros to define the policing rate).

You can verify the configuration in Example 16-10 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**
- **show queueing interface**
- **show table-map policed-discard-class-normal-burst-map** (as shown in Example 16-12)

To avoid excessive repetition, Example 16-11 amends and expands the policer from a single-rate two-color marker to a two-rate three-color marker only on a single-class (the Bulk Data class). However, similar amendments can be made on any AF class of traffic. As before, markdown table maps are required to be configured, not only for the normal policing rate but also for the peak/maximum policing rate.

Example 16-11 *(Two-Rate Three-Color) Per-Port Policing Configuration-Amendment Example on a Catalyst 6500*

```
 ! This section configures the discard-class/dscp markdown map
 ! for traffic exceeding normal burst
C6500(config)# table-map policed-discard-class-normal-burst-map
C6500(config-tablemap)# map from 18 to 20
 ! AF21 (18) is mapped to AF22 (20) if exceeding normal burst
C6500(config-tablemap)# map from 10 to 12
 ! AF11 (10) is mapped to AF12 (12) if exceeding normal burst
C6500(config-tablemap)# map from 0 to 8
 ! DSCP 0 is mapped to CS1 (Scavenger) if exceeding normal burst


 ! This section configures the discard-class/dscp markdown map
 ! for traffic violating maximum burst
C6500(config)# table-map policed-discard-class-max-burst-map
C6500(config-tablemap)# map from 18 to 22
 ! AF21 (18) is mapped to AF23 (22) if violating max/peak burst
C6500(config-tablemap)# map from 10 to 14
 ! AF21 (10) is mapped to AF13 (14) if violating max/peak burst


 ! This section configures a dual-rate per-port policing policy map
C6500(config)# policy-map TWO-RATE-POLICER

<snip>

C6500(config-pmap)# class BULK-DATA
```

```
C6500(config-pmap-c)# set dscp af11
C6500(config-pmap-c)# police 10m pir 15m
  ! Bulk data is policed to 10 Mbps rate and 15 Mbps peak rate
C6500(config-pmap-c-police)# conform-action set-dscp-transmit af11
 ! Bulk data under 10 Mbps will be marked AF11
C6500(config-pmap-c-police)# exceed-action policed-dscp-transmit
 ! Bulk data traffic between 10 Mbps and 15 Mbps will be marked AF12
 ! per policed-discard-class-normal-burst-map table map
C6500(config-pmap-c-police)# violate-action policed-dscp-transmit
 ! Bulk data traffic over 15 Mbps will be marked AF13
 ! per policed-discard-class-max-burst-map table map
```

You can verify the configuration in Example 16-11 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**
- **show table-map policed-discard-class-normal-burst-map** (as shown in Example 16-12)
- **show table-map policed-discard-class-max-burst-map** (as shown in Example 16-13)

Example 16-12 *Verifying Table Maps on a Catalyst 6500: **show table-map***

```
C6500# show table-map
<snip>

   policed-discard-class-normal-burst-map:
(discard-class= d1d2)
     d1 :  d2 0  1  2  3  4  5  6  7  8  9
     ------------------------------------
      0 :     08 01 02 03 04 05 06 07 08 09
      1 :     12 11 12 13 14 15 16 17 20 19
      2 :     20 21 22 23 24 25 26 27 28 29
      3 :     30 31 32 33 34 35 36 37 38 39
      4 :     40 41 42 43 44 45 46 47 48 49
      5 :     50 51 52 53 54 55 56 57 58 59
      6 :     60 61 62 63
```

Example 16-13 *Verifying Table-Maps on a Catalyst 6500:* ***show table-map***

```
C6500# show table-map
<snip>
   policed-discard-class-max-burst-map:
(discard-class= d1d2)
     d1 :  d2 0  1  2  3  4  5  6  7  8  9
     -------------------------------------
      0 :     00 01 02 03 04 05 06 07 08 09
      1 :     14 11 12 13 14 15 16 17 22 19
      2 :     20 21 22 23 24 25 26 27 28 29
      3 :     30 31 32 33 34 35 36 37 38 39
      4 :     40 41 42 43 44 45 46 47 48 49
      5 :     50 51 52 53 54 55 56 57 58 59
      6 :     60 61 62 63

C6500#
```

In Examples 16-12 and 16-13, the policing discard class markdown mappings are shown in two tables:

- The first table (the policed-discard-class-normal-burst-map) defines the re-marking action for packets exceeding the committed information rate (CIR).
- The second table (the policed-discard-class-max-burst-map) defines the re-marking action for packets exceeding the peak information rate (PIR).

The first digit of the DSCP value of a packet offered to a policer is shown along the Y-axis of the table; the second digit of the DSCP value of a packet offered to a policer is shown along the X-axis of the table.

For example, the DSCP value for the Bulk Data application class (AF11 / DSCP 10) is found in both tables in the row d1=1 and column d2=0. And, as shown, packets with this offered DSCP value are re-marked to 12 (AF12) if found to be in excess of the normal policing rate or will be re-marked to 14 (AF13) if found to be in violation of the peak/ maximum policing rate.

Also highlighted are DSCP 0, which is re-marked as scavenger (CS1 / DSCP 8) if found to be exceeding the normal policing rate, and DSCP 18 (AF21), which will be re-marked to DSCP 20 (AF22) if exceeding the normal policing rate, but re-marked to DSCP 22 (AF23) if violating the peak policing rate.

Microflow Policing

The Catalyst 6500 also supports microflow policing, such that individual/discrete flows can be metered and limited, not just aggregates. Microflow policers are enabled with the **police flow** policy map command.

Microflow policers are very useful and flexible. For example, you can configure a microflow policer to use only source addresses, which applies the microflow policer to all traffic from a source address regardless of the destination addresses. This is an effective way to enforce user-based rate limiting, such as in a university campus environment where an administrator might want to rate limit individual student traffic.

Conversely, you can configure a microflow policer to use only destination addresses, which applies the microflow policer to all traffic to a destination address regardless of the source addresses.

Also, you can include both an aggregate policer and a microflow policer in each policy map class to police a flow based on both its own bandwidth utilization and on its bandwidth utilization combined with that of other flows. For example, you could create a microflow policer with a bandwidth limit suitable for individuals in a group, and you could create an aggregate policer with bandwidth limits suitable for the group as a whole.

Note If traffic is both aggregate and microflow policed, the aggregate and microflow policers must both be in the same policy map class, and each must use the same **conform-action** and **exceed-action** keyword option: **drop**, **set-dscp-transmit**, or **transmit**.

Example 16-14 shows how to configure a flow-based QoS policy that uses microflow policing in the context of user-based rate limiting: any and all flows sourced from a given unique IP address are microflow policed to 1 Mbps.

Example 16-14 *Configuring Microflow Policing (in a User-Based Rate-Limiting Context) on a Catalyst 6500*

```
 ! This section defines the microflow policer policy map
C6500(config)# policy-map 1MBPS-MICROFLOW-POLICER
C6500(config-pmap)#  class class-default
C6500(config-pmap-c)# police flow mask src-only 1000000 conform-action transmit
exceed-action drop
 ! Configures a user-based rate-limiter that meters each discrete
 ! microflow from a given source and limits these to 1 Mbps


 ! This section applies the microflow policer to the interface
C6500(config)# interface GigabitEthernet1/1
C6500(config-if)# service-policy input 1MBPS-MICROFLOW-POLICER
 ! Applies the 1-Mbps microflow policer to the interface
```

You can verify the configuration in Example 16-14 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Per-VLAN QoS Design

The Catalyst 6500 supports VLAN-based QoS via the **platform qos vlan-based** interface command, as shown in Example 16-15.

Example 16-15 *Per-VLAN Marking Configuration Example on a Catalyst 6500*

```
 ! This section configures the interface(s) for conditional trust,
C6500(config)# interface range GigabitEthernet 1/1-48
C6500(config-if-range)# switchport
C6500(config-if-range)# switchport access vlan 10
C6500(config-if-range)# switchport voice vlan 110
C6500(config-if-range)# spanning-tree portfast edge
C6500(config-if-range)# platform qos vlan-based
 ! Enables VLAN-based QoS on the interface(s)


 ! This section attaches the DVLAN policy to the DVLAN interface
C6500(config)# interface Vlan 10
C6500(config-if)# description DVLAN
C6500(config-if)# service-policy input DVLAN-MARKING
 ! Attaches the DVLAN Per-VLAN Marking policy to the DVLAN interface


 ! This section attaches the VVLAN policy to the VVLAN interface
C6500(config)# interface Vlan 110
C6500(config-if)# description VVLAN
C6500(config-if)# service-policy input VVLAN-MARKING
 ! Attaches the VVLAN Per-VLAN Marking policy to the VVLAN interface
```

You can verify the configuration in Example 16-15 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface vlan** *vlan-number*

Note As previously discussed, it is not recommended to deploy (aggregate) policing policies on a per-VLAN basis, because the number of endpoints in a given VLAN can vary, which can easily result in significant fluctuations of per-endpoint policing limits.

EtherChannel QoS Design

EtherChannel interfaces configured on Catalyst 6500 switches require *ingress and egress (nonqueuing) QoS policies* (including classification, marking, and policing policies) to be attached to the *logical port channel interface* via **service-policy** statements. (Trust statements are not required because this C3PL-based platform trusts by default.) However, both *ingress and egress queuing policies* are attached to the *physical port member interfaces* via **service-policy** statements.

- Classification, marking, and policing policies are applied to the logical port channel interfaces.
- Queuing policies (both ingress and egress) are applied to the physical port member interfaces.

Example 16-16 demonstrates Catalyst 6500 EtherChannel QoS design.

Example 16-16 *EtherChannel QoS Design on a Catalyst 6500*

```
 ! This section configures the logical port channel interface
C6500(config)# interface Port-channel1
C6500(config-if)# description ETHERCHANNEL-LOGICAL-INTERFACE
C6500(config-if)# service-policy input MARKING
 ! Attaches a marking policy to the logical port-channel interface
C6500(config-if)# platform qos channel-consistency
 ! [Enabled by default] checks if physical member interfaces
 ! have the same ingress and egress queuing policies


 ! This section configures queuing on physical port-member interfaces
C6500(config)# interface range TenGigabitEthernet1/1-2
C6500(config-if-range)# description PORT-CHANNEL1-PORT-MEMBER
C6500(config-if-range)# service-policy type lan-queuing input INGRESS-8Q4T
 ! Attaches the INGRESS-8Q4T queuing policy to the interfaces
C6500(config-if-range)# service-policy type lan-queuing output EGRESS-1P7Q4T
 ! Attaches the EGRESS-1P7Q4T queuing policy to the interfaces
```

You can verify the configuration in Example 16-16 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

AutoQoS SRND4

AutoQoS SRND4 is not supported on the Catalyst 6500 at the time of this writing. (However, AutoQoS for IP telephony is available on this platform.)

Control Plane Policing

Control plane policing is supported on the Catalyst 6500 and is detailed in Appendix B, "Control Plane Policing."

Summary

This design chapter primarily discussed the best-practice QoS design recommendations for the Cisco Catalyst 6500 (Supervisor 2T) series switch in the role of a campus core layer switch (which incidentally are generally equivalent to the QoS designs required were it serving in the role of a campus distribution switch).

Because the Catalyst 6500 is a C3PL-based QoS platform, QoS is enabled by default, as is DSCP trust on all ports. Therefore, there are effectively only two required steps to configure QoS on a Catalyst 6500 performing the role of a core switch: configure an ingress queuing policy, and configure an egress queuing policy.

To this end, 4-class, 8-class, and 12-class (ingress and egress) queuing policies were detailed, along with corresponding configurations and verification examples, leveraging the Catalyst 6500's flexible 8Q8T ingress and 1P7Q4T egress hardware queuing capabilities.

Additional platform-specific design options and considerations were discussed, including how the Catalyst 6500 could be deployed as an access-edge switch, and how to configure microflow policing, per-VLAN QoS, and EtherChannel QoS designs.

AutoQoS SRND4 is not supported on the Catalyst 6500 (at the time of this writing). However, control plane policing is supported on this platform and is covered in Appendix B.

Further Reading

Cisco Enterprise Medianet Campus QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html

Medianet Campus QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampusaag.html

Medianet Catalyst 6500 QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampuscat6500aag.html

Catalyst 6500 Sup2T System QOS Architecture: http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_paper_c11-652042.html

Cisco Catalyst 6500 Series Switch IOS Release 15.1SY Supervisor Engine 2T Software Configuration Guide: QoS Configuration Guide: http://www.cisco.com/en/US/partner/docs/switches/lan/catalyst6500/ios/15.1SY/config_guide/sup2T/qos_overview.html

This page intentionally left blank

Chapter 17

Campus QoS Design Case Study

Having defined Tifosi Software's strategic eight-class end-to-end QoS model (in Chapter 12, "Strategic QoS Design Case Study," and illustrated in Figure 12-2), the networking team is now ready to apply these policies on a per-platform basis to their (wired) campus network infrastructure.

Tifosi has a multitier campus network, consisting of Cisco Catalyst 3750-X switches at the access layer, Cisco Catalyst 4500-E Supervisor 7-E switches at the distribution layer, and Cisco Catalyst 6500-Supervisor 2T switches at the core layer (with both WS-X6908-10GE and WS-X6904-40G-2TXL modules).

Tifosi has multiple types of endpoints connecting to the access layer of the campus, including the following:

- Desktop PCs and Macs, and UNIX/Linux workstations (abbreviated simply as PCs)
- Cisco IP phones
- Cisco wireless access points
- Cisco TelePresence Systems
- Printers

The networking team wants to ensure that all endpoints have corresponding trust or explicit classification and marking policies applied to them.

In addition, Tifosi has both wired and wireless guest VLANs (for customers, partners, vendors, and so on) to use while on site. However, they have noticed that often a disproportionate amount of traffic is being used on the endpoints connecting to the guest VLAN and, therefore, they want to limit these flows to 1 Mbps each.

Finally, the networking team wants to ensure that eight-class egress queuing policies are applied on all ports (and eight-class ingress queuing on all platforms/linecards that support this feature).

Figure 17-1 shows the QoS policies to be applied to Tifosi's campus network.

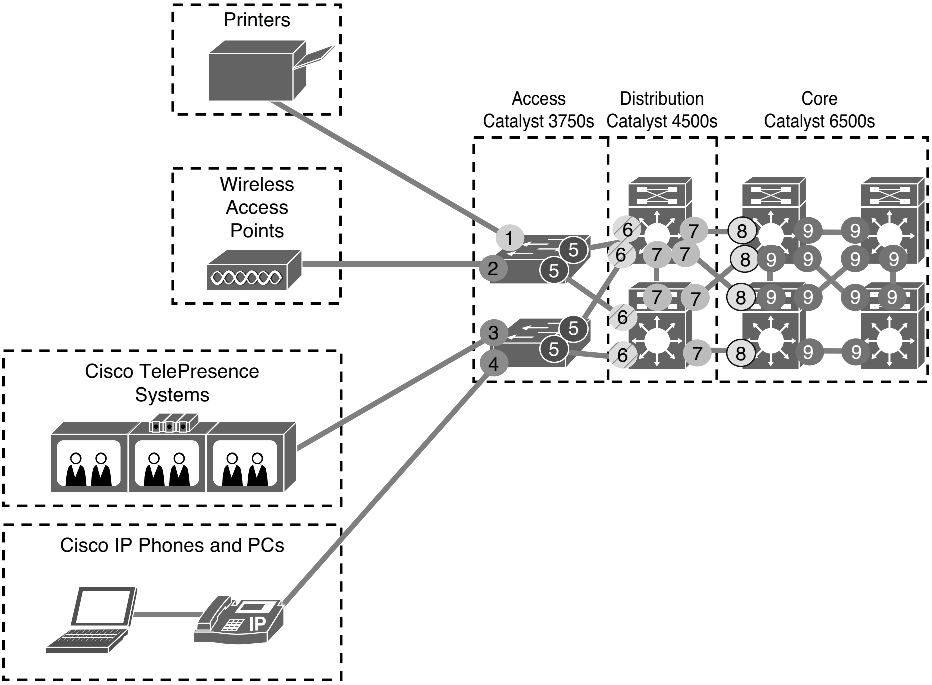


Figure 17-1 *Tifosi Software Case Study: Campus Network QoS Design*

These campus QoS policies are detailed as follows:

- **Policy 1:** Untrusted endpoints (Catalyst 3750-X)
  - Disable trust.
  - Enable 1P1Q3T ingress queuing for an eight-class QoS model.
  - Enable 1P3Q3T egress queuing for an eight-class QoS model.
- **Policy 2:** Trusted endpoints (Catalyst 3750-X)
  - Enable static DSCP trust.
  - Enable 1P1Q3T ingress queuing for an eight-class QoS model.
  - Enable 1P3Q3T egress queuing for an eight-class QoS model.
- **Policy 3:** Conditionally trusted endpoints—TelePresence (Catalyst 3750-X)
  - Enable conditional DSCP trust for a Cisco TelePresence System (CTS).
  - Enable 1P1Q3T ingress queuing for an eight-class QoS model.
  - Enable 1P3Q3T egress queuing for an eight-class QoS model.

- **Policy 4:** Conditionally trusted endpoints—IP phones + PCs (Catalyst 3750-X)
  - Enable conditional CoS trust for a Cisco IP phone.
  - Map CoS 5 to DSCP EF.
  - Classify and mark traffic according to Table 17-1.
  - Enable 1P1Q3T ingress queuing for an eight-class QoS model.
  - Enable 1P3Q3T egress queuing for an eight-class QoS model.

Table 17-1 *Tifosi Eight-Class Classification and Marking Policy Table for Cisco IP Phone + PC Endpoints*

Application Class	Classification Criteria	Marking
Voice	Conditionally Trusted from IP Phones	EF
Real-Time Interactive	Conditionally Trusted from Cisco TelePresence Systems	CS4
Signaling	Conditionally Trusted (VVLAN) or SCCP (DVLAN-TCP 2000) or SIP (DVLAN-TCP 5060-5061)	CS3
Multimedia Conferencing	Cisco Jabber (UDP/RTP 16384-32767) or Microsoft Lync (TCP 50000-59999)	AF4
Transactional Data	HTTPS (TCP 443) or Citrix (TCP 3389, 5985, 8080) or Oracle (TCP 1521, 1527, 1575, 1630, 6200)	AF2
Bulk Data	FTP (TCP 20 and 21) or Secure FTP (TCP 22) or SMTP (TCP 25) or Secure SMTP (TCP 465) or IMAP (TCP 143) or Secure IMAP (TCP 993) or POP3 (TCP 11) or Secure POP3 (TCP 995) or Connected PC Backup (TCP 1914)	AF1
Scavenger	BitTorrent (TCP 6881-6999) or Apple iTunes (TCP/UDP 3689) or Microsoft Direct X Gaming (TCP/UDP 2300-2400)	CS1
Best Effort	Default	DF

Note Table 17-1 presents *example* values for TCP/UDP port-based classification that are being used by Tifosi. These values may change and may not be exhaustive. Refer to application-specific documentation to verify TCP/UDP ports to be used for application classification.

- **Policy 5:** Access layer uplink ports (Catalyst 3750-X)
  - Enable static DSCP trust.
  - Enable 1P1Q3T ingress queuing for an eight-class QoS model.
  - Enable 1P3Q3T egress queuing for an eight-class QoS model.
- **Policy 6:** Distribution layer downlink ports (Catalyst 4500E Supervisor 7-E)
  - (Trust DSCP by default.)
  - Enable per-flow policing of guest VLAN traffic.
  - Enable 1P7Q1T+DBL egress queuing for an eight-class QoS model.
- **Policy 7:** Distribution layer distribution-link / core-uplink ports (Catalyst 4500E Supervisor 7-E)
  - (Trust DSCP by default.)
  - Enable 1P7Q1T+DBL egress queuing for an eight-class QoS model.
- **Policy 8:** Core layer downlink (10-GE) ports (Catalyst 6500 Supervisor 2T with X6908-10GE module)
  - (Trust DSCP by default.)
  - Enable 8Q4T ingress queuing for an eight-class QoS model.
  - Enable 1P7Q4T egress queuing for an eight-class QoS model.
- **Policy 9:** Core layer core-link (40-GE) ports (Catalyst 6500 Supervisor 2T with WS-X6904-40G-2TXL module)
  - (Trust DSCP by default.)
  - Enable 2P6Q4T ingress queuing for an eight-class QoS model.
  - Enable 2P6Q4T egress queuing for an eight-class QoS model.

Tifosi Campus Access QoS Design

Tifosi's campus access layer consists of Cisco Catalyst 3750's and includes combinations of the following policies:

- Disabling trust for printers and similar devices
- Enabling static DSCP trust for wireless access points and similar devices

- Enabling conditional trust for Cisco TelePresence Systems
- Enabling conditional trust for Cisco IP phones and PCs
- Enabling 1P1Q3T ingress queuing for an eight-class QoS model
- Enabling 1P3Q3T egress queuing for an eight-class QoS model

The configuration for each of these campus access policies is shown in turn. However, to minimize redundancy, the queuing policies—although applicable to all campus access models—are shown only at the end.

Policy 1: Access-Edge Design for Printer Endpoints (No Trust)

Example 17-1 details the QoS policy to be used on all ports connecting to untrusted endpoints.

Example 17-1 *Tifosi Access-Edge Design for Untrusted Endpoints (Printer Examples) on a Catalyst 3750*

```
C3750(config)# interface GigabitEthernet 1/0/46
C3750(config-if)# switchport access vlan 10
C3750(config-if)# no mls qos trust
 ! Port is configured to be statically untrusted
 ! Queuing configurations detailed in Examples 17-6 and 17-7
```

Policy 2: Access-Edge Design for Wireless Access Endpoints (DSCP Trust)

Example 17-2 shows the QoS policy to be used on all ports connecting to DSCP-trusted endpoints.

Example 17-2 *Tifosi Case Study: Access-Edge Design for Trusted Endpoints (Wireless Access Endpoint Example) on a Catalyst 3750*

```
C3750(config)# interface GigabitEthernet 1/0/47
C3750(config-if)# switchport access vlan 10
C3750(config-if)# mls qos trust dscp
 ! Port is configured to statically trust DSCP
 ! Queuing configurations detailed in Examples 17-6 and 17-7
```

Policy 3: Access-Edge Design for Cisco TelePresence Endpoints (Conditional Trust)

Example 17-3 shows the QoS policy to be used on all ports connecting to Cisco TelePresence Systems.

Example 17-3 *Tifosi Case Study: Access-Edge Design for Conditionally Trusted Endpoints (CTS Endpoint Example) on a Catalyst 3750*

```
C3750(config)# interface GigabitEthernet 1/0/48
C3750(config-if)# switchport access vlan 10
C3750(config-if)# switchport voice vlan 110
C3750(config-if)# spanning-tree portfast
C3750(config-if)# mls qos trust device cts
 ! Port is configured to conditionally trust the CTS endpoint
C3750(config-if)# mls qos trust dscp
 ! Port is configured to statically trust DSCP
 ! Queuing configurations detailed in Examples 17-6 and 17-7
```

Policy 4: Access-Edge Design for Cisco IP Phones or PCs (Conditional Trust and Classification and Marking)

Example 17-4 presents the QoS policy to be used on all ports connecting to Cisco IP Phones + PCs (based on Table 17-1).

Example 17-4 *Tifosi Case Study: Access-Edge Design for Conditionally Trusted Endpoints (Cisco IP Phones or PCs Example) on a Catalyst 3750*

```
 ! This section configures the classification extended ACLs
C3750(config)# ip access-list extended SIGNALING
C3750(config-ext-nacl)# remark SCCP
C3750(config-ext-nacl)# permit tcp any any eq 2000
C3750(config-ext-nacl)# remark SIP
C3750(config-ext-nacl)# permit tcp any any range 5060 5061

C3750(config)# ip access-list extended MULTIMEDIA-CONFERENCING
C3750(config-ext-nacl)# remark CISCO-JABBER-RTP
C3750(config-ext-nacl)# permit udp any any range 16384 32767
C3750(config-ext-nacl)# remark MICROSOFT-LYNC
C3750(config-ext-nacl)# permit tcp any any range 50000 59999

C3750(config)# ip access-list extended TRANSACTIONAL-DATA
C3750(config-ext-nacl)# remark HTTPS
C3750(config-ext-nacl)# permit tcp any any eq 443
```

```
C3750(config-ext-nacl)# remark CITRIX
C3750(config-ext-nacl)# permit tcp any any eq 3389
C3750(config-ext-nacl)# permit tcp any any eq 5985
C3750(config-ext-nacl)# permit tcp any any eq 8080
C3750(config-ext-nacl)# remark ORACLE
C3750(config-ext-nacl)# permit tcp any any eq 1521
C3750(config-ext-nacl)# permit tcp any any eq 1527
C3750(config-ext-nacl)# permit tcp any any eq 1575
C3750(config-ext-nacl)# permit tcp any any eq 1630
C3750(config-ext-nacl)# permit tcp any any eq 6200

C3750(config)# ip access-list extended BULK-DATA
C3750(config-ext-nacl)# remark FTP
C3750(config-ext-nacl)# permit tcp any any eq ftp
C3750(config-ext-nacl)# permit tcp any any eq ftp-data
C3750(config-ext-nacl)# remark SSH/SFTP
C3750(config-ext-nacl)# permit tcp any any eq 22
C3750(config-ext-nacl)# remark SMTP/SECURE SMTP
C3750(config-ext-nacl)# permit tcp any any eq smtp
C3750(config-ext-nacl)# permit tcp any any eq 465
C3750(config-ext-nacl)# remark IMAP/SECURE IMAP
C3750(config-ext-nacl)# permit tcp any any eq 143
C3750(config-ext-nacl)# permit tcp any any eq 993
C3750(config-ext-nacl)# remark POP3/SECURE POP3
C3750(config-ext-nacl)# permit tcp any any eq pop3
C3750(config-ext-nacl)# permit tcp any any eq 995
C3750(config-ext-nacl)# remark CONNECTED PC BACKUP
C3750(config-ext-nacl)# permit tcp any eq 1914 any

C3750(config)# ip access-list extended SCAVENGER
C3750(config-ext-nacl)# remark BITTORRENT
C3750(config-ext-nacl)# permit tcp any any range 6881 6999
C3750(config-ext-nacl)# remark APPLE ITUNES MUSIC SHARING
C3750(config-ext-nacl)# permit tcp any any eq 3689
C3750(config-ext-nacl)# permit udp any any eq 3689
C3750(config-ext-nacl)# remark MICROSOFT DIRECT X GAMING
C3750(config-ext-nacl)# permit tcp any any range 2300 2400
C3750(config-ext-nacl)# permit udp any any range 2300 2400

C3750(config)# ip access-list extended DEFAULT
C3750(config-ext-nacl)# remark EXPLICIT CLASS-DEFAULT
C3750(config-ext-nacl)# permit ip any any


 ! This section configures the class maps
```

```
C3750(config-cmap)# class-map match-all VOICE
C3750(config-cmap)#  match dscp ef
 ! VoIP is trusted (from the VVLAN)

C3750(config-cmap)# class-map match-any SIGNALING
C3750(config-cmap)#  match dscp cs3
 ! Signaling is trusted (from the VVLAN)
C3750(config-cmap)#  match access-group name SIGNALING
 ! Signaling is classified by ACL (from the DVLAN)

C3750(config-cmap)# class-map match-all MULTIMEDIA-CONFERENCING
C3750(config-cmap)#  match access-group name MULTIMEDIA-CONFERENCING
 ! Associates MULTIMEDIA-CONFERENCING access-list with class map

C3750(config-cmap)# class-map match-all TRANSACTIONAL-DATA
C3750(config-cmap)#  match access-group name TRANSACTIONAL-DATA
 ! Associates TRANSACTIONAL-DATA access-list with class map

C3750(config-cmap)# class-map match-all BULK-DATA
C3750(config-cmap)#  match access-group name BULK-DATA
 ! Associates BULK-DATA access-list with class map

C3750(config-cmap)# class-map match-all SCAVENGER
C3750(config-cmap)#  match access-group name SCAVENGER
 ! Associates SCAVENGER access-list with class map

C3750(config-cmap)# class-map match-all DEFAULT
C3750(config-cmap)#  match access-group name DEFAULT
 ! Associates DEFAULT access-list with class map


 ! This section configures the PC-MARKING policy map
C3750(config-cmap)# policy-map PC-MARKING
C3750(config-pmap-c)# class VOICE
C3750(config-pmap-c)#  set dscp ef
 ! VoIP is marked EF
C3750(config-pmap-c)# class SIGNALING
C3750(config-pmap-c)#  set dscp cs3
 ! Signaling is marked CS3
C3750(config-pmap-c)# class MULTIMEDIA-CONFERENCING
C3750(config-pmap-c)#  set dscp af41
 ! Multimedia-conferencing is marked AF41
C3750(config-pmap-c)# class TRANSACTIONAL-DATA
C3750(config-pmap-c)#  set dscp af21
 ! Transactional Data is marked AF21
```

```
C3750(config-pmap-c)# class BULK-DATA
C3750(config-pmap-c)#  set dscp af11
 ! Bulk Data is marked AF11
C3750(config-pmap-c)# class SCAVENGER
C3750(config-pmap-c)#  set dscp cs1
 ! Scavenger traffic is marked CS1
C3750(config-pmap-c)# class DEFAULT
C3750(config-pmap-c)#  set dscp default
 ! An explicit class-default marks all other IP traffic to 0


 ! This section configures conditional trust on the interfaces
 ! and applies the PC-MARKING service policy
C3750(config)# interface range GigabitEthernet 1/0/1-45
C3750(config-if-range)# switchport access vlan 10
C3750(config-if-range)# switchport voice vlan 110
C3750(config-if-range)# mls qos trust device cisco-phone
 ! The interface is set to conditionally trust Cisco IP Phones
C3750(config-if-range)# mls qos trust cos
 ! CoS trust will be dynamically extended to Cisco IP Phones
C3750(config-if-range)# service-policy input PC-MARKING
 ! Attaches the PC-MARKING policy to the interface(s)
 ! Queuing configurations detailed in Examples 17-6 and 17-7
```

Eight-Class 1P1Q3T Ingress Queuing Design

Figure 17-2 illustrates Tifosi's eight-class 1P1Q3T ingress queuing policy.

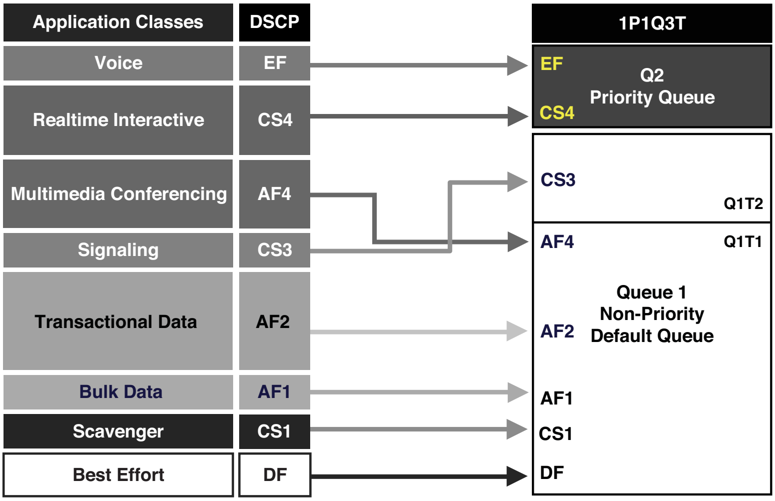


Figure 17-2 *Tifosi Case Study: Catalyst 3750 Eight-Class (1P1Q3T) Ingress Queuing Model*

Example 17-5 details the corresponding configuration for this eight-class 1P1Q3T ingress queuing policy.

Example 17-5 *Tifosi Case Study: Eight-Class 1P1Q3T Ingress Queuing on a Catalyst 3750*

```
 ! This section configures the ingress queues
C3750(config)# mls qos srr-queue input priority-queue 2 bandwidth 30
 ! Q2 is enabled as a strict-priority ingress queue with 30% BW
C3750(config)# mls qos srr-queue input bandwidth 70 30
 ! Q1 is assigned 70% BW via SRR shared weights
 ! Q2 SRR shared weight is ignored (as it has been configured as a PQ)
C3750(config)# mls qos srr-queue input buffers 90 10
 ! Q1 is assigned 90% of queuing buffers and Q2 (PQ) is assigned 10%
C3750(config)# mls qos srr-queue input threshold 1 80 90
 ! Q1 thresholds are configured at 80% (Q1T1) and 90% (Q1T2)
 ! Q1T3 is implicitly set at 100% (the tail of the queue)
 ! Q2 thresholds are all set (by default) to 100% (the tail of Q2)


 ! This section configures ingress CoS-to-Queue mappings
C3750(config)# mls qos srr-queue input cos-map queue 1 threshold 1 0 1 2
 ! CoS values 0, 1 and 2 are mapped to Q1T1
C3750(config)# mls qos srr-queue input cos-map queue 1 threshold 2 3
 ! CoS value 3 is mapped to ingress Q1T2
C3750(config)# mls qos srr-queue input cos-map queue 1 threshold 3 6 7
 ! CoS values 6 and 7 are mapped to ingress Q1T3
C3750(config)# mls qos srr-queue input cos-map queue 2 threshold 1 4 5
 ! CoS values 4 and 5 are mapped to ingress Q2 (the PQ)


 ! This section configures ingress DSCP-to-Queue Mappings
C3750(config)# mls qos srr-queue input dscp-map queue 1 threshold 1 0 8 10 12 14
 ! DSCP DF, CS1 and AF1 are mapped to ingress Q1T1
C3750(config)# mls qos srr-queue input dscp-map queue 1 threshold 1 16 18 20 22
 ! DSCP CS2 and AF2 are mapped to ingress Q1T1
C3750(config)# mls qos srr-queue input dscp-map queue 1 threshold 1 26 28 30 34 36
38
 ! DSCP AF3 and AF4 are mapped to ingress Q1T1
C3750(config)# mls qos srr-queue input dscp-map queue 1 threshold 2 24
 ! DSCP CS3 is mapped to ingress Q1T2
C3750(config)# mls qos srr-queue input dscp-map queue 1 threshold 3 48 56
 ! DSCP CS6 and CS7 are mapped to ingress Q1T3 (the tail of Q1)
C3750(config)# mls qos srr-queue input dscp-map queue 2 threshold 3 32 40 46
 ! DSCP CS4, CS5 and EF are mapped to ingress Q2T3 (the tail of the PQ)
```

Note The 8-class 1P1Q3T model configuration in Example 17-5 is identical to the 8-class or 12-class 1P1Q3T model configuration in Example 14-12. The only (optional) differences between the models are highlighted in Example 17-5. However, these are relatively minor and—if configured—will significantly expedite future class expansion.

Eight-Class 1P3Q3T Egress Queuing Design

Figure 17-3 illustrates Tifosi's eight-class 1P3Q3T egress queuing policy.

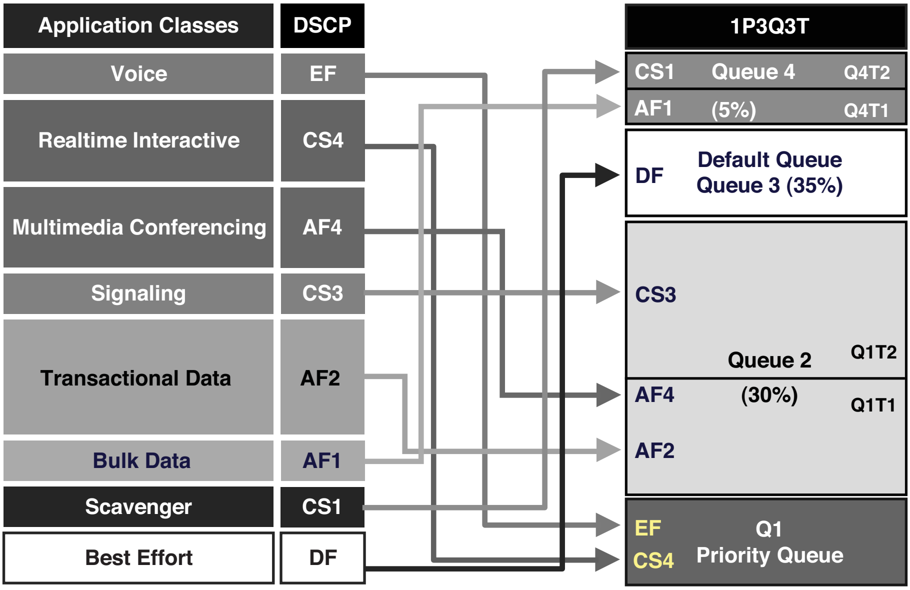


Figure 17-3 *Tifosi Case Study: Catalyst 3750 Eight-Class (1P3Q3T) Egress Queuing Model*

Example 17-6 presents the corresponding configuration for this eight-class 1P3Q3T egress queuing policy.

Example 17-6 *Tifosi Case Study: Eight-Class 1P3Q3T Egress Queuing on a Catalyst 3750*

```
 ! This section configures buffers and thresholds on Q1 through Q4
C3750(config)# mls qos queue-set output 1 buffers 15 30 35 20
 ! Queue buffers are allocated
C3750(config)# mls qos queue-set output 1 threshold 1 100 100 100 100
 ! All Q1 (PQ) Thresholds are set to 100%
C3750(config)# mls qos queue-set output 1 threshold 2 80 90 100 400
```

```
 ! Q2T1 is set to 80%; Q2T2 is set to 90%;
 ! Q2 Reserve Threshold is set to 100%;
 ! Q2 Maximum (Overflow) Threshold is set to 400%
C3750(config)# mls qos queue-set output 1 threshold 3 100 100 100 400
 ! Q3T1 is set to 100%, as all packets are marked the same weight in Q3
 ! Q3 Reserve Threshold is set to 100%;
 ! Q3 Maximum (Overflow) Threshold is set to 400%
C3750(config)# mls qos queue-set output 1 threshold 4 60 100 100 400
 ! Q4T1 is set to 60%; Q4T2 is set to 100%
 ! Q4 Reserve Threshold is set to 100%;
 ! Q4 Maximum (Overflow) Threshold is set to 400%



 ! This section configures egress CoS-to-Queue mappings
C3750(config)# mls qos srr-queue output cos-map queue 1 threshold 3 4 5
 ! CoS 4 and 5 are mapped to egress Q1T3 (the tail of the PQ)
C3750(config)# mls qos srr-queue output cos-map queue 2 threshold 1 2
 ! CoS 2 is mapped to egress Q2T1
C3750(config)# mls qos srr-queue output cos-map queue 2 threshold 2 3
 ! CoS 3 is mapped to egress Q2T2
C3750(config)# mls qos srr-queue output cos-map queue 2 threshold 3 6 7
 ! CoS 6 and 7 are mapped to Q2T3
C3750(config)# mls qos srr-queue output cos-map queue 3 threshold 3 0
 ! CoS 0 is mapped to Q3T3 (the tail of the default queue)
C3750(config)# mls qos srr-queue output cos-map queue 4 threshold 3 1
 ! CoS 1 is mapped to Q4T3 (tail of the less-than-best-effort queue)



 ! This section configures egress DSCP-to-Queue mappings
C3750(config)# mls qos srr-queue output dscp-map queue 1 threshold 3 32 40 46
 ! DSCP CS4, CS5 and EF are mapped to egress Q1T3 (tail of the PQ)
C3750(config)# mls qos srr-queue output dscp-map queue 2 threshold 1 16 18 20 22
 ! DSCP CS2 and AF2 are mapped to egress Q2T1
C3750(config)# mls qos srr-queue output dscp-map queue 2 threshold 1 26 28 30 34 36
38
 ! DSCP AF3 and AF4 are mapped to egress Q2T1
C3750(config)# mls qos srr-queue output dscp-map queue 2 threshold 2 24
 ! DSCP CS3 is mapped to egress Q2T2
C3750(config)# mls qos srr-queue output dscp-map queue 2 threshold 3 48 56
 ! DSCP CS6 and CS7 are mapped to egress Q2T3
C3750(config)# mls qos srr-queue output dscp-map queue 3 threshold 3 0
 ! DSCP DF is mapped to egress Q3T3 (tail of the best effort queue)
C3750(config)# mls qos srr-queue output dscp-map queue 4 threshold 1 8
 ! DSCP CS1 is mapped to egress Q4T1
C3750(config)# mls qos srr-queue output dscp-map queue 4 threshold 2 10 12 14
```

```
 ! DSCP AF1 is mapped to Q4T3 (tail of the less-than-best-effort queue)


 ! This section configures interface egress queuing parameters
C3750(config)# interface range GigabitEthernet1/0/1-48
C3750(config-if-range)# queue-set 1
 ! The interface(s) is assigned to queue-set 1
C3750(config-if-range)# srr-queue bandwidth share 1 30 35 5
 ! The SRR sharing weights are set to allocate 30% BW to Q2
 ! 35% BW to Q3 and 5% BW to Q4
 ! Q1 SRR sharing weight is ignored, as it will be configured as a PQ
C3750(config-if-range)# priority-queue out
 ! Q1 is enabled as a strict-priority queue
```

Note The 8-class model 1P3Q3T configuration in Example 17-6 is identical to the 8-class or 12-class model 1P3Q3T configuration in Example 14-15. The only (optional) differences between the models are highlighted in Example 17-6. However, these are relatively minor and—if configured—will significantly expedite future class expansion.

Policy 5: Access Layer Uplink Design

Example 17-7 shows the QoS policy to be used on all EtherChannel uplink ports connecting to distribution layer switches. Incidentally, in addition to the data VLAN (10) and the voice VLAN (110), an additional guest VLAN (99) is traversing the EtherChannel trunk, which will be featured in the following example.

Example 17-7 *Tifosi Access Layer EtherChannel Uplink QoS Design on a Catalyst 3750*

```
 ! This section configures EtherChannel source-and-dest load balancing
C3750(config)# port-channel load-balance src-dst-ip


 ! This section configures the (logical) EtherChannel interface
C3750(config)# interface Port-channel1
C3750(config-if)# description ETHERCHANNEL-TRUNK-TO-DISTRIBUTION-LAYER
C3750(config-if)# switchport mode trunk
C3750(config-if)# switchport trunk encapsulation dot1q
C3750(config-if)# switchport trunk allowed vlan 10,99,110


 ! This section configures Trust-DSCP and (1P3Q3T) Egress Queuing on
 ! (physical) EtherChannel member-ports
C3750(config)# interface range TenGigabitEthernet1/0/1-2
C3750(config-if-range)# description PORT-CHANNEL1-PHYSICAL-PORT-MEMBER
C3750(config-if-range)# switchport mode trunk
C3750(config-if-range)# switchport trunk encapsulation dot1q
C3750(config-if-range)# switchport trunk allowed vlan 10,99,110
C3750(config-if-range)# channel-group 1 mode auto
 ! Associates the physical ports with the logical EtherChannel bundle
C3750(config-if-range)# mls qos trust dscp
 ! The physical port-member interfaces are set to statically trust DSCP
C3750(config-if-range)# queue-set 1
 ! The interfaces are assigned to queue-set 1
C3750(config-if-range)# srr-queue bandwidth share 1 30 35 5
 ! The SRR sharing weights are set to allocate 30% BW to Q2
 ! 35% BW to Q3 and 5% BW to Q4
 ! Q1 SRR sharing weight is ignored, as it will be configured as a PQ
C3750(config-if-range)# priority-queue out
 ! Q1 is enabled as a strict-priority queue
```

Tifosi Campus Distribution QoS Design

Tifosi's campus distribution layer consists of Cisco Catalyst 4500-E Supervisor 7-E switches and includes combinations of the following policies:

- Enabling microflow policing for guest VLAN traffic
- Enabling 1P7Q1T+DBL egress queuing for an eight-class QoS model

The configuration for both of these campus distribution layer QoS policies is shown in turn. However, to minimize redundancy, the queuing policy map—although applicable to both models—is shown only once.

Policy 6: Distribution Layer Downlink Ports (Catalyst 4500E Supervisor 7-E)

Tifosi Software wants to implement per-flow policing on traffic sourced from the guest VLAN (limiting these flows to 1 Mbps each). However, the Catalyst 3750 does not have this capability, so the nearest switch that can perform this is the distribution layer Catalyst 4500E with Supervisor 7-E. The IP subnet for the guest VLAN (VLAN 99) in this example is 192.168.10.0/24.

Therefore, the QoS policy to be used on all distribution downlink ports connecting to campus access switches is detailed in Example 17-8.

Note This policy will limit traffic *sourced* from the guest VLANs. However, it may be desirable to also limit traffic *destined* to the guest VLANs, in which case a similar policy may be implemented in the reverse direction, either at this distribution switch or (more efficiently) at the Internet edge. In either case, the access control list (ACL) will need to be modified to match on the guest VLAN subnet as a destination, rather than as a source.

Note The ingress policing service policy is applied to the logical port channel interface for the downlink EtherChannel bundle, and the egress queuing policy is applied to the physical interface.

Example 17-8 *Tifosi Case Study: Distribution Layer Downlink Design on a Catalyst 4500E-Supervisor 7E*

```
 ! This section defines an ACL to match traffic from subnet
C4500(config)# ip access-list extended GUEST-VLAN-SUBNET
C4500(config-ext-nacl)# permit ip 192.168.10.0 0.0.0.255 any
 ! Traffic sourced from the 1922.168.10.x subnet is matched


 ! This section defines a flow record with source address as key
C4500(config)# flow record FLOW-RECORD-1
C4500(config-flow-record)# match ipv4 source address
 ! Source address is defined as the key tuple


 ! This section defines the class map to match on USERGROUP-1 ACL
 ! and specify FLOW-RECORD-1 definition for flow creation
C4500(config)# class-map match-all GUEST-VLAN-SUBNET
C4500(config-cmap)# match access-group name GUEST-VLAN-SUBNET
C4500(config-cmap)# match flow record FLOW-RECORD-1
 ! A "match-all" class map binds the ACL and flow-record
 ! to identify unique flows


 ! This section defines the microflow policer policy map
C4500(config)# policy-map GUEST-VLAN-1MBS-MICROFLOW-POLICER
C4500(config-pmap)# class GUEST-VLAN-SUBNET
C4500(config-pmap-c)# police cir 1m
C4500(config-pmap-c-police)# conform-action transmit
C4500(config-pmap-c-police)# exceed-action drop
 ! Specifies each discrete microflow is to be limited to 1 Mbps
! This section configures the logical port channel interface
C4500(config)# interface Port-channel1
C4500(config-if)# description ETHERCHANNEL-LOGICAL-INTERFACE
C4500(config-if)# switchport mode trunk
C4500(config-if)# switchport trunk encapsulation dot1q
C4500(config-if)# switchport trunk allowed vlan 10,99,110
C4500(config-if)# service-policy input GUEST-VLAN-1MBS-MICROFLOW-POLICER


 ! This section configures 1P3Q1T+DBL queuing on physical port-member interfaces
C4500(config)# interface range TenGigabitEthernet1/1-2
C4500(config-if-range)# description PORT-CHANNEL1-PORT-MEMBER
C4500(config-if-range)# switchport mode trunk
C4500(config-if-range)# switchport trunk encapsulation dot1q
C4500(config-if-range)# switchport trunk allowed vlan 10,99,110
C4500(config-if-range)# channel-group 1 mode auto
C4500(config-if-range)# service-policy output 1P7Q1T-QUEUING
 ! The 1P7Q1T-QUEUING is detailed in Example 17-10
```

Policy 7: Distribution Layer Distribution-Link / Core-Uplink Ports

Figure 17-4 illustrates Tifosi's eight-class 1P7Q1T+DBL egress queuing policy.

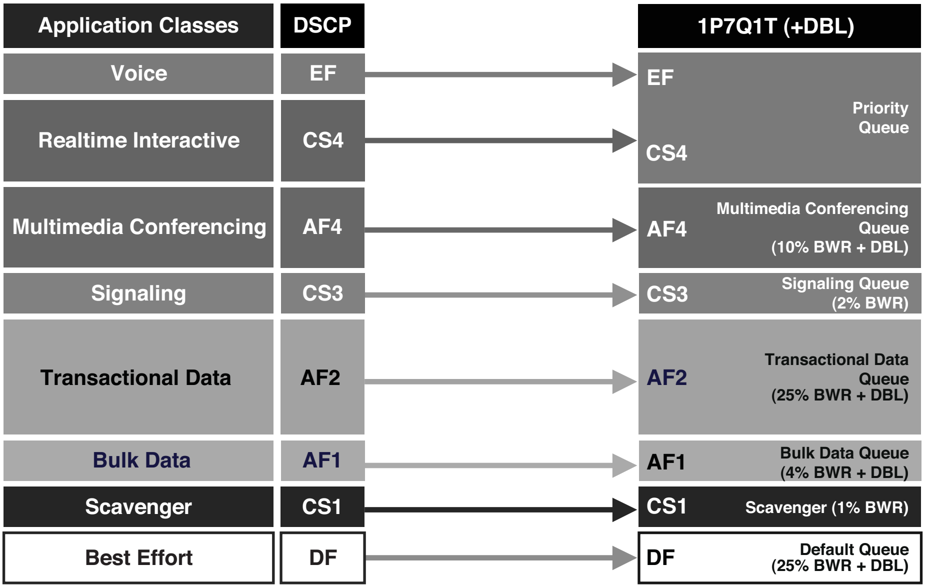


Figure 17-4 *Tifosi Case Study: Catalyst 4500E-Supervisor 7-E Eight-Class (1P7Q1T+DBL) Egress Queuing Model*

The corresponding configuration for this eight-class 1P7Q1T+DBL egress queuing policy is detailed in Example 17-9.

Example 17-9 *Tifosi Case Study: Eight-Class 1P7Q1T+DBL Egress Queuing on a Catalyst 4500*

```
! This section configures the class maps for the egress queuing policy
C4500(config)# class-map match-any PRIORITY-QUEUE
C4500(config-cmap)# match dscp ef
 ! VoIP (EF) is mapped to the PQ
C4500(config-cmap)# match dscp cs4
 ! Realtime Interactive (CS4) is also mapped to the PQ
C4500(config)# class-map match-all MULTIMEDIA-CONFERENCING-QUEUE
C4500(config-cmap)# match dscp af41 af42 af43
 ! Multimedia-Conferencing (AF4) is assigned a dedicated BW queue
C4500(config)# class-map match-all SIGNALING-QUEUE
C4500(config-cmap)# match dscp cs3
 ! Signaling (CS3) is mapped to a dedicated BW queue
C4500(config)# class-map match-all TRANSACTIONAL-DATA-QUEUE
C4500(config-cmap)# match dscp af21 af22 af23
 ! Transactional Data (AF2) is assigned a dedicated queue
C4500(config)# class-map match-all BULK-DATA-QUEUE
C4500(config-cmap)# match dscp af11 af12 af13
 ! Bulk Data (AF1) is assigned a dedicated queue
C4500(config)# class-map match-all SCAVENGER-QUEUE
C4500(config-cmap)# match dscp cs1
 ! Scavenger (CS1) is assigned a dedicated queue


 ! This section configures the 1P7Q1T+DBL egress queuing policy map
C4500(config)# policy-map 1P7Q1T+DBL
C4500(config-pmap-c)# class PRIORITY-QUEUE
C4500(config-pmap-c)#  priority
 ! Defines a priority queue
C4500(config-pmap-c)# class MULTIMEDIA-CONFERENCING-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 10
C4500(config-pmap-c)#  dbl
 ! Defines a multimedia-conferencing queue with 23% BW remaining + DBL
C4500(config-pmap-c)# class SIGNALING-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 2
 ! Defines a signaling queue with 2% BW remaining
C4500(config-pmap-c)# class TRANSACTIONAL-DATA-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 25
C4500(config-pmap-c)#  dbl
 ! Defines a transactional data queue with 25% BW remaining + DBL
```

```
C4500(config-pmap-c)# class BULK-DATA-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 4
C4500(config-pmap-c)#  dbl
 ! Defines a bulk data queue with 4% BW remaining + DBL
C4500(config-pmap-c)# class SCAVENGER-QUEUE
C4500(config-pmap-c)#  bandwidth remaining percent 1
 ! Defines a (minimal) scavenger queue with 1% BW remaining/limit
C4500(config-pmap-c)# class class-default
C4500(config-pmap-c)#  bandwidth remaining percent 25
C4500(config-pmap-c)#  dbl
 ! Provisions the default/Best Effort queue with 25% BW remaining + DBL


 ! This section attaches the egress queuing policy to the interface(s)
C4500(config)# interface range TenGigabitEthernet 1/1-8
C4500(config-if-range)# service-policy output 1P7Q1T+DBL
```

Note Technically speaking, Figure 17-4 and Example 17-9 present a 1P6Q1T+DBL model (as two classes are mapped to the PQ, leaving only six classes to be mapped to the nonpriority hardware queues). However, to avoid confusion, the designation of 1P7Q1T+DBL is used in this text.

Tifosi Campus Core QoS Design

Tifosi's campus core layer consists of Cisco Catalyst 6500 Supervisor 2T switches (with both WS-X6908-10GE and WS-X6904-40G-2TXL modules) and includes the following policies:

- Enable 8Q4T/1P7Q4T ingress/egress queuing for an eight-class QoS model (for 10GE interfaces).
- Enable 2P6Q4T ingress/egress queuing for an eight-class QoS model (for 40GE interfaces).

In this example, the downlinks to the distribution layer are 10GE, and the core links are 40GE.

Policy 8: Core Layer (10GE) Downlink Design

Figure 17-5 illustrates Tifosi's eight-class 8Q4T Ingress and 1P7Q4T egress queuing models for 10GE interfaces.

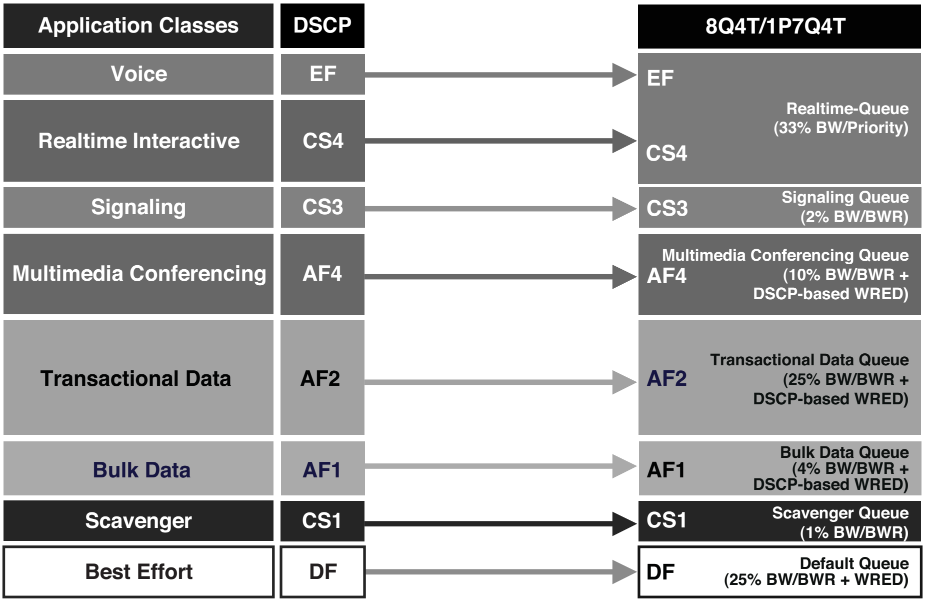


Figure 17-5 *Tifosi Case Study: Catalyst 6500-Supervisor 2T Eight-Class (8Q4T Ingress and 1P7Q4T Egress) Queuing Models for 10GE Interfaces*

Example 17-10 shows the corresponding configuration for this eight-class 8Q4T ingress and 1P7Q4T egress queuing policy.

Example 17-10 *Tifosi Case Study: Eight-Class 8Q4T Ingress and 1P7Q4T Egress Queuing on a Catalyst 6500*

```
 ! This section configures the class maps for the queuing policy maps
 ! Note both ingress and egress policy maps share these common class maps
C6500(config-cmap)# class-map type lan-queuing REALTIME-QUEUE
C6500(config-cmap)#  match dscp ef cs4
 ! Voice (EF) and TelePresence (CS4) are mapped to the REALTIME-QUEUE
C6500(config-cmap)# class-map type lan-queuing SIGNALING-QUEUE
C6500(config-cmap)#  match dscp cs3
 ! Signaling (CS3) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing MULTIMEDIA-CONF-QUEUE
C6500(config-cmap)#  match dscp af41 af42 af43
 ! Multimedia-Conf (AF4) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing TRANSACTIONAL-DATA-QUEUE
C6500(config-cmap)#  match  dscp af21  af22  af23
 ! Transactional Data (AF2) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing BULK-DATA-QUEUE
C6500(config-cmap)#  match  dscp af11  af12  af13
 ! Bulk Data (AF1) is mapped to a dedicated nonpriority queue
```

```
C6500(config-cmap)# class-map type lan-queuing SCAVENGER-QUEUE
C6500(config-cmap)#  match dscp cs1
 ! Scavenger (CS1) is mapped to a bandwidth-constrained queue



 ! This section configures the eight-class ingress (8Q4T) policy map
C6500(config-pmap)# policy-map type lan-queuing INGRESS-8Q4T
C6500(config-pmap)#  class REALTIME-QUEUE
C6500(config-pmap-c)#   bandwidth percent 33
 ! Defines the REALTIME-QUEUE with 33% BW
C6500(config-pmap-c)#  class SIGNALING-QUEUE
C6500(config-pmap-c)#   bandwidth percent 2
 ! Defines the Signaling-Queue with 2% BW
C6500(config-pmap-c)#  class MULTIMEDIA-CONF-QUEUE
C6500(config-pmap-c)#   bandwidth percent 10
 ! Defines the Multimedia-Conferencing-Queue with 10% BW
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af41 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af42 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af43 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF4 PHB
C6500(config-pmap-c)#  class TRANSACTIONAL-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth percent 25
 ! Defines the Transactional-Data-Queue with 25% BW
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af21 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af22 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af23 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF2 PHB
C6500(config-pmap-c)#  class BULK-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth percent 4
 ! Defines the Bulk-Data-Queue with 4% BW
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af11 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af12 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af13 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF1 PHB
C6500(config-pmap-c)#  class SCAVENGER-QUEUE
C6500(config-pmap-c)#   bandwidth percent 1
 ! Defines the Scavenger-Queue with 1% BW
C6500(config-pmap-c)#  class class-default
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp default percent 80 100
 ! DSCP-based WRED is enabled and tuned for default DSCP (0)
 ! This section configures the eight-class egress (1P7Q4T) policy map
C6500(config-pmap)# policy-map type lan-queuing EGRESS-1P7Q4T
C6500(config-pmap)#  class REALTIME-QUEUE
C6500(config-pmap-c)#   priority
 ! Enables strict-priority queuing on the REALTIME-QUEUE
C6500(config-pmap-c)#  class SIGNALING-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 2
 ! Defines the Signaling-Queue with 2% BW remaining
C6500(config-pmap-c)#  class MULTIMEDIA-CONF-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 10
 ! Defines the Multimedia-Conferencing-Queue with 10% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af41 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af42 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af43 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF4 PHB
C6500(config-pmap-c)#  class TRANSACTIONAL-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 25
 ! Defines the Transactional-Data-Queue with 25% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af21 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af22 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af23 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF2 PHB
C6500(config-pmap-c)#  class BULK-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 4
 ! Defines the Transactional-Data-Queue with 4% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af11 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af12 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af13 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF1 PHB
C6500(config-pmap-c)#  class SCAVENGER-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 1
 ! Defines the Scavenger-Queue with 1% BW remaining
C6500(config-pmap-c)#  class class-default
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp default percent 80 100
 ! DSCP-based WRED is enabled and tuned for default DSCP (0)


 ! This section attaches the ingress and egress queuing policies
 ! to the interface(s)
C6500(config)# interface range TenGigabitEthernet 3/1-8
C6500(config-if-range)# service-policy type lan-queuing input INGRESS-8Q4T
 ! Attaches the INGRESS-8Q4T queuing policy to the interfaces
C6500(config-if-range)# service-policy type lan-queuing output EGRESS-1P7Q4T
 ! Attaches the EGRESS-1P7Q4T queuing policy to the interfaces
```

Note Technically speaking, Figure 17-5 and Example 17-10 present a 1P6Q4T egress queuing model (as two classes are mapped to the PQ, leaving only six classes to be mapped to the nonpriority hardware queues). However, to avoid confusion, the designation of 1P7Q4T is used in this text.

Policy 9: Core Layer (40GE) Core-Link Design

Figure 17-6 illustrates Tifosi's eight-class 2P6Q4T Ingress and Egress queuing models for 40GE interfaces.

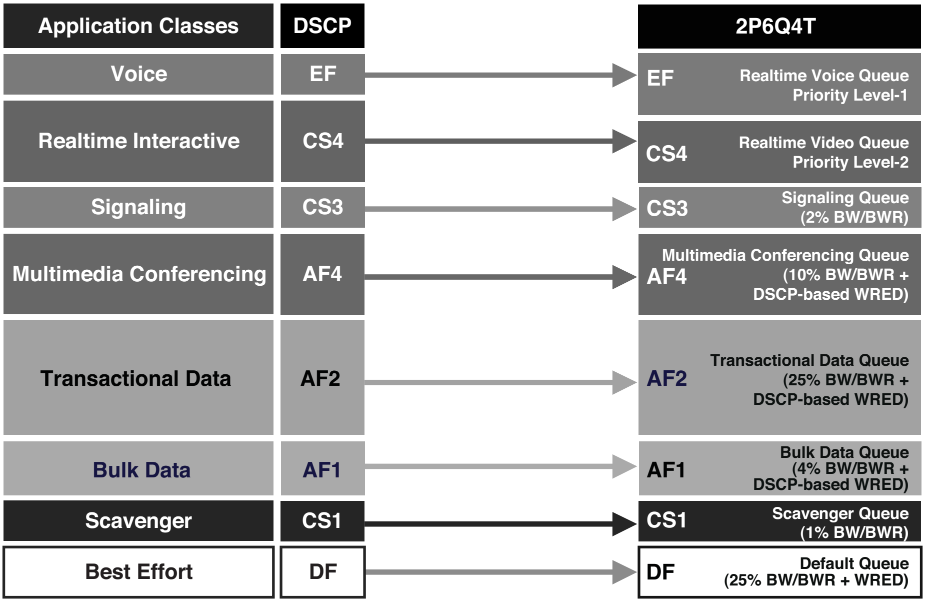


Figure 17-6 *Tifosi Case Study: Catalyst 6500-Supervisor 2T Eight-Class (2P6Q4T Ingress and Egress) Queuing Models for 40GE Interfaces*

Example 17-11 provides the corresponding configuration for this eight-class 2P6Q4T ingress and egress queuing policy.

Example 17-11 *Tifosi Case Study: Eight-Class 2Q6Q4T-Ingress and Egress Queuing on a Catalyst 6500*

```
 ! This section configures the class maps for the queuing policy map
C6500(config-cmap)# class-map type lan-queuing REALTIME-VOICE-QUEUE
C6500(config-cmap)#  match dscp ef
 ! Voice (EF) is mapped to the REALTIME-VOICE-QUEUE
C6500(config-cmap)# class-map type lan-queuing REATIME-VIDEO-QUEUE
C6500(config-cmap)#  match dscp ef cs4
 ! Realtime-Interactive (CS4) is mapped to the REALTIME-VIDEO-QUEUE
C6500(config-cmap)# class-map type lan-queuing SIGNALING-QUEUE
C6500(config-cmap)#  match dscp cs3
 ! Signaling (CS3) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing MULTIMEDIA-CONF-QUEUE
C6500(config-cmap)#  match dscp af41 af42 af43
 ! Multimedia-Conf (AF4) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing TRANSACTIONAL-DATA-QUEUE
C6500(config-cmap)#  match  dscp af21  af22  af23
 ! Transactional Data (AF2) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing BULK-DATA-QUEUE
C6500(config-cmap)#  match  dscp af11  af12  af13
 ! Bulk Data (AF1) is mapped to a dedicated nonpriority queue
C6500(config-cmap)# class-map type lan-queuing SCAVENGER-QUEUE
C6500(config-cmap)#  match dscp cs1
 ! Scavenger (CS1) is mapped to a bandwidth-constrained queue
! This section configures the Eight-Class (2P6Q4T) policy map
C6500(config-pmap)# policy-map type lan-queuing 2P6Q4T
C6500(config-pmap)#  class REALTIME-VOICE-QUEUE
C6500(config-pmap-c)#   priority level 1
 ! Enables strict-priority queuing on the REALTIME-VOICE-QUEUE
C6500(config-pmap)#  class REALTIME-VIDEO-QUEUE
C6500(config-pmap-c)#   priority level 2
 ! Enables level-2 priority queuing on the REALTIME-VIDEO-QUEUE
C6500(config-pmap-c)#  class SIGNALING-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 2
 ! Defines the Signaling-Queue with 2% BW remaining
C6500(config-pmap-c)#  class MULTIMEDIA-CONF-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 10
 ! Defines the Multimedia-Conferencing-Queue with 10% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af41 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af42 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af43 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF4 PHB
C6500(config-pmap-c)#  class TRANSACTIONAL-DATA-QUEUE
```

```
C6500(config-pmap-c)#   bandwidth remaining percent 25
 ! Defines the Transactional-Data-Queue with 25% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af21 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af22 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af23 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF2 PHB
C6500(config-pmap-c)#  class BULK-DATA-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 4
 ! Defines the Transactional-Data-Queue with 4% BW remaining
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp af11 percent 80 100
C6500(config-pmap-c)#   random-detect dscp af12 percent 70 100
C6500(config-pmap-c)#   random-detect dscp af13 percent 60 100
 ! DSCP-based WRED is enabled and tuned for AF1 PHB
C6500(config-pmap-c)#  class SCAVENGER-QUEUE
C6500(config-pmap-c)#   bandwidth remaining percent 1
 ! Defines the Scavenger-Queue with 1% BW remaining
C6500(config-pmap-c)#  class class-default
C6500(config-pmap-c)#   random-detect dscp-based
C6500(config-pmap-c)#   random-detect dscp default percent 80 100
 ! DSCP-based WRED is enabled and tuned for default DSCP (0)



 ! This section attaches the ingress and egress queuing policies
 ! to the interface(s)
C6500(config)# interface FortyGigabitEthernet 1/1
C6500(config-if)# service-policy type lan-queuing input 2P6Q4T
 ! Attaches the 2P6Q4T queuing policy in the ingress direction
C6500(config-if)# service-policy type lan-queuing output 2P6Q4T
 ! Attaches the 2P6Q4T queuing policy in the egress direction
```

Summary

This chapter continued the case study example of Tifosi Software and applied their strategic eight-class end-to-end QoS model to various Catalyst switching platforms in their multitier enterprise campus network.

Campus access layer policies were presented for untrusted, trusted, and conditionally trusted endpoints, as were explicit classification and marking policies, along with ingress and egress queuing policies for the Catalyst 3750.

Campus distribution layer policies included leveraging the Catalyst 4500 flow-based policing feature in addition to egress queuing policies.

Campus core layer policies for Catalyst 6500 platforms included both 10 GE and 40 GE ingress and egress queuing policies.

Further Reading

Cisco Enterprise Medianet Campus QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html

Medianet Campus QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampusaag.html

Medianet Catalyst 3560/3750 QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampuscat3xxxaag.html

Cisco Catalyst 3750-X and Catalyst 3560-X Switch Software Configuration Guide, Cisco IOS Release 15.0(2)SE—QoS Configuration Guide: http://www.cisco.com/en/US/docs/switches/lan/catalyst3750x_3560x/software/release/15.0_2_se/configuration/guide/swqos.html

Medianet Catalyst 4500 QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampuscat4500aag.html

Cisco Catalyst 4500 Series Switch Software Configuration Guide, Release IOS XE 3.3.0SG and IOS 15.1(1)SG—QoS Configuration Guide: http://www.cisco.com/en/US/docs/switches/lan/catalyst4500/15.1/XE_330SG/configuration/guide/qos_mrg.html

Medianet Catalyst 6500 QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampuscat6500aag.html

Cisco Catalyst 6500 Series Switch IOS Release 15.1SY Supervisor Engine 2T Software Configuration Guide—QoS Configuration Guide (requires a Cisco.com login): http://www.cisco.com/en/US/partner/docs/switches/lan/catalyst6500/ios/15.1SY/config_guide/sup2T/qos_overview.html

This page intentionally left blank

Chapter 18

Wireless LAN QoS Considerations and Recommendations

The primary role of quality of service (QoS) in the wireless networks is to reduce the latency and jitter of real-time applications over the wireless media. A secondary role of QoS in the WLAN is to provide application-management for traffic originating on wireless devices, including classification (which may incorporate deep packet inspection), marking, and policing.

Therefore, several of the strategic QoS design principles discussed in Chapter 11, "QoS Design Principles and Strategies," apply to WLAN QoS designs, including the following:

- **Classify and mark applications as close to their sources as technically and administratively feasible:** Traffic originating on wireless devices may have their differentiated service code point (DSCP) markings set right on the device. You may choose to trust or to re-mark these settings, in accordance with your enterprise strategic QoS model. In addition, deep packet inspection engines—such as application visibility control (AVC)—may be available to correctly identify WLAN-sourced or destined traffic.
- **Police unwanted traffic flows as close to their sources as possible:** You may want to place bandwidth limits on specific WLANs, specific applications, or specific users (or combinations thereof).
- **Enable queuing policies at every node where the potential for congestion exists:** Recognize that wireless media is different from wired, and therefore real-time flows need to be optimized over this transport.

To better understand how to implement these strategic QoS principles over wireless networks, a review of wireless networking technologies may be helpful.

Designing an 802.11 network that supports real-time applications such as Voice over IP (VoIP) and video introduces a new set of challenges and a whole different paradigm compared to the traditional wired Ethernet network. Although there are certain familiar aspects used in WLAN QoS, such as traffic prioritization based on DSCP values, the core differences in the underlying MAC and PHY layer protocols used in Wi-Fi networks have necessitated the development of a completely new set of QoS tools just for the wireless medium.

The 802.11 QoS toolset has slowly matured over the past decade, with much progress being made by the IEEE 802.11e Working Group (WG). In 2007, the WLAN QoS enhancements proposed by the 802.11e WG were rolled into a new wider definition of the 802.11 standard, called IEEE 802.11-2007, which includes the current definitive standard for wireless QoS. In 2012, a series of further revisions were rolled into a new standard called IEEE 802.11-2012. However, keep in mind that although the IEEE 802.11 body sets the standards, they are not responsible for ensuring that the various vendors comply with this standard. To address this, the Wi-Fi Alliance has formed a wireless QoS compatibility standard (which is based on the 802.11e enhancements and 802.11-2012 standard), called Wireless Multimedia, or more commonly known as WMM.

Note To avoid confusion, unless otherwise called out, the terms WMM and 802.11e are used interchangeably.

This chapter begins with a discussion of the foundational building blocks of the IEEE 802.11 MAC layer and how data is transmitted onto the wireless medium. This discussion might seem to have little to do with QoS specifically, but a fundamental understanding of these concepts is critical to comprehending how WLAN QoS works and is ultimately designed in your network. Next, this chapter examines how the 802.11e and WMM QoS standards work to provide differentiated levels of service to different types of traffic and how these can be integrated into your network.

Finally, several WMM QoS design considerations are discussed, including the various QoS marking and queuing mechanisms that are used in controller-based wireless networks. This discussion leads to the key design recommendations in this chapter.

Comparing QoS in Wired and Wireless LAN Environments

Recommendation:

- Understand the fundamental differences between wired and wireless networks, particularly as these pertain to QoS.

Although most LAN networks operate far below full capacity most of the time, microbursts sometimes traffic cause congestion and therefore require the more important traffic types to be delivered in priority sequence. This is a well-understood concept of QoS in wired networks, and it applies just as well to wireless networks. Whether QoS is deployed on a copper Gigabit Ethernet port or on a wireless access point (AP) radio interface, QoS tools are responsible for managing how packets are transmitted onto the network in a reliable way. On a well-designed wired network, it is possible to make the reliable delivery of high-priority traffic almost a certainty. However, in wireless networks, because of their inherent half-duplex nature, this same level of certainty is not possible.

In Wi-Fi networks, every station associated to a particular AP must share the radio frequency (RF) with all the other stations. However, the physical constraint is that only one station, including the AP itself, may transmit at a given time. The result of this is that each station must contend with all the other stations for airtime. WLANs are not half duplex by choice. Wireless is, by definition, a multiple-access, broadcast medium—meaning that if more than one station transmits at any one time, the receiver cannot understand what either sender said.

This situation is quite familiar to humans. Have you ever tried to listen to two conversations at one time? Although our brains have the ability interpret even the subtlest sounds coming from all around us, it is almost impossible to effectively listen to more than one sound (or person) at a time. The brain's limbic system is responsible for sorting out what we really want to listen to versus all other background information. Therefore, if more than one person tries to talk to us at the same time, this audio interference usually results in not being able to understanding anything at all! Another similar example is if you and a person you want to talk to try to speak at the same time. Instead of doubling the efficiency of your conversation, neither of you would be able to understand what the other person is saying. This effectively demonstrates how half-duplex communications network functions.

You would be quite correct to compare this to a hub environment. Not only is a hub a half-duplex medium, but it is also a shared broadcast medium. QoS in a hub seems like an unlikely possibility, and you can probably think of all kinds of reasons it would not ever work, and this is exactly the challenge faced by in 802.11 wireless networks.

To illustrate this, contrast the half-duplex shared medium of wireless to a modern wired Ethernet network. Wired Ethernet operating in full-duplex mode creates a point-to-point link between each two stations, allowing the simultaneous transmit and receive of Ethernet frames. For example, a server connected to a switch on a 1-Gbps full-duplex link can theoretically both send and receive at 1 Gbps simultaneously without having to contend with other stations for access to the medium.

Now compare this with how a wireless AP communicates. Because the RF spectrum used by an AP and all it is connected stations is shared, only one station can transmit at any given time on a given channel without causing interference. If two stations were to transmit at the same time, neither transmission would be understood, thus causing a *collision*. This physical limitation of the wireless medium means that all stations—even the AP itself md]must contend for a chance to transmit their traffic. If there were no mechanism to coordinate whose turn it is to transmit onto the wireless medium, chaos would reign, and the Wi-Fi network would be all but useless.

To muddy the waters even further, unlike wired networks that operate at fixed speeds (such as 1 Gbps, 10 Gbps, and so on), wireless networks operate at variable speeds. The speed of a wireless connection is influenced by such things as interference patterns, signal-to-noise ratio (SNR), and overall signal strength. To make matters even more interesting, a single AP may have associated clients that are all sending and receiving at different data rates.

QoS tools in wired networks are chiefly responsible for managing which packet, according to its class, is sent next onto the wire. In a Wi-Fi network, the job of QoS is far more complicated. Because the wireless medium is both shared and half duplex, the QoS toolset must manage priority access to the RF channel for all end-station transmissions in an organized and predictable way.

WLAN QoS Building Blocks

Recommendation:

- Understand how WLAN QoS has evolved to its current state.

In the early days of 802.11, there was literally no standards-based approach that supported wireless QoS. If you were to add IP phones or other real-time applications to a wireless network, they would all try to access the medium on a best effort basis. In situations like this, the quality was often poor, and in some cases critical applications would not even work at all.

The following section examines the fundamentals of the 802.11 MAC layer and the technical limitations related to QoS. Even though the first incarnation of the 802.11 MAC layer definition had no ability to support QoS, a good grasp of how it works will enable you to understand the modifications introduced by the 802.11e WG that have made WLAN QoS a reality.

The Distributed Coordination Function

Before the IEEE developed a systematic way to address the QoS challenges faced in Wi-Fi networks, the Distributed Coordination Function (DCF), operating at the 802.11 MAC layer, was responsible for scheduling and transmitting Ethernet frames onto the wireless medium. Although much has changed since the early days of DCF, it still remains the foundation for modern 802.11 MAC layer operation. This fact alone makes DCF an important subject; a good grasp of DCF is also fundamental to understanding the modern 802.11 QoS toolset.

Wi-Fi networks are completely egalitarian, meaning that all wireless stations have equal access to the medium. In fact, even the AP has no more priority to access the medium than the client stations do. For example, a wireless IP phone has to abide by exactly the same principles as a wireless laptop, regardless of the fact that one of them is transmitting real-time VoIP traffic and the other might be transmitting P2P traffic. Because each station, and therefore each application, has equal opportunity to transmit frames at any given time, there needs to be an orderly system to coordinate the transmission of packets onto the medium. If no control were implemented, you can easily imagine a situation where anytime a client wanted to send a frame onto the air there would be a high probability of a collision. The more clients that are associated to the AP, the higher the likelihood of more collisions occurring. In turn, after each collision, the end stations would attempt to retransmit, thus causing even more collisions, until the situation snowballs to the point where the network is all but paralyzed.

CSMA/CA

A similar problem was encountered in the early days of wired Ethernet, when half-duplex links and hubs were common. In half-duplex wired Ethernet environments, collisions were a natural outcome of multiple stations trying to send frames on the wire at the same time. To address this situation, a system called carrier sense multiple access with collision detection (CSMA/CD) was developed. CSMA/CD is a set of transmission and retransmission rules that all end stations are required to follow when trying to send a frame over a medium. According to the CSMA/CD rules, the sending station must wait until the medium is idle before sending its frame. Once it does so, the sending station listens to see whether a collision occurred. If there was a collision, the station waits a random backoff period before resending the frame.

Could CSMA/CD be used to solve the same problems encountered in 802.11 wireless networks? Similar to hubs, wireless networks are also inherently half-duplex. However, wireless networks bring their own unique challenges that cannot be solved simply by applying CSMA/CD to the problem. A key point to consider is that wireless stations have no way to detect a collision because each transmission is a broadcast over the open air. In addition, some wireless clients may experience the “hidden node” problem—that is, two transmitting clients might not be able to see each other when listening to see whether the medium is idle (because they are either too far away from each other or an obstruction sits between them). This may result in both stations attempting to send at the same time as soon as the medium is free, thus causing a collision.

In an effort to alleviate the collision problem in Wi-Fi networks, CSMA/CD was modified into carrier sense multiple access with *collision avoidance* (CSMA/CA). To break down the difference between these two systems, CSMA/CD deals with what to do *after* a collision occurs, whereas CSMA/CA works to *prevent* a collision before it occurs, plus how to handle retransmission if a collision does occur.

To understand this better, take the simple analogy of a telephone conference call. If many people are on a call together, there is a good chance that as soon as the line is quiet more than one person may try to talk at the same time, making both speakers unintelligible to everyone else. (The two speakers are causing a collision.) If this were CSMA/CD, when the two parties realized that they just talked at the same time, they would randomly pause for a few seconds and then try talking again in the hopes that they will not talk over each other the next time around. This process would repeat itself until one of them owns the floor.

CSMA/CA, in contrast, is a much more polite system. Instead of just starting to speak when the line seems free (and hoping you don’t talk at the same time as someone else), everyone waits patiently for his or her own random period once the line becomes quiet. If the call continues to be quiet, you can begin speaking after your random wait time has expired. When you begin talking, everyone else must respect the fact that you own the floor and that they cannot interrupt you.

As with the conference call illustration, CSMA/CA opts to listen and wait to see whether any transmissions are in progress, and only after it has waited its random backoff period does it attempt to send a frame. If a collision does happen occur even after listening and waiting (because two stations may start a transmission at exactly the same moment), the wireless station deals with it in a similar way to CSMA/CD—by waiting for another random backoff period before it tries to resend the frame.

Note that CSMA/CA can never fully guarantee that a collision won't occur; rather, it simply reduces the *probability* that a collision will occur by trying to avoid a future collision. CSMA/CA is something like stopping your car at a four-way stop. Although you might try very hard to avoid a collision by carefully looking in all directions before driving into the intersection, you can never fully guarantee what other drivers will do. If you decide to step on the gas because it *looks* like everyone else is letting you go, a slight possibility always exists that another driver might do the same thing at the same time; thus, possibility of a collision never fully goes away. The same goes for WLAN stations that operate using CSMA/CA: Even though the stations listen before sending, they can never fully guarantee a collision will not occur after the data is transmitted.

So, how does a sending station know that its transmission was successful? Due to the broadcast nature of the wireless medium, collisions cannot actually be detected. They can only be avoided with a measure of probability, yet there is an obvious need to confirm whether a transmission was successful or not. To solve this problem, DCF ensures that each frame is acknowledged after the transmission has been successfully received. For example, if a wireless station transmits and does not receive an acknowledgment, the wireless station knows that it must resend the frame. If an acknowledgment is received, the wireless station knows it can move on to the next frame.

An interesting question is raised when you consider the possibility that an acknowledgment frame itself might get caught in a collision. How is this possibility avoided? To deal with this possibility, there is a provision in DCF where all clients must keep silent after a transmission finishes so that the receiving station has a chance to send the acknowledgment. This is period is called the short interframe space (SIFS).

The DCF Contention Window

To help control and organize the transmission of frames on the wireless medium, DCF uses some clever rules where contending stations wait for *different* periods of time before they can transmit their frames onto the channel. A central and key concept of how DCF operates is the DCF interframe space (DIFS). DIFS is a preestablished, fixed wait timer observed by all stations before they begin an attempt to transmit a frame onto the channel.

> **Note** Several interframe space (IFS) types are used in 802.11 networks, such as SIFS (mentioned earlier) and Point-coordinated IFS (PIFS). This discussion focuses on DIFS, though, because it more directly relates to the IEEE 802.11e QoS toolset.

So how does the DIFS timer work to help avoid collisions on a wireless network? As mentioned earlier, CSMA/CA provides a framework of "listen before you talk" for wireless stations. When a wireless station wants to transmit a frame, the first thing it does is to wait the appropriate DIFS time. Once this DIFS countdown has expired, if the medium is still clear it transmits the frame. DIFS is like a level set for all stations that want to transmit. If they all just started transmitting as soon as they had a frame in the queue, collisions would be plentiful. However, by waiting the DIFS period, it gives a chance for each station to confirm that the channel is indeed clear for transmission.

What about case when a station detects that the medium is not idle? If this is the case, the station must wait (technically speaking, the station will *defer*) for a random period of time, called the *contention window*, or CW. The first time a station needs to defer, the CW backoff is a randomly set from a minimum of zero to a maximum value known as CW_{min}. There is an obvious advantage to waiting a random period of time: If multiple stations were to attempt a retransmission at the exact same time, collisions would continually occur, and the channel would never clear up. After the CW timer expires, the station again looks to see whether the medium is free. If the channel is free, the station begins sending the frame. Figure 18-1 demonstrates this process.

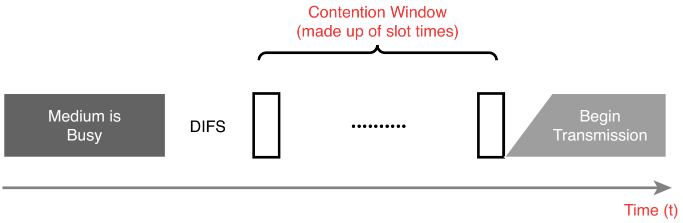


Figure 18-1 *Sending a Frame After the CW Random Back-Off Period*

So, what happens if the medium is still not clear after the CW timer expires? In this case, the sending station does the following:

1. The station first defers again until the wireless medium is clear.
2. All stations must now wait for the DIFS period to expire.
3. Double the CW value was previously used and defer for this new random period of time.
4. If the medium is free, send the data.

If the station finds that the medium is still not clear after waiting a larger CW value, this process repeats, and the station continues doubling the backoff window each time it tries to resend the frame. It continues to this up to a maximum amount of time, known as the CW_{max} value, until it either transmits the frame or the Time to Live (TTL) expires and the frame is dropped. After the CW countdown timer has finished and the medium appears to be free, the frame is finally sent.

So, what is the actual amount of time that the station counts down? The CW is not measured in seconds, but rather in *slot times*. The slot time is a time value derived from the PHY based on the RF characteristics of the radio network, so it is unique for each network, but the actual time is measured in microseconds. For example, in the case of IEEE 802.11n, the CW_{min} default is 15 slot times, and CW_{max} is 1023 slot times.

To illustrate how this works, if the random backoff for a station was initially set to 10 slot times, the second backoff would be 20. If the channel is still busy, the CW backoff is increased to 40, then 80, then 160, and so on up to a maximum of 1023 slot times. Once the CW is set, the station must count down this number of slot times until zero and then it attempts to retransmit the frame again.

Figure 18-2 illustrates the overall DCF decision process for a station that wants to send a frame onto the wireless medium.

Now that the DCF frame transmission process has been examined, let’s take a look at an example of how this might apply to a real situation of sending data from a wireless station. In the following example, illustrated in Figure 18-3, five stations are associated with the same AP and all are trying to send data at approximately the same time.

In this example, Station A is already sending an Ethernet frame. Stations B, C, and D would all like to send as well, but because Station A is in the midst of a transmission, the other stations must all wait and defer until the channel is clear. Notice from Figure 18-3 that when Station A finishes transmission, all other stations must first wait the DIFS period before anyone can attempt to send.

When the DIFS period has expired, the remaining three stations all want to transmit at the same time, so they must contend for their chance. Because the three stations are all competing for access to the medium, they all run a contention window random backoff timer. As can be seen from this example, as Station B generates the smallest random backoff period, it sends first.

While Station B is transmitting, the backoff timer expires for Stations C and D. However, they must both wait until Station B has finished sending its frame before they can try again. In harmony with DCF’s “listen before you talk” approach, Stations C and D patiently wait for Station B to finish. Notice that during Station B’s transmission, Station E suddenly wants to send as well, so now there are again three stations contending for access to the medium. When Station B finishes, the process begins where all three stations wait for the DIFS period to elapse, and they all generate a random timer. In this case, Station D wins and begins to send while Stations C and E must wait for another turn.

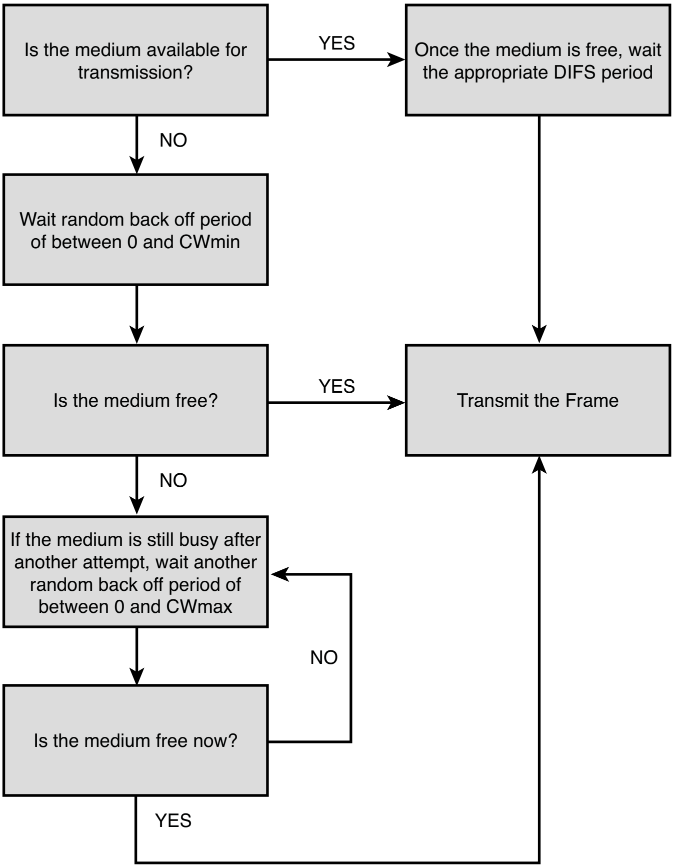

Is the medium available for transmission?
YES
Once the medium is free, wait the appropriate DIFS period
NO
Wait random back off period of between 0 and CWmin
Is the medium free?
YES
Transmit the Frame
NO
If the medium is still busy after another attempt, wait another random back off period of between 0 and CWmax
NO
Is the medium free now?
YES

Figure 18-2 *The DCF Decision Process*

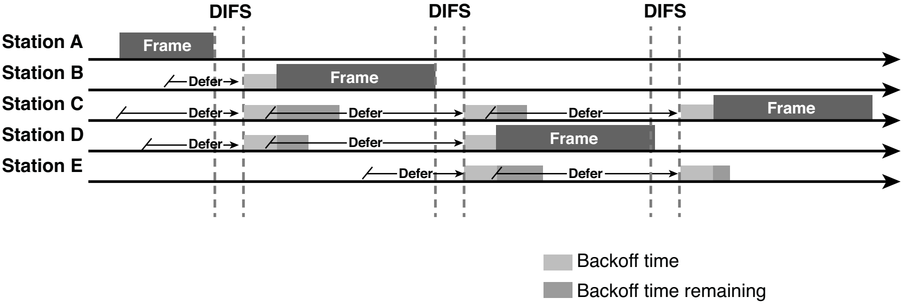


Figure 18-3 *DCF in Action*

Up to this point you have been introduced to the basics of the Distributed Coordination Function. Although DCF as it has been presented here does not have much to do with QoS, it forms the basic principles from which the QoS toolset was derived. This is discussed in the following section.

IEEE 802.11e and Wireless Multimedia (WMM)

Recommendation:

- Recognize the enhancements to DCF that 802.11e offers and how the WMM model achieves QoS over WLANs.

As you can understand from the previous section's analysis of the DCF model, there is no provision to differentiate traffic types, and therefore QoS is not possible using legacy DCF. The challenges faced in supporting QoS were indeed many. For example, how would QoS be added to a half-duplex environment? How would a shared multiple-access medium prioritize one traffic type over another? To address these challenges, the 802.11e task group provided enhancements to the original 802.11 specification that recommended several modifications to the way DCF operates that would facilitate a differentiated services model. These 802.11e modifications to the DCF model have been rolled into the wider 802.11-2012 standard, which is essentially a retrofit of the original 802.11 specification.

In addition to reworking DCF, 802.11e also proposes a method for call admission control (CAC) over the radio network, a power-save enhancement for clients, and an improved implementation of the Point Coordination Function, called Hybrid Controlled Channel Access (HCCA).

Note PCF is an alternative to DCF that was never widely adopted

The following section examines the QoS enhancements introduced by making some subtle but significant changes to the operation of DCF.

Retrofitting DCF: Enhanced Distributed Channel Access

The 802.11e task group has provided many enhancements to the overall 802.11 specification, but the key goal was to introduce an intelligent system of traffic queuing on the wireless radio interfaces. One of the most significant enhancements proposed by this task group was a rework of the way the MAC layer accesses the physical channel. As discussed in the previous section, DCF was the governing rule of the road for the MAC layer when 802.11 was first introduced. However,, the limitations imposed by this DCF model prevented any type of differentiated handling of various traffic types. To address this, the 802.11e task group developed a new MAC layer protocol, known as Enhanced Distributed Channel Access (EDCA).

EDCA introduces five major enhancements over DCF, which in turn enable a QoS toolset on Wi-Fi networks. These enhancements include the following:

- The establishment of four priority queues, or access categories (ACs)
- Different interframe spacing values for each AC, as opposed to a single fixed DIFS for all traffic
- Different contention window values for each AC (that is, different CW_{min} and CW_{max} values for each AC)
- Transmission Opportunity (TXOP)
- Call admission control (TSpec)

EDCA is similar to DCF in many ways, especially in the way that it uses a contention-based access model, meaning that each station is responsible for handling its own access to the channel. Although the 802.11e task group also proposed HCCA, which is a centrally coordinated access model where the AP tells each station when it has its turn to send, the market has by and large only adopted EDCA. Therefore, with EDCA, prioritization is accomplished by giving different traffic types varied access levels based on how long they must wait to transmit relative to the other traffic types.

A critical point to understand about wireless QoS, and EDCA in particular, is that unlike strict-priority queues in wired environments, which can guarantee the transmission of one traffic type over another, EDCA can only offer high-priority traffic a greater *probability* of being sent over low-priority traffic, but it can never guarantee it.

The five key enhancements of EDCA listed earlier are discussed in the following sections.

Access Categories

802.11 EDCA and WMM specify four different ACs, as listed in Table 18-1:

Table 18-1 *EDCA/WMM Access Categories Definitions*

Access Category Name	Description	Cisco WLC QoS Profile Name
AC_VO	Voice	Platinum
AC_VI	Video	Gold
AC_BE	Best effort traffic	Silver
AC_BK	Background traffic	Bronze

The following chapters focus much more heavily on Cisco wireless LAN controller (WLC) QoS design, but you still want to be aware of the naming convention used by the WLC for these four access categories. For reference, Table 18-1 also shows the corresponding precious metal naming convention that Cisco uses in the WLC 5500 series controllers (AireOS controllers) for each of these four ACs, where Platinum refers to the WMM Voice AC, Gold refers to the WMM Video AC, and so on.

To distinguish different classes of service, the 802.11e Ethernet frame header incorporates a 3-bit field known as the 802.11e User Priority (UP). The 802.11e UP is analogous to the 802.1p Class of Service (CoS) field used in wired 802.1Q Ethernet trunks; however, 802.11e UP values are used only between wireless stations.

Similar to the 802.1p CoS field used in a wired 802.1Q trunk connection, the 3 bits of the 802.11e UP field offer up to eight classes of service. The obvious question arises of how to correlate these eight classes of service to the four ACs that are available according to the 802.11e/WMM standard.

Table 18-2 demonstrates how the four wireless ACs map to their corresponding 802.11e/WMM UP values. For reference, this table also shows the corresponding name of these ACs that is used in the Cisco AireOS-based WLCs. Again, instead of using the normal WMM naming convention for the four ACs, Cisco uses a precious metals naming system, but a direct correlation exists to these four ACs.

Table 18-2 *EDCA/WMM Access Categories*

EDCA / WMM Access Category	Description	Corresponding 802.11e UP	Cisco WLC QoS Profile Name
Voice	Highest-priority voice and network control traffic	7, 6	Platinum
Video	Video, voice control traffic	5, 4	Gold
Best Effort	Legacy devices or applications that lack QoS capabilities	3, 0	Silver
Background	Lowest-priority traffic	2, 1	Bronze

As data is received by each radio interface, it is assigned to one of four egress queues that align with the four ACs (as shown in Table 18-2). Based on the 802.11e UP value (meaning the type of data that is being sent), the EDCF mechanism assigns the frame to the appropriate access categories. Once the frame is assigned to the appropriate AC, the radio interface attempts to send it over the medium according to the relative priority of the AC. In this way, the AC is really acting as a data queue, waiting to send its traffic onto the medium. The rules that govern the processing and transmission of the four ACs are discussed in the following sections of this chapter.

Figure 18-4 illustrates this process as a flow of frames is being queued up for transmission.

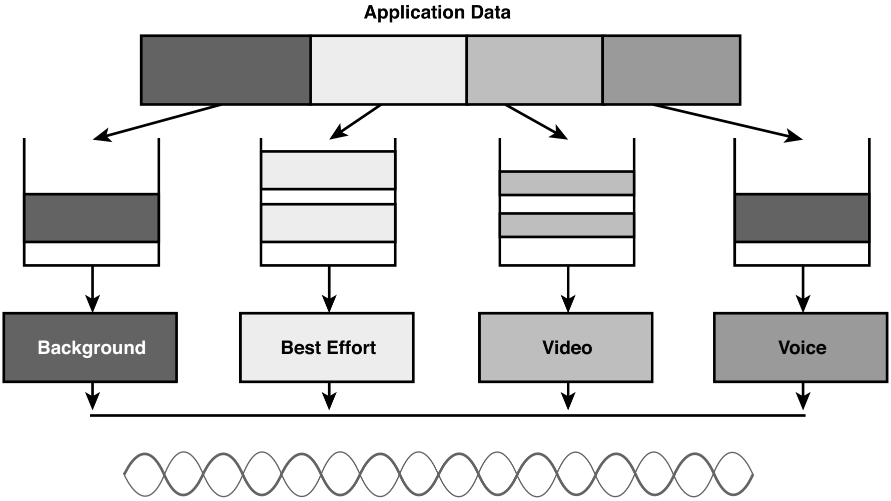


Figure 18-4 *Application Classes and Queuing*

One perplexing aspect of the 802.11e/WMM AC model shown here is that voice is mapped to CoS 6, rather than 5, which differs from the standard 802.1p convention used in wired networks. This difference often causes a significant amount of confusion for network engineers trying to design a consistent QoS model across a network, because the normal QoS marking scheme for voice differs in wired and wireless networks. Nonetheless, this is the standard by which 802.11e and WMM is defined, so it must be worked with.

Note The root cause of this discrepancy is that the IETF defines Layer 3 marking standards (differentiated service code point [DSCP]—specifically RFC 2474 and 4594), while the IEEE defines Layer 2 standards (like CoS and 802.11e UP). Unfortunately, this has led to some confusion and inconsistencies with the marking schemes that must be dealt with by the network engineer who works with both L2 and L3. This challenge is considered later in this chapter.

Arbitration Interframe Spacing

One of the key limitations of the legacy DCF model is that the DIFS value is the same for all traffic types. The rule to remember here is that once the channel is declared available, all stations wanting to send must wait the same DIFS time period following the end of the current station's transmission. The problem is that if multiple stations are waiting to send, they all have to wait the exact *same* DIFS gap, regardless of how latency sensitive their data is, thus giving no preferential treatment to either high- or low-priority traffic.

To address this, EDCA introduced a variable IFS period for data and management frames, called the arbitration interframe spacing number (AIFSN). The intention of assigning different IFS values to each AC is that the higher-priority ACs get a shorter IFS wait time compared to the lower-priority AC. This approach thus gives the high-priority traffic a much better probability of being sent first.

AIFSNs are usually configurable, but the default values defined in EDCA are as follows (measured in slot times).

Table 18-3 *EDCA/WMM Default AIFSNs*

AC Priority Queue	AIFSN Slot Times
Voice	2
Video	2
Best effort	3
Background	7

Clearly, with the voice and video queues having a much shorter AIFSN, you would expect these types of latency-sensitive data to always be sent first and therefore solve any QoS issues on a wireless network. It is true that assigning different IFS values to the different queues goes a long way toward improving QoS. However, it is important to remember that the wireless medium is still a multiple-access technology and that collisions are still possible. Therefore, what AIFS really accomplishes is that it greatly improves the overall *probability* that higher-priority traffic is serviced first by giving it a statistical advantage over lower-priority traffic. However, this approach still cannot guarantee that a collision will not occur.

Contention Window Enhancements

In legacy DCF environments, once the DIFS has expired, each station backs off for a random contention window period. Just like when using a common DIFS for all traffic types, DCF in general gives no preferential treatment to high-priority traffic during the CW, meaning that all traffic types have the same statistical probability of being the next one to transmit.

The final enhancement EDCA introduces is to give preferential CW random backoff ranges for higher-priority traffic, thus making it much more likely for a voice or video frame to be transmitted before a best effort or background frame. This is particularly important for latency sensitive traffic, such as voice and video, which suffer greatly if they have to wait up to the CW_{max} interval. Similar to the different AIFSN values assigned to the different priority queues, different contention window values serve to give higher priority traffic a better probability of having to wait a shorter period of time before having a chance to send, and also limit the impact of long CW wait times.

Table 18-4 lists the default EDCA CW values for 801.11a/g/n.

Table 18-4 *EDCA/WMM Default Contention Window Values*

	CW_{min} (Slot Times)	CW_{max} (Slot Times)
Legacy DCF CW values (for comparison)	15	1023
Voice	3	7
Video	7	15
Best effort	15	1023
Background	15	1023

Table 18-4 shows that voice only backs off between 3 and 7 slot times; in comparison, the background traffic backs off between 15 and 1023 slot times (which is still the same as the legacy DCF CW backoff period). As Table 18-4 shows, voice traffic will always generate a smaller CW than video traffic, and in turn video traffic will always generate a smaller CW than either best effort or background traffic. Therefore, using EDCA, the higher-priority queues are statistically serviced much more often than the lower-priority queues.

Putting this all together, Figure 18-5 demonstrates how improved IFS and CW backoff timers work together to improve the overall handling of high-priority traffic. In this example, the voice queue waits for five slot times before attempting to send its data onto the channel (two AIFS slots and a randomly generated CW of three slots), thus resulting in a significantly improved probability that the voice traffic is sent over the air before anything else.

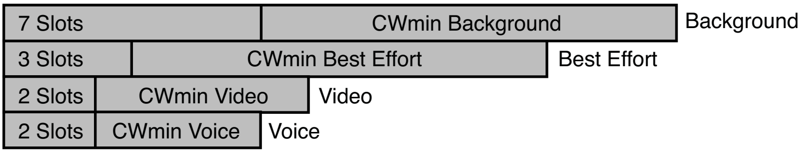


Figure 18-5 *AIFSN and Contention Window Sizing*

This example thus illustrates how the AIFSN and CW values work together to statistically improve the overall QoS handling of higher-priority traffic types by lowering the amount of wait time until the frames are sent over the air.

Transmission Opportunity

In addition to the three enhancements mentioned earlier, EDCA also provides contention-free access periods to the wireless medium, called the Transmission Opportunity (TOXP). The TXOP is a set period of time when a wireless station may send as many frames as possible without having to contend with the other stations. In the legacy DCF model, once a station has access to the medium, it can keep sending frames as long as it wants. It can be compared to a child who just got a toy and will not share it with anyone else until he is bored with it. When a low data-rate station gains access to the medium, it forces all other stations to wait until it finishes its transmission.

With EDCA's TXOP enhancement, each station has a set TXOP time limit during which it can transmit. When the TXOP limit expires, it must give up access to the medium.

802.11e TSpec: Call Admission Control

One last major enhancement introduced by 802.11e that is important to be aware of is the way in which the EDCA approaches the problem of maintaining QoS during times of high congestion. In traditional wired environments, this is handled through the use of a strict-priority queue on the egress interface. As examined in the previous sections, however, because of the 802.11's contention-based model this type of behavior is not possible in the wireless networks. Taking a closer look at the dynamics of how contention-based access works, you can see that the wireless medium can be susceptible to performance issues during periods of high congestion.

To illustrate, suppose that you have many stations contending for access to the network at the same time. As more traffic is being transmitted and more stations contend for access, more and more time is spent by all stations in the backoff state. The result is that performance for all stations is degraded.

This problem was addressed in 802.11e by a mechanism of call admission control (CAC) called Transmission Specification (TSpec). TSpec allows real-time applications, such as voice calls that are in progress, to be prioritized over requests for new calls. To use this feature of EDCA, TSpec must be configured on the AP and optionally on the client stations.

When running TSpec, which is highly recommended when deploying real-time applications, a client station signals its traffic requirements (mean data rate, power save mode, frame size, and so on) to the AP. In this way, before a client sends traffic of a certain priority type (which is managed by the appropriate AC), it must first request permission via the TSpec mechanism. For example, a WLAN client device wanting to use the voice AC must first make a request for use of that AC to see if there is sufficient space on the network to do so. If the AP decides that there is insufficient availability on the network, it denies access for that client station, thus protecting the currently sending stations.

QoS Design Considerations

Recommendations:

- Define upstream versus downstream QoS requirements.
- Realize the discrepancies between IEEE and IETF QoS models.
- Reconcile these differences with L2/L3 mapping:
  - Define an upstream QoS marking/mapping strategy.
  - Define an downstream QoS marking/mapping strategy.

When APs are configured to work with a centralized WLC, they are controlled through the Control and Provisioning of Wireless Access Points (CAPWAP) protocol, which allows the WLC to manage APs using UDP ports 5246 (control traffic) and 5247 (data). CAPWAP is defined by RFCs 5415 and 5416.

While the IEEE 802.11 standards body governing Wi-Fi networks is focused on the L2 aspects of this medium, the IETF RFCs mentioned earlier are focused on the L3 aspects of wireless communications. The marriage of the L2 and L3 standards for wireless has produced some interesting results, but the end result of this work has produced a coherent system for transporting L2 802.11 wireless frames over an IP backbone via CAPWAP. Therefore, CAPWAP essentially functions as an L3 tunneling mechanism through which the 802.11 wireless Ethernet frames are sent between the controller and the AP. (CAPWAP tunnels only exist between the AP and the controller, not the end-user wireless device.)

To maintain the proper QoS handling of IP packets, both over the WLAN and across the transport network, L3 QoS markings of the IP packets must be preserved end to end.

To accomplish this, a system of mapping DSCP and CoS values is required both over the CAPWAP tunnel and back to the wired infrastructure. Both the upstream and downstream QoS marking strategies are examined in the following sections.

Defining Upstream and Downstream Traffic Flow

At this point, it is important to clarify some terms that are used throughout the wireless QoS chapters:

- **Upstream:** Indicates the flow of packets from the mobile wireless device to the wired network infrastructure, including the following:
- **Radio upstream:** Refers to traffic sent by the WLAN clients and traveling to the AP. WMM provides upstream QoS for WLAN clients over the air. Each IP packet also has an internal DSCP marking that is used for prioritization over the IP network.

- **Network upstream:** Refers to traffic leaving the AP, traveling to the WLC. This traffic is encapsulated within CAPWAP. The DSCP value on the CAPWAP tunnel *may* differ from the inner packet's DSCP value. Wired campus QoS policies provision upstream QoS.

- **Downstream:** Refers to the flow of packets from the wired network infrastructure to the wireless devices, including the following:

- **Network downstream:** Refers to traffic leaving the WLC traveling to the AP. This traffic is encapsulated within CAPWAP. Wired campus QoS policies provision downstream QoS.

- **Radio downstream:** Refers to traffic leaving the AP and traveling to the WLAN clients. WMM provides downstream QoS for WLAN clients.

Figure 18-6 illustrates both the upstream and downstream wireless QoS nomenclature used throughout this section of the book.

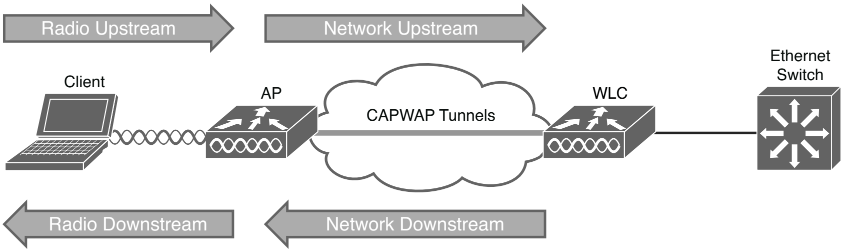


Figure 18-6 *Upstream and Downstream QoS*

QoS Mapping and Marking Considerations

When developing a QoS design strategy for a wireless network, it is important to consider how the wireless QoS model interoperates with the QoS design used in the wired network. One of the significant challenges faced by wireless network engineers is that the recommended IETF DSCP to IEEE 802.11e UP QoS mapping scheme differs significantly from what is used by the IETF DSCP to IEEE 802.1p CoS mapping model. In reality, it is the IEEE marking scheme that is out of sync with the IETF; however, the result is that applications such as voice, video, signaling, and others end up using an incompatible marking scheme.

To bring consistency to both wireless and wired networks, and to develop a QoS design that works end to end, careful consideration must be given to how the respective QoS markings are handled and mapped as packets travel between these two parts of the network.

Table 18-5 compares the DSCP mappings used by the Cisco WLC for various traffic types with the 802.11e UP and 802.1p CoS. For reference, the AireOS-based WLC QoS profiles that are used for these values are also shown.

Table 18-5 *DSCP to 802.11e UP and 802.1p CoS QoS Mapping Table*

Traffic Type	IETF DSCP	802.11e UP	802.1p CoS	WLC QoS Profile
Network control	56 (CS7)	7	7	Platinum
Internetwork control (CAPWAP, 802.11 management traffic)	48 (CS6)	7	6	Platinum
Voice	46 (EF)	6	5	Platinum
Multimedia conferencing	34 (AF41)	5	4	Gold
Multimedia streaming	26 (AF31)	4	3	Gold
Transactional data	18 (AF21)	3	2	Silver
Bulk data	10 (AF11)	2	1	Bronze
Best effort	0 (BE)	0	0	Silver

It is interesting to note some of the differences between the IEEE 802.11e UP values and the IEEE 802.1p CoS values shown in Table 18-5. For example, voice traffic, which is marked as DSCP 46, gets mapped to CoS 5 in wired networks, but is marked to an 802.11e UP value of 6 in wireless networks.

This is a key design consideration to be aware of so that inconsistencies are not encountered when deploying a pervasive QoS strategy. In an effort to reconcile these two mapping schemes, the WLC automatically converts the DSCP values to the values shown in this table, both in the upstream and downstream directions.

The IEEE 802.11e UP mappings for DSCP values that are not mentioned in Table 18-5 are calculated by considering the 3 MSB bits of the DSCP value. For example, the IEEE 802.11e UP value for DSCP 32 (100 000 in binary) would be the decimal equivalent of the MSB (100), which is 4. In this case, DSCP 32 is thus mapped to an 802.11e UP value of 4.

With the exceptions noted in Table 18-5, the DSCP to 802.11e UP markings are exactly the same as the standard 802.1p CoS values typically used in a wired network. This obviously can cause some confusion as the 802.1p CoS and 802.11e UP models seem to line up except for the well-defined exceptions in Table 18-5. To summarize, Table 18-6 illustrates the default DSCP to 802.1p CoS and 802.11e UP mappings used in the WLC. The fact that the WLC does these mapping automatically as packets come and go from the wireless network removes much of the complexity of having to deal with this yourself, but it also imposes some limitations, especially with AireOS-based controllers. With the newer IOS-based controllers, customization of this mapping table is possible, but it is not possible to manipulate these default mappings with AireOS-based controllers.

Table 18-6 *Default DSCP to 802.11e UP and 802.1p CoS Mappings (Excluding Exceptions Mentioned in Table 18-4)*

DSCP Range	IEEE 801.11e UP	IEEE 802.1p CoS	WLC QoS Profile
56–63	7	7	Platinum
48–55	6	6	
40–47	5	5	Gold
32–39	4	4	
24–31	3	3	Silver
0–7	0	0	
16–23	2	2	Bronze
8–15	1	1	

The Upstream QoS Marking Strategy

With the DSCP to 802.11e UP and 802.1p CoS mapping methods in place, the questions of where these translations happen and how they are done needs to be addressed.

The objective of upstream QoS is to ensure that all IP packets traveling from the wireless client, through the AP, into the CAPWAP tunnel, through the WLC, and finally into the LAN switch maintain their QoS markings and treatment. Although this appears to have many hops, at a high level, there are only two QoS remapping steps taking place:

1. The 802.11e UP marking on the upstream frame received by an AP is translated to a DSCP value on the outside of the CAPWAP tunnel. The inner DSCP making is preserved from the originating wireless device and does not change.
2. After the CAPWAP packet is decapsulated at the WLC, the original IP packet's DSCP value us used to derive the appropriate IEEE 802.1p CoS value and is sent toward the wired network switch.

Figure 18-7 illustrates how the QoS markings are preserved as an IP packet is sent from a wireless client all the way to the wired switch.

Now take a closer look at these two steps as a packet is transmitted upstream from a wireless client.

When IP packets are transmitted from the wireless client device, they are automatically marked by the application or the operating system with the appropriate DSCP value. Of course, some applications do not mark the DSCP value at all, and sometimes the application marks the DSCP value but is overruled by the OS. However, for the purpose of keeping things simple, assume that packets are marked with the correct DSCP on the wireless client.

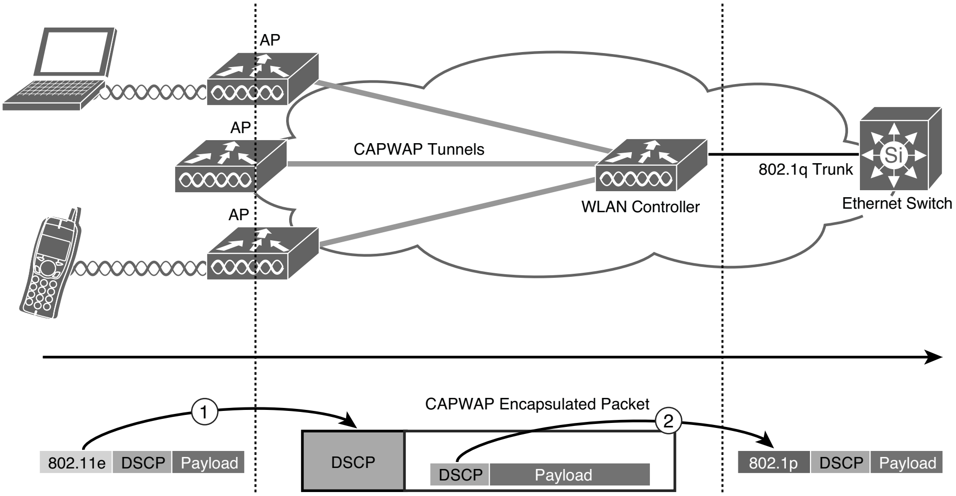


Figure 18-7 *Upstream QoS Mapping*

When the client device WMM radio interface classifies these packets, the DSCP value is internally mapped to the appropriate IEEE 802.11e UP value (as outlined in Table 18-5), and the 802.11 frame is then sent through the correct AC according to its 802.11e UP value.

Note If the client does not support WMM, there will be no 802.11e UP value marked into the frame (because 802.11e UP is only supported by WMM), and the AP will be forced to apply a default QoS setting to the traffic. This situation is described in more detail in the following chapter.

When the AP receives the 802.11 wireless frame, it takes note of the 802.11e UP marking and automatically maps it to the corresponding DSCP value on the IP header of the CAPWAP tunnel. For example, if the wireless frame is a voice packet with an 802.11e UP value of 6, the DSCP marking on the outside of the CAPWAP IP tunnel is automatically mapped to 46 (EF), which is consistent with the IETF marking scheme for voice RTP packets. The packet is then sent upstream over the IP network toward the WLC. It is imperative to understand that the upstream CAPWAP DSCP value is not derived directly from the inner packet's DSCP value. Rather, the CAPWAP packet's DSCP value (which is ultimately used for queuing across the IP backbone network) is only derived from the incoming upstream UP value.

From the underlying IP transport network's perspective, the CAPWAP tunnel is simply a flow of UDP packets with certain DSCP values that need to be handled with the appropriate levels of QoS based on the DSCP marking in the IP header. The IP transport network may be using the standard 4-, 8-, or 12-class model that is presented throughout this book. These UDP CAPWAP packets are thus handled like any other packet by the IP transport network.

When the WLC receives a wireless Ethernet frame sent through the CAPWAP tunnel, it examines the inner packet's DSCP value and automatically maps it to the appropriate 802.1p CoS value on the 802.1Q trunk port before sending it to the connected wired switch. For example, if the DSCP value is 46, a CoS value of 5 is automatically then written into the 802.1p header.

As you can see from this example, as each voice packet is sent across the network, the QoS markings start with a DSCP value of 46 and an 802.11e UP value of 6, are mapped to DSCP 46 on the outside header of the CAPWAP tunnel, and are finally mapped to an 802.1p CoS value of 5 on the wired port. That's a lot of facelifts for a single packet!

Note As a reminder, 802.11e UP values only apply to WMM clients. If the client does not support WMM, the QoS behavior will result in default treatment for that WLAN.

The Downstream QoS Marking Strategy

The process of downstream QoS is quite similar to the upstream model. As with the upstream model, the purpose of downstream QoS is to ensure that packets traveling from the wired LAN switch through the controller toward AP to the client maintain their QoS markings and treatment. As with the upstream direction, there are two QoS re-marking steps involved:

1. A frame with an 802.1p CoS marking arrives at the WLC wired-network-side interface. The DSCP of the inner IP packet is examined and is copied directly to the DSCP value of the CAPWAP packet header. (Note that the 802.1p CoS value plays no role in determining the DSCP value used on the CAPWAP header.)
2. When the AP receives the incoming CAPWAP packet, the DSCP value on the CAPWAP header is examined and mapped to a corresponding 802.11e UP value on the 802.11 wireless Ethernet frame and is then sent over the air.

Figure 18-8 illustrates the downstream QoS re-marking and mapping process.

Again, take a closer look at an example of how a packet's QoS markings are handled as it is sent from a wired LAN switch all the way down to a wireless client.

As the downstream Ethernet frames exit the wired LAN switch on the 802.1Q, trunk they are given an 802.1p CoS value that corresponds to the DSCP value of the IP packet (based on the three MSB of the DSCP value; see Table 18-4). When the trunked Ethernet frame arrives at the WLC, the controller separates the various VLANs from the 802.1Q trunk into the appropriate wireless service set identifier (SSID) networks and assigns the Ethernet frame to the correct wireless network. This complete, the WLC then uses the DSCP value from the original IP packet and copies it to the header of the CAPWAP tun-

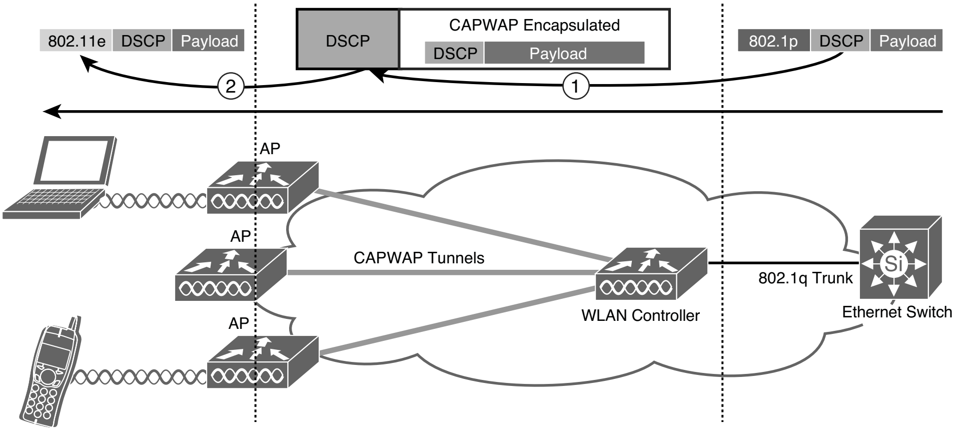


Figure 18-8 *Downstream QoS Mapping*

nel, thus ensuring proper QoS handling across the IP transport network. Note here that the 802.1p CoS value plays no role in determining the CAPWAP DSCP marking.

When the Ethernet frame arrives at the AP via the CAPWAP tunnel, the AP maps the CAPWAP IP DSCP header value (which is the same as the inner DSCP value) to the appropriate 802.11e UP value (see Table 18-4), and then sends the frame over the air to the client. In this way, the downstream QoS scenario is much simpler than the upstream model because only one intelligent mapping needs to be accomplished in the wireless network (on the AP), whereas two intelligent mappings are required in the upstream model (on the wireless client itself and on the AP).

Summary

This chapter began by examining some of the significant differences between wired and wireless networks that lead to significantly different approaches in solving QoS problems. The legacy protocols and building blocks that eventually made QoS possible in 802.11 WLANs were also discussed, including CSMA/CA and DCF. In the early days of 802.11, DCF provided for an equal-opportunity contention-based model where all wireless stations would have the same access to the wireless medium. Although this model laid the foundation for 802.11 WLANs, it had no concept of differentiated services or QoS.

This chapter then took a close look at some of the key QoS improvements proposed by the IEEE 802.11e task group. In particular, five key improvements over legacy DCF were examined, which are now known as EDCA:

- The establishment of access categories
- Variable interframe spacing based on traffic type
- Variable contention window sizing based on traffic type

- Transmission Opportunity,
- Call admission control with TSpec

The DSCP to 802.11e UP QoS marking scheme was examined in detail (and how it relates to the DSCP to 802.1p CoS marking). Certain inconsistencies between these two QoS marking schemes were discussed, as was the impact these may have on the end-to-end QoS design.

This was followed by a discussion of upstream and downstream QoS handling and how the various re-marking schemes work to provide comprehensive QoS services for wireless networks.

Additional Reading

IEEE 802.11 Standards Page: http://standards.ieee.org/about/get/802/802.11.html

Wi-Fi Alliance WMM Whitepaper: http://www.wi-fi.org/media/press-releases/backgrounder-wi-fi®-quality-service-features-enable-connected-world

IETF CAPWAP Page: http://datatracker.ietf.org/wg/capwap/charter/

Cisco WLC Default QoS Marking and Translation Matrix: http://www.cisco.com/en/US/partner/docs/wireless/controller/7.4/configuration/guides/consolidated/b_cg74_CONSOLIDATED_chapter_01010111.html#ID1534

Chapter 19

Centralized (Cisco 5500 Wireless LAN Controller) QoS Design

The subject of designing a wireless network to support voice and video using a Cisco centralized WLAN controller (WLC) can be relatively complex. For example, beyond the fundamentals of adding quality of service (QoS) to the WLAN, there is the question of the appropriate radio frequency (RF) requirements needed to support acceptable data rates and to reduce interference. There is also the necessity of enabling high-speed roaming between access points (APs) that must be met so that voice and video calls are not severely interrupted or even dropped when endpoints move around. Furthermore, there is the question of how to properly secure the real-time applications on the network.

Although all of these aforementioned topics are critical aspects of the overall WLAN design that must be considered carefully, and indeed they all contribute significantly to the quality of user experience (especially when using real-time applications), the primary focus of this chapter is QoS. Therefore, the assumption is made that you have taken the appropriate steps to have adequate RF coverage, roaming, and security designs in place before considering the QoS components.

This chapter begins with an examination of the big picture—that is, how end-to-end QoS is designed, including wired LAN integration. Next, the specifics of how the Cisco wireless LAN controller (WLC) 5500 series can be designed to support differentiated levels of services. From there, this chapter takes a closer look at some of the Cisco extensions to the QoS standards supported by the IEEE and Wi-Fi Multimedia (WMM), including special handling for voice and video stations that are running Session Initiation Protocol (SIP). Next, this chapter discusses how the application visibility control (AVC) toolset can be used to further enhance the QoS design of the WLAN. Finally, a comprehensive QoS model that supports the standard four-class and eight-class model reference designs will be discussed.

Note that this chapter focuses on centralized wireless QoS designs using the WLC 5500 series. Although other wireless architectures are supported by Cisco, such as Autonomous Mode and FlexConnect (previously known as Hybrid Remote Edge Access Point, or H-REAP), these are not the focus of this chapter. The following two chapters (Chapters 20 and 21) focus on wireless QoS using the Catalyst 3850 series and the WLC 5760.

QoS Enforcement Points in the WLAN

Design of QoS in the WLAN must be looked at holistically (that is, end to end). Both the wireless and wired portions of the network must support QoS in a consistent manner (especially the QoS marking scheme) such that applications receive the same level of service no matter what part of the network they reside in.

QoS in the WLAN is thus enforced in four places:

1. **The network switch connected to the WLC:** The network switch is usually configured as an IEEE 802.1Q trunk with an 802.1p class of service (CoS) value associated. The wired network switch is also responsible for queuing downstream traffic toward the WLC according to the network QoS policy.
2. **The WLC itself:** The WLC is capable of assigning various QoS policies on a per-service set identifier (SSID) or per-user basis (these are actually implemented by the AP, not the WLC). In the downstream direction, the WLC marks the differentiated services code point (DSCP) values on the Control and Provisioning of Wireless Access Points (CAPWAP) tunnel, and in the upstream direction, the WLC marks the 802.1p CoS values on the Ethernet trunk. The WLC is also capable of re-marking the DSCP values on a per-application basis using the AVC feature set.
3. **The access point:** The AP is the ultimate enforcement point for wireless QoS on the WLAN. It has no ability to intelligently re-mark traffic, but in the downstream direction it maps the CAPWAP DSCP value to the correct 802.11e User Priority (UP) value and then classifies it into the correct WMM access category (AC). In the upstream direction, the AP is responsible for mapping the 802.11e UP value to the corresponding DSCP value used on the CAPWAP tunnel.
4. **The wireless client:** The clients need to have the capability at the application layer to correctly mark the DSCP values of packets that are being sent. Furthermore, if the client supports modern WMM capabilities, it can take advantage of several QoS features, such as marking the correct UP values which ensure proper access to the wireless medium. In addition to setting the UP value on the 802.11 frame, WMM clients are able to take advantage of other key features including Traffic Specification (TSpec) and Transmission Opportunity (TXOP).

Note UP, WMM, TSPEC, and TXOP are discussed in more detail in Chapter 18, "Wireless LAN QoS Considerations and Recommendations."

Of the four QoS enforcement places mentioned in the preceding list, three of them are configurable by the network engineer: the network switch, the WLC, and the client. Although most QoS enforcement actually happens on the AP, from the WLC's perspective it is just a drone—an extension of the WLC itself. Figure 19-1 illustrates these four QoS enforcement points across both the wired and wireless networks.

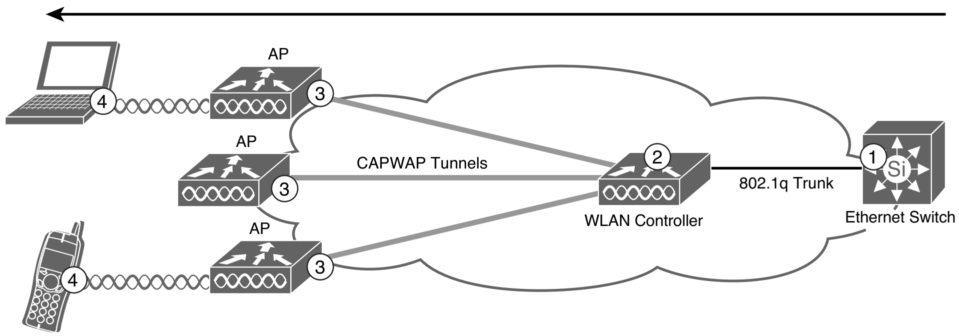


Figure 19-1 *QoS Enforcement Points in the Wired and Wireless Networks*

Managing QoS Profiles in the Wireless LAN Controller

As a best practice, if possible, it is recommended to configure a separate WLAN for each class of service used in the wireless network; for example, it is recommended to deploy mobile Wi-Fi IP phones on a "voice WLAN/SSID." For engineers working in the wired side of the network, this is a familiar concept; in this case, IP phones are given a dedicated voice VLAN, which is seen primarily as a way to address security concerns. Although security plays a key role in isolating wireless handsets to the voice SSID, the other compelling driver is that it allows a single QoS policy to be applied to the entire SSID.

QoS Marking and Conditional Trust Boundaries

You will have noted by now that the QoS marking and trust boundary design is one of the first things considered in each design chapter of this book. The Cisco 5500 series wireless LAN controller, which runs an operating system called AireOS, presents a different challenge in this regard. In short, aside from the AVC feature set (considered later in this chapter, in the section, "Application Visibility Control in the WLC"), the 5500 series WLC has only a limited capability to implement QoS trust boundaries.

The implication of this is that when designing wireless networks with the 5500 series WLC, the trust boundary must be placed either at the wired edge that connects to the WLC, or the wireless clients must be implicitly trusted. For example, if a wireless IP phone is marking packets as Expedited Forwarding (EF), this is always implicitly trusted until it reaches the switch connected to the WLC. Compared to the traditional trust model used in the wired world, this is certainly a major paradigm shift.

To solve this problem, dedicated WLANs are often assigned to real-time applications. Using this method, the real-time applications operate only in these WLANs; for example there may be a voice WLAN and another one for video. This in itself does not provide a trust boundary; rather, to be trusted and gain access to a particular real-time WLAN, a wireless client must pass the authentication phase, and thus any and all packets coming from this client are implicitly trusted.

Note In the WLC, SSIDs are referred to as WLANs, which in turn usually have a direct one-to-one mapping to an Ethernet VLAN on the wired side (although role-based VLAN assignment from a common SSID is also supported through 802.1x).

WLAN QoS Profiles

As a best practice, if possible, it is recommended to configure a separate WLAN for each class of service used in the wireless network. For engineers working in the wired side of the network this is a familiar concept; for example, it is recommended to deploy mobile Wi-Fi IP phones on a “voice WLAN/SSID.” In the case of the wired network, allocating IP phones to a dedicated voice VLAN is seen primarily as a way to address security concerns. Although security plays a key role in isolating wireless handsets to the voice SSID, the other compelling driver is that it allows a single QoS policy to be applied to the entire SSID.

In many ways, assigning real-time applications to a dedicated SSID makes the job of designing QoS for voice and video relatively simple because real-time traffic is not mixed with any other type of best-effort traffic and because the QoS policy handling it can be applied to the whole WLAN. That being said, any packets received by the AP from a client are expected to have the correct UP and DSCP markings set. If they do not, even if they are in a dedicated SSID that is given the highest level of QoS, it still treats these packets according to their markings. For example, if a voice packet is received on the voice WLAN but has a DSCP value of zero, it is treated as best effort, even though the voice WLAN is configured with the “Platinum” level of QoS. As was stated earlier, the WLAN does not set a trust boundary and does not influence the DSCP markings of the packets after they are received (with the exceptional case of when AVC is used).

Although segregating voice, video, and data by class into different SSIDs is becoming increasingly difficult these days (such as when using a softphone on a laptop or a tablet PC that has multimedia applications), where possible this simplified model allows the WLC to directly map a single QoS policy to a whole WLAN.

Although the design guidance to create a separate SSID per device type is somewhat restrictive, and may not suit many modern applications that mix voice and data (such as smart phones, tablets, and so on), it does provide a simple approach to deploying QoS in the WLAN. The first step in deploying QoS in the WLAN is to plan which applications will use each of the four QoS profiles that are available.

The 5500 series WLC supports four QoS policies, which have a direct correlation to the four 802.11e/WMM ACs. Cisco uses a precious metal naming scheme to refer to the four WMM ACs as defined in Table 19-1.

Table 19-1 *The Four QoS Profiles in the 5508 WLC*

QoS Profile Name	Application Assignment	Note
Platinum	Voice	Ensures the highest level of QoS handling for an SSID.
Gold	Video	This setting supports high-quality video applications.
Silver	Best effort	This is the default setting for all WLANs. It supports normal bandwidth for clients.
Bronze	Background	This setting provides the lowest bandwidth and is often used for guest services.

Although only four QoS profiles are available in the system, it does not mean that you are restricted to configuring only four SSIDs. The WLC allows QoS profiles to be applied to multiple SSIDs. For example, if you were to create two internal voice SSIDs, they may both have the same Platinum QoS profile attached to them, even though the WLANs are kept separate within the system itself.

Although some minor tweaks can be made to these QoS profiles, for the most part they are meant to have a direct mapping to the four Enhanced Distributed Channel Access (EDCA) access categories defined by WMM and IEEE 802.11e.

So how do these QoS profiles in the WLC actually work? In the downstream direction, QoS is handled by mapping the incoming packet's DSCP value to the CAPWAP DSCP (as illustrated in Figure 18-8 in Chapter 18). At the AP, the DSCP on the CAPWAP tunnel header is mapped to the corresponding 802.11e UP value and is placed in the correct egress AC. As packets enter the WLC, the DSCP makings are compared with the QoS profile applied to the WLAN that the packets are a part of. If the DSCP value exceeds the QoS profile's maximum allowable DSCP value, it is downgraded to the maximum. For example, if a WLAN has the Gold profile applied, the maximum DSCP allowed is 34. Therefore, if a packet enters the WLC from the wired side with a DSCP of 46, the WLC will write a DSCP 34 onto the packet header of the CAPWAP tunnel instead of allowing the value of 46 to be used. To illustrate, the Silver QoS profile has a maximum allowable DSCP value of zero. Thus, if the QoS profile is set to Silver, every CAPWAP packet header is marked with a DSCP value of 0 on the CAPWAP header.

Note Remember that these mappings in the AP and the controller never impact the inner IP packet's DSCP value. The DSCP markings referred to here are only applied to the CAPWAP packet header. The only exception to this rule is the AVC feature set which has the ability to rewrite the original IP packet's DSCP value.

Table 19-2 outlines the default maximum DSCPs for each WLC QoS profile.

Table 19-2 *Default WLC QoS Profile QoS Values*

WLC QoS Profile	Default DSCP Mapping on to the CAPWAP Header
Platinum	EF (46)
Gold	AF41 (34)
Silver	DF (0)
Bronze	AF11 (10)

If a packet enters the WLC from the wired side with a DSCP lower than the default maximum of the QoS profile on that WLAN, the original packet's DSCP value is simply copied to the CAPWAP header and is in turn used to map to the 802.11e UP value at the AP.

In the upstream direction, the AP behaves in a very similar way to how the WLC acts in the downstream direction. Because the 802.11e UP values are set by the client, the AP needs to compare this with the maximum allowed UP value for the QoS profile on that WLAN so that the correct DSCP value will be mapped into the CAPWAP packet.

Note The 802.11e UP to DSCP mapping was discussed in Chapter 18 and is detailed in Table 18-5.

Again, the QoS profile sets a maximum allowable 802.11e UP value for each WLAN. Although the WLC sets a default for each of the four profiles, it is possible to customize the maximum allowable 802.11e UP value for each profile. For example, while the default DSCP value for Silver is 0, it can be set to another value if you wish. The DSCP value in turn is automatically mapped to an UP value at the AP before it is sent over the radio interface.

Consider the following scenario. If the QoS profile is set to Gold, then by default a maximum UP value of 5 is allowed on that SSID. If a wireless frame with an 11e UP value of 6 is received, the AP will automatically map the UP value to a DSCP value of 34 (34 is the corresponding DSCP value to UP 5) on the upstream CAPWAP packet—essentially downgrading the QoS handling of that packet across the IP transport network. By way of contrast, if the WLAN receives any lower 11e UP values (1–5), these will just be mapped to the corresponding DSCP value shown in Table 18-5, thus leaving them unchanged. Although this does not really constitute as a trust model, it does allow the AP and WLC to establish a ceiling on the maximum QoS levels that are accepted per WLAN.

If the wireless client does not support WMM, meaning that there is no 802.11e UP present in the 802.11 frame, the default 802.11e UP value for that WLAN is used in the upstream mapping (see Table 19-2). For example, if a non-WMM client sends an 802.11

frame upstream to an AP in the Platinum profile, the AP will automatically write 46 into the upstream CAPWAP header simply because the upper DSCP limit of the Platinum profile is 46. For WMM clients, it is the UP value that is used to map to the DSCP value on the upstream CAPWAP tunnel. Because there are no UP values on non-WMM clients, the QoS capabilities of these clients is thus quite limited.

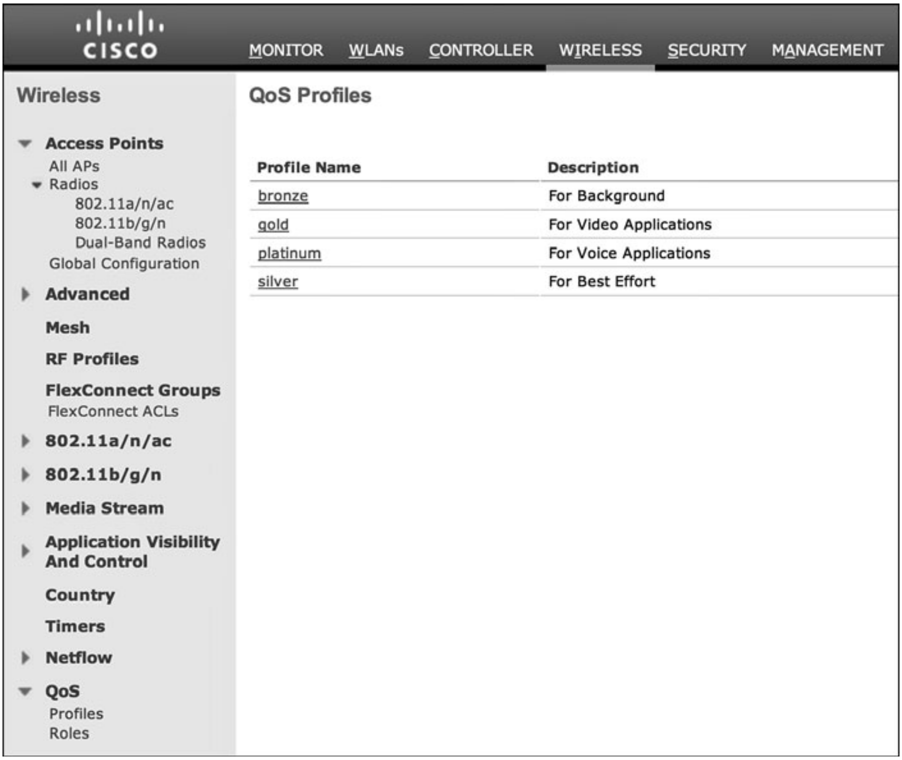
Consider another example where the QoS policy can be used to mark traffic down to best effort. This might be the case with a guest wireless SSID where no traffic is trusted and must be re-marked to a DSCP value of 0. Although you cannot control the DSCP and 802.11e UP markings that originate on the client device, the AP can use the Silver QoS profile, which has a default maximum DSCP value of 0. The Silver QoS profile by default maps all incoming UP values to a DSCP value of 0 on the upstream CAPWAP packets, regardless of what the originating DSCP or UP value was. Of course, the AP does not have the ability to re-mark the actual inner DSCP value to 0; rather, it is only the CAPWAP DSCP value that is manipulated. This means that while the original DSCP values are still present as the packets leave the WLC in the upstream direction toward the wired network, at least the traffic in this WLAN will be handled as best effort as it transits the IP backbone between the AP and the WLC.

It is important to note that the mappings shown in Tables 18-5 and 19-2 are not customizable. This means that certain situations may arise where the AP maps the UP value to a DSCP that is not aligned with the QoS policy in the campus network, thus affecting the handling of CAPWAP packets as they are transported across the IP backbone. Note that in the newer IOS-based 5760 controller, it is possible to customize this mapping table, thus providing much more granular control.

To configure the QoS policies in the WLC 550 series, navigate to Wireless, then to QoS, and then click **QoS Profiles**. Figure 19-2 shows how these four QoS profiles are presented in the WLC graphical user interface (GUI).

Note Although the 5508 WLC does support a command-line interface (CLI) through AireOS, the GUI interface is the common and recommended way to configure this controller. All configurations in this chapter utilize the WLC GUI. Chapter 20, “Converged Access (Cisco Catalyst 3850 and the CT 5760) Wireless LAN Controller QoS Design,” examines the 5760 controller, which is based on IOS and gives a much greater degree of freedom when configuring the WLC from CLI.

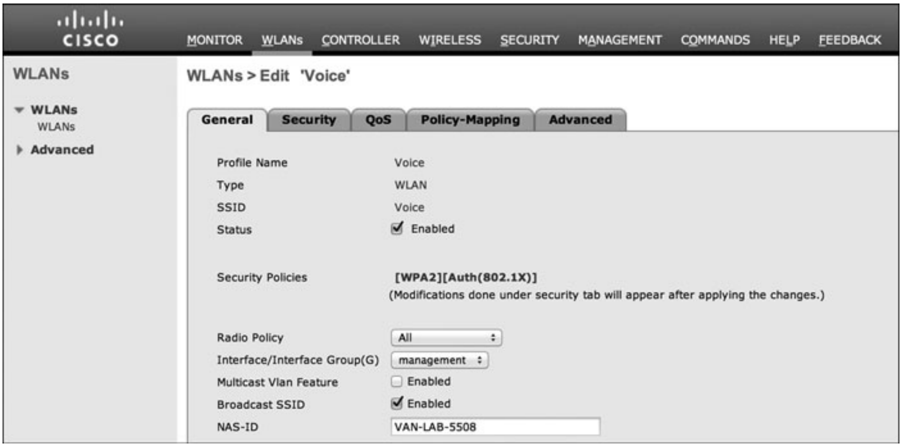
Because these QoS profiles must be applied to an entire SSID, the first configuration step is to create a WLAN SSID for each of these application types. First navigate to WLANs on the top tab, and then follow the GUI to create a new WLAN. Figure 19-3 illustrates how to configure a dedicated WLAN SSID for VoIP.



QoS Profiles

Profile Name	Description
bronze	For Background
gold	For Video Applications
platinum	For Voice Applications
silver	For Best Effort

Figure 19-2 *The Four WLAN QoS Profiles*



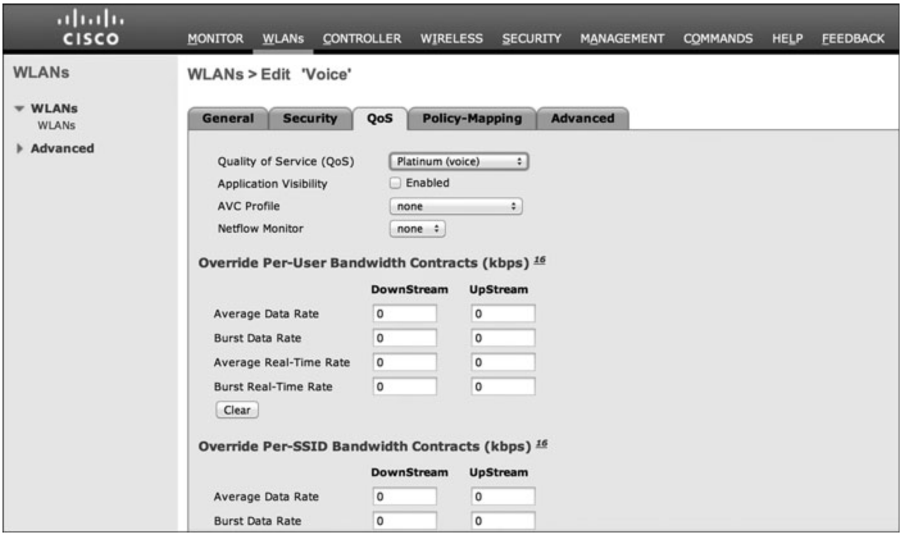
WLANs > Edit 'Voice'

Profile Name	Voice
Type	WLAN
SSID	Voice
Status	☑ Enabled
Security Policies	[WPA2][Auth(802.1X)] (Modifications done under security tab will appear after applying the changes.)
Radio Policy	All
Interface/Interface Group(G)	management
Multicast Vlan Feature	□ Enabled
Broadcast SSID	☑ Enabled
NAS-ID	VAN-LAB-5508

Figure 19-3 *Configuring a Dedicated WLAN for Voice*

In Figure 19-3, the **voice** WLAN has been created and enabled. The Radio Policy has been set to **All**, meaning both 802.11b/g/n and 802.11a/n clients can associate. It is usually better to keep voice stations on the 802.11a/n radio (5.8-GHz band) rather than the 802.11b/g/n (2.4-GHz band) because of the much higher channel availability, thus giving clients better access to the network.

After the WLAN has been created for this application class, it is now an easy matter to apply the correct QoS profile to the SSID. Figure 19-4 illustrates how the Platinum QoS policy is applied to the voice WLAN.



WLANs > Edit 'Voice'

Quality of Service (QoS) Platinum (voice)

Application Visibility □ Enabled

AVC Profile none

Netflow Monitor none

Override Per-User Bandwidth Contracts (kbps) [16]

	DownStream	UpStream
Average Data Rate	0	0
Burst Data Rate	0	0
Average Real-Time Rate	0	0
Burst Real-Time Rate	0	0

Override Per-SSID Bandwidth Contracts (kbps) [16]

	DownStream	UpStream
Average Data Rate	0	0
Burst Data Rate	0	0

Figure 19-4 *Applying the Platinum QoS Policy to the Voice WLAN*

As you can see from Figure 19-4, the **Platinum** QoS profile is applied to the voice WLAN. This page also lets you set the per-user bandwidth contracts for each user who is associated to this WLAN. Typically the bandwidth contracts are established in the QoS profile configuration section, but this page gives you the ability to override the default contracts that are configured for this QoS profile (Platinum in this case).

Further down on this same page in the WLC GUI (not shown in the example in Figure 19-4, but scroll down to see it), several options are available to enforce WMM policy on the QoS profile. For example, if the WMM Policy is set to **Allowed**, this means that both WMM and non-WMM capable clients are allowed on this WLAN.

Note that the default QoS profile for all new WLANs is **Silver**. This means that unless you manually configure the QoS profile, all traffic on the WLAN will be treated as best effort—meaning all CAPWAP packets are assigned a DSCP of 0, and correspondingly, all downstream 802.11e frames are assigned an UP value of 0.

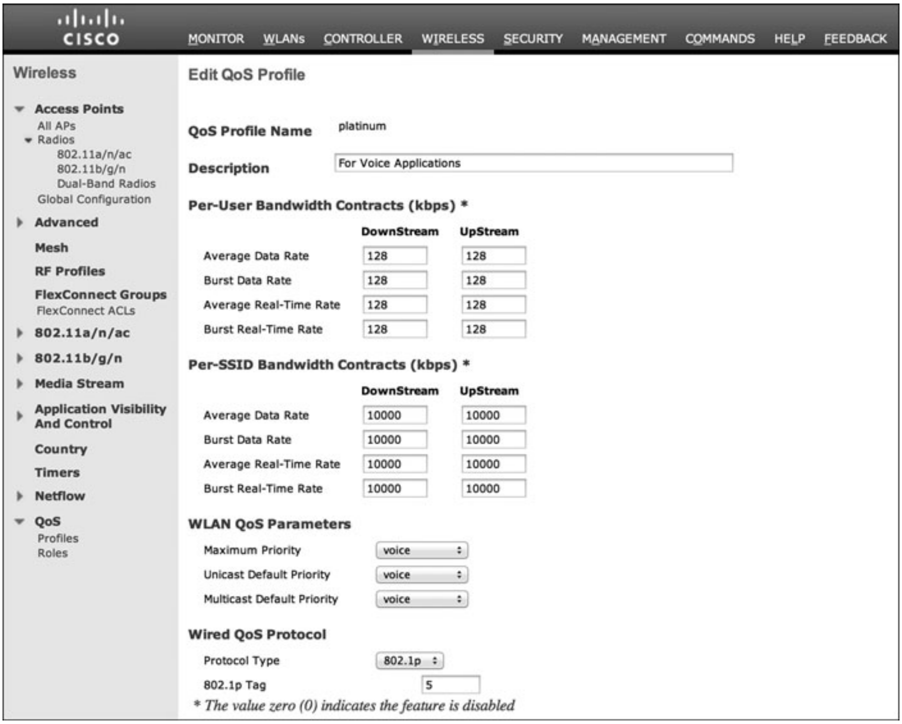
There are three options for configuring the WMM policy:

- **Disabled:** Turns off WMM.

- **Allowed:** This is default setting. If the WMM policy is set to **Allowed**, it means the WLAN will support WMM QoS *if* requested by a WMM client. If a client does not explicitly request WMM support, it will not be enabled.
- **Required:** The option requires that all clients must support WMM on this WLAN.

Remember that WMM support is tied to a client's capability to use the 802.11e UP QoS marking. If WMM is not supported on the client, or if WMM is disabled on the QoS profile, all QoS markings are set to the default, as shown in Table 19-2.

The configurable attributes of the four "precious metal" QoS policies can be found by clicking **Wireless**, then **QoS**, and then **Profiles**. Figure 19-5 illustrates the configurable attributes.



Edit QoS Profile

QoS Profile Name platinum

Description For Voice Applications

Per-User Bandwidth Contracts (kbps) *

	DownStream	UpStream
Average Data Rate	128	128
Burst Data Rate	128	128
Average Real-Time Rate	128	128
Burst Real-Time Rate	128	128

Per-SSID Bandwidth Contracts (kbps) *

	DownStream	UpStream
Average Data Rate	10000	10000
Burst Data Rate	10000	10000
Average Real-Time Rate	10000	10000
Burst Real-Time Rate	10000	10000

WLAN QoS Parameters

Maximum Priority voice

Unicast Default Priority voice

Multicast Default Priority voice

Wired QoS Protocol

Protocol Type 802.1p

802.1p Tag 5

* *The value zero (0) indicates the feature is disabled*

Figure 19-5 *Editing the QoS Policies*

As shown in Figure 19-5, you can configure several attributes of the QoS profile for each category. For example, this screen allows you to establish per-user bandwidth controls both in the upstream and downstream directions. For example, on the voice WLAN, the per-user bandwidth contract can be set to the expected bandwidth used for a G.711 codec. Note that although G.711 can be calculated to consume 87.2 Kbps per call, as a best practice it is recommended to set the per-user bandwidth to no smaller than 128 Kbps.

Similarly, for multipurpose WLANs, it might be desirable to establish a baseline bandwidth for each user in the system so that no one user will consume all the available bandwidth and thus starve the other associated clients.

Another useful option here is that you can configure per-SSID bandwidth contracts, meaning the WLC can enforce a bandwidth rate limiter for the entire WLAN that each profile is attached to, in both the upstream and downstream directions. This is particularly useful if you are using many SSIDs on a single wireless network where they are all competing for available bandwidth.

Although the design guidance around per-user and per-SSID bandwidth controls varies depending on the situation, it should be considered in places where there is a heavy concentration of users (such as in a university campus network), or in places where the AP's wired connection is bandwidth limited, such as at a remote site.

In each of the Per-User and Per-SSID bandwidth contracts fields, you have an option to configure both the average data rate and the average real-time data rate. Here the average real-time rate applies to all UDP traffic. This proves particularly useful if your SSID uses a mix of applications and is not reserved strictly for real-time applications.

One of the interesting attributes of the QoS profile is the WLAN QoS Parameters, as shown in Figure 19-5. As mentioned previously, one of the main functions of the QoS profile is to restrict DSCP and 802.11e UP values to a certain maximum level. In this part of the configuration, you can set the maximum and default QoS priorities of the profile. For example, in the Silver profile, it is possible to change the default from **best effort** to **video** if you want to (not that this is a recommended thing to do). The Maximum Priority field shown in Figure 19-5 controls the maximum UP value accepted from a WMM-enabled client. The unicast default priority controls the upper value of CAPWAP DSCP marking coming from non-WMM clients.

Compare how this changes the QoS behavior of the profile. When Silver is left at the default of **best effort**, the WLC and AP ensure that all QoS markings are set to 0, both in the upstream and downstream directions. This means that any upstream UP value is always mapped to DSCP 0, without exception. If this value is changed to **video**, it means the profile allows all incoming UP values up to 5 without modification, and in turn a DSCP value of 34. However, if an UP value of 6 is received, the QoS profile still downgrades it to 5, which is the maximum for the video profile.

Another important aspect of the QoS profile configuration is the mapping of the WLAN QoS markings to the 802.1p CoS value on the wired side of the network. As mentioned previously, the IEEE and WMM standard for QoS marking in wireless environments does not exactly line up with the standard approach used in wired networks. For example, in a wired network, voice applications are always mapped to a CoS value of 5 (with a corresponding DSCP value of 46). However, the IEEE standard approach in WLAN environments is to use an 802.11e UP value of 6 for voice instead (still DSCP 46). Therefore, it is important to correctly map the CoS values for each WLAN on the wired side. To make things convenient, by default the WLC automatically maps the inner packet's DSCP value to the correct 802.1p CoS value (see Table 18-5 in Chapter 18); however, if you want to modify the CoS mapping as frames are sent from the controller toward the wired side of the network, this can also be done from the same screen shown in Figure 19-5.

Table 19-3 describes some of the key fields in this screen where the QoS profile is edited.

Table 19-3 *Editing the QoS Profile Fields Description*

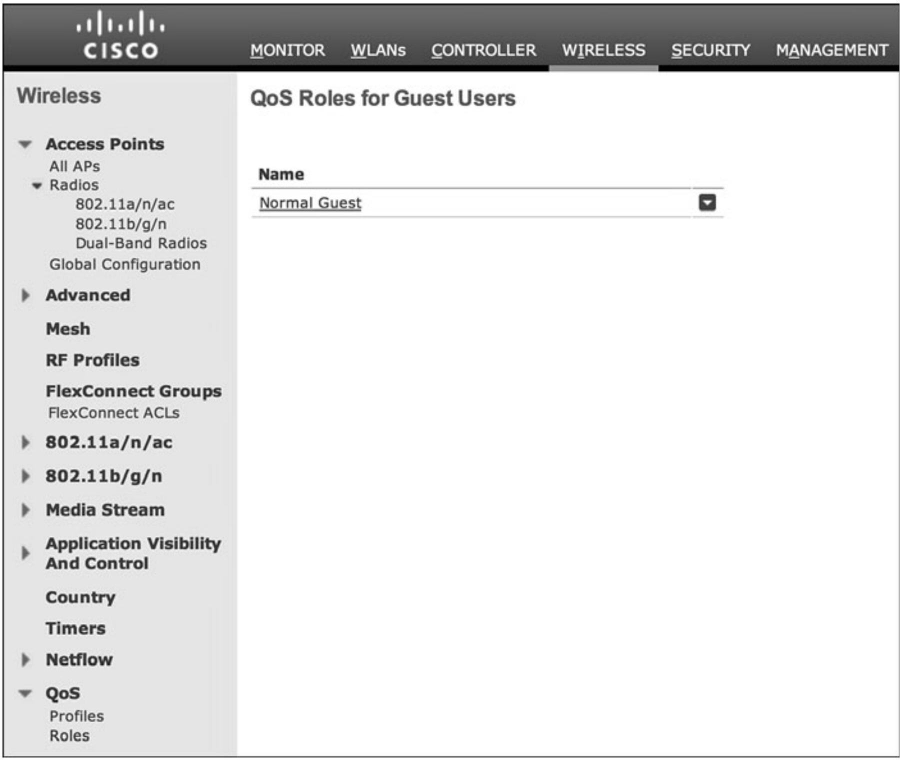
Feature Name	Description
Description	Operator-defined description for the QoS profile.
Average Data Rate	0 to 60,000 Kbps bits per second. Operator-defined average data rate for non-UDP traffic. Default of 0 = OFF.
Burst Data Rate	0 to 60,000 Kbps bits per second. Operator-defined peak data rate for non-UDP traffic. Default of 0 = OFF.
Average Real-Time Rate	0 to 60,000 Kbps bits per second. Operator-defined average data rate for UDP traffic. Default of 0 = OFF.
Burst Real-Time Rate	0 to 60,000 Kbps bits per second. Operator-defined peak data rate for UDP traffic. Default of 0 = OFF.
Wired QoS Protocol	Select 802.1P to activate 802.1p priority tags, or select None to deactivate 802.1p priority tags (default).
802.1P Tag	802.1p priority tag for the wired connection from 0 to 7. *This tag is used for wireless traffic and is used to map the DSCP value used by CAPWAP packets.* By name, 1 for Bronze, 3 for Silver, 4 for Gold, and 6 for Platinum.

Building a Guest QoS Profile

The four basic QoS profiles discussed in the previous section can be aligned to the classic four-class QoS model. In turn, certain applications, such as voice and video, can be mapped to the four different profiles presented. However, wireless networks often present a unique vector that is rarely seen in wired networks—support for guest users.

As wireless networks have become more and more pervasive, people have begun to expect Wi-Fi access wherever they go. Whether you go to a coffee shop, a fast-food outlet, a museum, a subway station, or visit as a contractor to another company, the Wi-Fi guest network has become so universal that if a company doesn't offer guest wireless access, it has the potential to negatively affect business. The question then comes up: Which QoS profile should be used for guest users?

The 5500 series WLC allows you to create custom roles that allow a unique QoS policy to be applied to a specific group of guest users. The WLC software allows you to create up to ten guest user roles, which can be applied to different guest groups. To create or configure QoS Roles for Guest Users, navigate to **Wireless > QoS > Roles**. Figure 19-6 demonstrates how to add a new guest user QoS profile.



QoS Roles for Guest Users

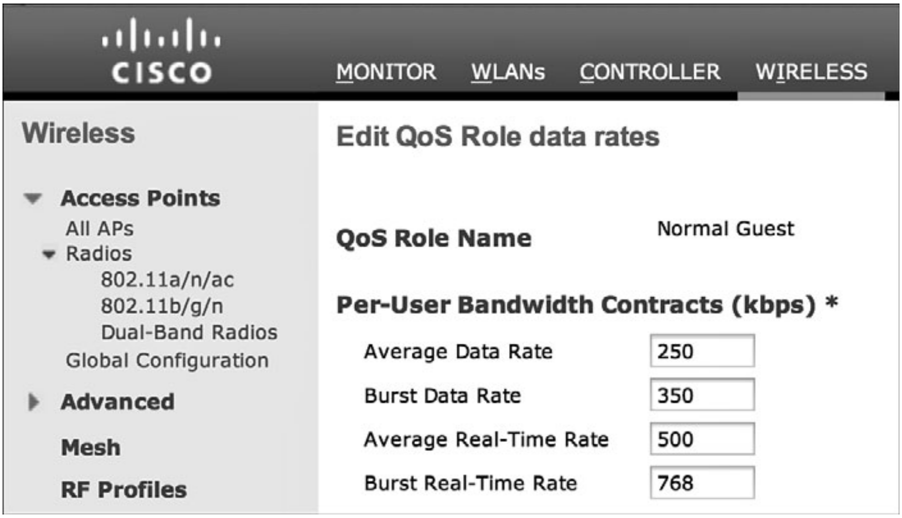
Name
Normal Guest

Figure 19-6 *Adding a Guest User QoS Profile*

Once the roles are created, select the role name to edit the role and to set the per-user bandwidth contracts. This approach helps to ensure that each guest user is restricted in the amount of bandwidth that they can use while associated with your wireless network. Although it is not necessary to even provide a bandwidth contract, it is recommended that some contract be given so that guest users do not consume valuable bandwidth required for corporate users. Figure 19-7 illustrates this step.

The guest users themselves are either configured locally on the controller or can be linked with an external authentication, authorization, and accounting (AAA) server. However, a detailed discussion of how to create users within this role is beyond the scope of this book.

To define the average data rate for TCP traffic on a per-user basis, enter a rate between 0 and 60,000 in Kbps in the Average Data Rate field. A value of 0 imposes no bandwidth restriction on the QoS role. For example, if you wanted to rate limit every client in the guest profile to 3 Mbps, configure a value of **3000** here.



Edit QoS Role data rates

QoS Role Name: Normal Guest

Per-User Bandwidth Contracts (kbps) *

Average Data Rate	250
Burst Data Rate	350
Average Real-Time Rate	500
Burst Real-Time Rate	768

Figure 19-7 *Creating the Guest User QoS Profile*

To define the peak data rate for TCP traffic on a per-user basis, enter a value between 0 and 60,000 Kbps in the Burst Data Rate field. As before, a value of 0 imposes no bandwidth restriction on the QoS role. The Burst Data Rate should be greater than or equal to the Average Data Rate. If this is not done correctly, the QoS policy may actually block traffic to and from the wireless client.

To define the Average Real-Time rate for UDP traffic on a per-user basis, enter the rate in kilobits per second in the Average Real-Time Rate field. This refers to voice and video traffic, so it is configured separately as the non-real-time traffic. As before, enter a value between 0 and 60,000 Kbps, where a value of 0 imposes no bandwidth restriction on the QoS role.

To define the peak real-time rate for UDP traffic on a per-user basis, enter the rate in kilobits per second in the Burst Real-Time Rate field. Enter a value between 0 and 60,000 Kbps (inclusive). A value of 0 imposes no bandwidth restriction on the QoS role.

Note The values you configure for the per-user bandwidth contract only affect the amount of bandwidth going downstream (from the AP to the wireless client). The values do not affect the bandwidth for upstream traffic (from the client to the AP). Interestingly, the WLC 5500 does not allow upstream rate limiting contracts for guest users. Yet, on an SSID-wide basis for nonguest users, the WLC can do per-user bidirectional rate limiting.

QoS Design for VoIP Applications

Once the QoS profiles have been configured and applied to the correct WLAN SSIDs, you can consider modifying some options that help optimize QoS on the WLANs that carry real-time traffic.

The areas of optimization can be summarized into two groups:

- EDCA optimization
- Call admission control (CAC)

These are both discussed in turn.

Tweaking the EDCA Configuration

As was discussed at length in Chapter 18, Enhanced Distributed Channel Access (EDCA) is the governing standard that dictates how QoS functions in the wireless medium. EDCA governs many things for wireless transmission, such as the contention window (CW) size, the interframe sizing that is used for each access category (AC), and so on. The Cisco 5500 series WLAN controller includes feature support for the standards-based WMM EDCA model, plus three other EDCA profiles that adapt the handling to different traffic types. The four EDCA profiles are as follows:

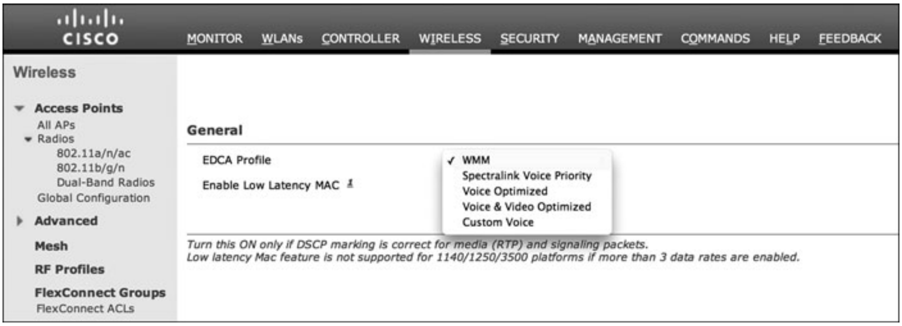
- **WMM** (the default setting): This enables the Wi-Fi Multimedia (WMM) default parameters. Choose this option when voice or video services are not deployed on your network.
- **Spectralink Voice Priority:** Enables Spectralink voice priority parameters. To improve the quality of calls, choose this option if Spectralink phones are deployed on your network.
- **Voice Optimized:** Enables EDCA voice-optimized profile parameters. Choose this option when voice services *other* than Spectralink are deployed on your network.
- **Voice and Video Optimized:** Enables EDCA voice- and video-optimized profile parameters. Choose this option when both voice and video services are deployed on your network.

What makes these profiles different is how the various EDCA parameters are optimized for a given traffic type. The EDCA values that are adjusted in these profiles include the following:

- CW_{min}
- CW_{max}
- Fixed-slot time values
- TXOP values

As a best practice, if you are planning to enable voice services on your wireless network, ensure that the **Voice Optimized** profile is selected. It is important to remember that EDCA governs how frames are transmitted over the air, so this feature applies to the entire AP radio interface, not just a particular WLAN SSID.

To configure EDCA optimization, navigate to the Wireless window, then select one of the radio interfaces (802.11a/n or 802.11b/g/n), and select **EDCA Parameters.** Figure 19-8 demonstrates how to configure the EDCA profile for a given wireless network. In addition, the EDCA parameters must be separately configured for both the 802.11g/b/n and the 802.11a/n radio interfaces on a given lightweight AP.



General

EDCA Profile ✓ WMM

Enable Low Latency MAC

Turn this ON only if DSCP marking is correct for media (RTP) and signaling packets.
Low latency Mac feature is not supported for 1140/1250/3500 platforms if more than 3 data rates are enabled.

Figure 19-8 *Customizing the EDCA Parameters*

There is another useful feature in this window called Enable Low Latency MAC. This optional feature improves the acknowledgment (ACK) response time for RTP packets and can thus help improve the overall QoS handling for voice and video traffic stream.

For example, consider a situation where a voice packet is queued and is waiting to be sent over a wireless interface, but is delayed because the medium is busy. After several retries, the voice packet is eventually sent; however, it will be tossed out by the receiving client because it is too old, meaning the time it spent waiting to be transmitted was a waste of AP resources. The Enable Low Latency MAC feature enhances voice performance by controlling packet retransmits and appropriately aging out older voice packets on lightweight access points, thereby improving the number of voice calls serviced per access point.

There are a few notes of caution to be aware of with this feature, however. You should enable low-latency MAC only if the WLAN allows WMM clients. Also, as can be seen from the system warning message written in blue in Figure 19-8, the correct DSCP marking of RTP packets must be considered as a prerequisite before enabling this feature.

From a design perspective, APs can be configured to accept stations transmitting at different data rates. Although maintaining the lower data rates can improve the capability of a station to transmit when farther away from the AP, this might not be ideal for voice applications. The downside is that when the station transmits at a lower data rate, it could take longer to send a frame of a given size. This introduces more delay and decreases the overall throughput of the wireless medium—which can result in the ability to support fewer simultaneous voice calls. To address this concern, the recommendation is for WLANs that support voice traffic to disable the lower data rates to avoid this potential problem.

Call Admission Control on the Wireless Network

The capability of the Cisco WLC to support VoIP phones is dependent on two key features:

- The enablement of the QoS Platinum profile for voice
- Call admission control (CAC)

In this case, CAC refers to the WLC's capability to limit the number of voice calls on a per-AP basis. Although CAC is generally beyond the scope of this book, in the context of wireless networks it is considered a highly recommended feature to properly support VoIP applications.

Like many other functions in the WLAN, CAC is centrally managed on the WLC, but is enforced on the AP. CAC works to improve voice reliability on the WLAN by controlling the number of simultaneous voice calls that are permitted at any one time on each AP. To illustrate the need for CAC, consider the way in which WMM EDCA treats packets in the Platinum profile compared with non-real-time traffic in the other profiles. If a situation occurred where too many VoIP phones were all contending for access to the medium at the same time, the Platinum profile would be used far beyond its intended design limitations, since each wireless phone would have equal opportunity to transmit over the radio interface. The result would turn the QoS implementation on the AP back into a "best effort" scenario. The result is that voice quality of *all* VoIP calls would suffer without exception.

Enabling WMM QoS Policy on the WLAN

The WMM implementation of CAC in the WLC does not abide by a blind rule of limiting the number of wireless IP phone associations (as if to say, this AP only allows "X" number of VoIP phones and no more). Rather, CAC evaluates several variables, such as the amount of bandwidth used by the AP, the load on the APs, and interference patterns. For example, the WLC can monitor the load on each AP and decide whether there is enough free bandwidth to support another VoIP call beyond what is already in use. If not, any new calls are rejected, and the wireless phone will either have to roam to another AP or it will get a fast-busy signal. This might seem like a harsh enforcement of the "first come, first serve" principle, but the result is that the network does not become oversubscribed and continues to provide a general higher level of service for everyone.

To configure VoIP CAC in the WLC, first navigate to the WLANs screen (shown previously in Figure 19-4), choose **WLANs** from the side panel, and then click the WLAN ID of the WLAN that you want to edit. At the WLAN configuration page, select the **QoS** tab. From here, the WMM attribute must be set to **Allowed** or **Required** to enable WMM-based CAC. Note that if the attribute is set to **Required**, this will prevent any non-WMM clients from communicating on a given WLAN.

If you are using the older Cisco 7920 VoIP phones, which predate WMM, there are two methods of CAC available:

- AP (bandwidth) based
- Client (load) based

If you are using newer Cisco wireless VoIP phones, use WMM to enable CAC and WMM QoS.

Enabling WMM QoS Policy on the WLAN

One of the key enhancements introduced by EDCA is the capability for the wireless clients to signal traffic requirements to the AP before transmitting high-priority traffic. In the 802.11e standard, this is called Traffic Specification (TSpec). During the TSpec communication between client and AP, the client is given a specific Transmission Opportunity (TXOP), or a dedicated window of time where it has exclusive access to the medium so that it can send its high-priority traffic.

Newer WLAN clients (those that are compliant with the 802.11e EDCA specification) use the Add Traffic Stream (ADDTS) TSpec request function to request admission to an AP. The TSpec element communicates several key parameters to the AP, including data rates, frame sizes, and its bandwidth requirement. This information allows the AP to calculate a TXOP for a given client to optimize its access to the wireless medium. This process also allows the WLC to calculate whether an AP has the resources to even meet what has been requested through the TSpec element. Examples of devices that support this feature are the Cisco 792XG family of VoIP phones.

To configure CAC settings for TSpec voice clients, navigate to **Wireless > 802.11a/n** (or **802.11b/g/n**) > **Media**. CAC configuration is available in both the Voice and Video tabs on this page. From this page, CAC configuration parameters allow you to first enable admission control (ACM) for the AP. Note that this is done on a per-radio interface basis, not a per-WLAN basis, so it must be configured on *both* the 802.11a/n and the 802.11 b/g/n radios. Two CAC methods are available in this window:

- **Load Based:** Used only for non-mesh wireless deployments
- **Static:** Used only for mesh wireless deployments

The next parameter specifies the maximum radio-frequency (RF) bandwidth that a radio can use and still accept the initiation of a Voice over WLAN call. The reserved roaming bandwidth parameter defines how much capacity has been set aside for responses to ADDTS requests during association or reassociation, and identifies which are Voice over WLAN clients with calls in progress that are trying to roam to the AP.

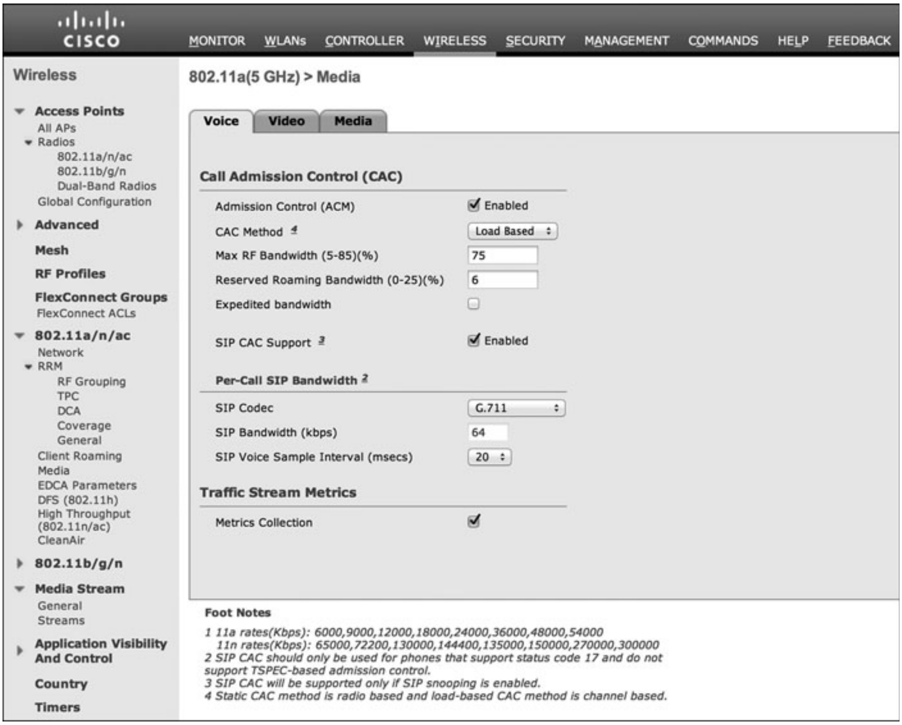
The expedited bandwidth request feature enables clients to indicate the urgency of a WMM traffic specifications (TSpec) request (for example, an emergency 911 call) to the WLAN. When the controller receives this request, it attempts to facilitate the urgency of the call in any way possible without potentially altering the quality of other TSpec calls that are in progress.

Note To take advantage of the expedited bandwidth request feature, clients must be compliant with Cisco Compatible Extensions Version 5 (CCXv5).

This configuration page also allows you to configure the bandwidth in kilobits per second that you want to assign per SIP call on the network. This parameter can be configured only when the SIP codec selected is user defined. It is also important to note that SIP-based CAC will only work if Media Session Snooping (also known as SIP Snooping) is also configured for a given WLAN. Media session snooping is examined more closely in the following section.

The Metrics Collection option determines whether data is collected on voice or video calls for use by Cisco Prime Infrastructure, which is the primary network management tool used for Cisco wireless networks.

Figure 19-9 illustrates how to configure the various CAC features on the WLC.



802.11a(5 GHz) > Media

Call Admission Control (CAC)

Admission Control (ACM) ☑ Enabled

CAC Method [4] Load Based

Max RF Bandwidth (5-85)(%) 75

Reserved Roaming Bandwidth (0-25)(%) 6

Expedited bandwidth ☐

SIP CAC Support [3] ☑ Enabled

Per-Call SIP Bandwidth [2]

SIP Codec G.711

SIP Bandwidth (kbps) 64

SIP Voice Sample Interval (msecs) 20

Traffic Stream Metrics

Metrics Collection ☑

Foot Notes

1 11a rates(Kbps): 6000,9000,12000,18000,24000,36000,48000,54000
11n rates(Kbps): 65000,72200,130000,144400,135000,150000,270000,300000
2 SIP CAC should only be used for phones that support status code 17 and do not support TSPEC-based admission control.
3 SIP CAC will be supported only if SIP snooping is enabled.
4 Static CAC method is radio based and load-based CAC method is channel based.

Figure 19-9 *Configuring CAC for TSpec Clients*

Media Session Snooping (a.k.a. SIP Snooping)

The WLAN QoS design so far has focused on assigning a specific QoS profile to an entire WLAN SSID. If using a VoIP phone, the WLAN gets the Platinum QoS profile. In the Platinum or Gold profiles (which are usually used for voice and video, respectively), the UP values are essentially trusted (up to the default UP value for the respective profile). However, in many cases, the application or the OS on the SIP-capable device does not correctly mark the voice and video packets. The result is that voice traffic is given only best effort handling over the wireless network, even if the WLAN is configured with the Platinum profile.

To overcome this drawback, a new feature was introduced in Version 6.0 of AireOS called Media Session Snooping and Reporting (also known as SIP snooping). The Media Session Snooping feature allows the WLC to detect RFC 3261 (SIP)-compliant voice and video traffic and give it high-priority handling. To configure Media Session Snooping, the WLAN QoS profile must be set to either **Platinum** or **Gold** (that is, this feature is not available if the Silver or Bronze profiles are used).

So how does this actually work? When this feature is enabled, Media Session Snooping allows the AP to investigate every session in a particular WLAN/SSID to determine whether it is SIP. It does this by looking for packets that are destined to or are originating from UDP port number 5060 (the standard SIP signaling port). When packets with these port numbers are recognized, they are considered for further inspection which allows it to detect the ports used for the RTP stream of that SIP session.

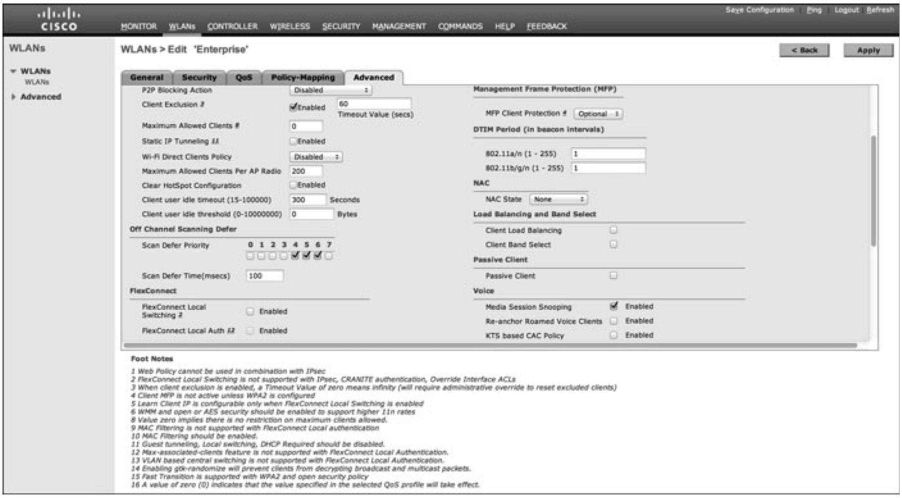
If a SIP session is identified on a WLAN with the Platinum QoS profile, downstream 802.11e frames are given the appropriate 802.11e UP value of 6, regardless of the inner packet's DSCP value. This allows them to be processed by the Voice AC, thus providing the best possible handling over the WLAN.

This feature enables APs to detect the establishment, termination, and failure of SIP voice calls and then report them to the controller and Cisco Prime Infrastructure. VoIP snooping and reporting can be enabled or disabled for each WLAN.

Note Although the AP by default looks for SIP on UDP port 5060, it is also possible to configure SIP inspection on custom ports, or even on a range of different ports.

Because this feature is aimed primarily at upgrading voice SIP packets to the appropriate level of QoS, the feature can only be activated on the Gold and Platinum QoS profiles. The WLC will not let you enable this feature on the Silver and Bronze profiles.

To enable VoIP SIP Snooping, navigate to WLANs, then select the WLAN that you want to enable this feature on. Next, select the **Advanced** tab, and finally, check the **Media Session Snooping** check box under the Voice heading. Figure 19-10 illustrates how to configure SIP Snooping in this way.


cisco
MONITOR WLANs CONTROLLER WIRELESS SECURITY MANAGEMENT COMMANDS HELP FEEDBACK
Save Configuration | Ping | Logout | Refresh
WLANs
▾ WLANs
WLANs
▸ Advanced
< Back
Apply
General
Security
QoS
Policy-Mapping
Advanced

WLANs > Edit 'Enterprise'

P2P Blocking Action: Disabled
Client Exclusion 3: ☑ Enabled 60 Timeout Value (secs)
Maximum Allowed Clients 8: 0
Static IP Tunneling 11: □ Enabled
Wi-Fi Direct Clients Policy: Disabled
Maximum Allowed Clients Per AP Radio: 200
Clear HotSpot Configuration: □ Enabled
Client user idle timeout (15-100000): 300 Seconds
Client user idle threshold (0-10000000): 0 Bytes

Off Channel Scanning Defer

Scan Defer Priority: 0 □ 1 □ 2 □ 3 □ 4 ☑ 5 ☑ 6 ☑ 7 □
Scan Defer Time(msecs): 100

FlexConnect

FlexConnect Local Switching 2: □ Enabled
FlexConnect Local Auth 12: □ Enabled

Management Frame Protection (MFP)

MFP Client Protection 4: Optional

DTIM Period (in beacon intervals)

802.11a/n (1 - 255): 1
802.11b/g/n (1 - 255): 1

NAC

NAC State: None

Load Balancing and Band Select

Client Load Balancing: □
Client Band Select: □

Passive Client

Passive Client: □

Voice

Media Session Snooping: ☑ Enabled
Re-anchor Roamed Voice Clients: □ Enabled
KTS based CAC Policy: □ Enabled

Foot Notes

1 Web Policy cannot be used in combination with IPsec
2 FlexConnect Local Switching is not supported with IPsec, CRANITE authentication, Override Interface ACLs
3 When client exclusion is enabled, a Timeout Value of zero means infinity (will require administrative override to reset excluded clients)
4 Client MFP is not active unless WPA2 is configured
5 Learn Client IP is configurable only when FlexConnect Local Switching is enabled
6 WMM and open or AES security should be enabled to support higher 11n rates
8 Value zero implies there is no restriction on maximum clients allowed.
9 MAC Filtering is not supported with FlexConnect Local authentication
10 MAC Filtering should be enabled.
11 Guest tunneling, Local switching, DHCP Required should be disabled.
12 Max-associated-clients feature is not supported with FlexConnect Local Authentication.
13 VLAN based central switching is not supported with FlexConnect Local Authentication.
14 Enabling gtk-randomize will prevent clients from decrypting broadcast and multicast packets.
15 Fast Transition is supported with WPA2 and open security policy
16 A value of zero (0) indicates that the value specified in the selected QoS profile will take effect.

Figure 19-10 *Enabling SIP Session Media Snooping*

Media Session Snooping is certainly a useful QoS tool that helps SIP traffic get the desired handling over wireless networks, but it is clearly limited in its capability to provide per-session or per-user QoS for any non-SIP traffic type. For example, Media Session Snooping is not able to recognize SCCP (Skinny) traffic. This means that if a legacy wireless VoIP phone is being used, it would be impossible to provide the level of QoS required for this traffic.

Although Media Session Snooping does have its limitations, another advance in Cisco WLC technology has opened up some new wireless QoS design options. This new feature is called application visibility control (AVC), and is discussed in the following section.

Application Visibility Control in the WLC

AVC was introduced in Chapter 9, "Application Visibility Control (AVC)," as a solution that uses multiple technologies, including Network Based Application Recognition Version 2 (NBAR2), Flexible NetFlow (FNF), and management tools that, when working together, provide powerful application visibility and control capabilities based on stateful deep packet inspection (DPI).

With the Cisco AVC solution available on wireless controllers from AireOS 7.4 onward, it is possible to identify applications within the mixed traffic flow using DPI and then mark them with an appropriate DSCP value. Based on the DPI results of the AVC feature, these applications can be identified and then mapped to one of the four QoS policies available on the WLC to ensure correct handling across the wireless network. Because wireless is also an access layer of the network, AVC allows the WLC to enforce QoS trust rules that were never possible before, albeit only on the WLC (in AireOS 7.4 AVC functions only on the WLC, not on the AP). With the AVC feature set enabled on the WLC, it is important to understand that DPI and DSCP marking occurs both in the upstream and downstream directions on the WLC—that is, the AVC feature set has no concept of the upstream or downstream directions.

With AVC enabled, it is possible on the WLC to identify over a thousand applications and re-mark the DSCP to whatever is desired. Unlike the WLAN QoS configuration that was discussed previously, AVC has the capability to mark the original IP packet's DSCP value, not just the CAPWAP tunnel DSCP value. Naturally, as has been discussed throughout this book, the trust boundary should be deployed as close to the edge of the network as possible, not as packets are entering the WLC from the wired network. If for some reason packets are not correctly marked at the edge, however, this could be easily done by the AVC functionality in the WLC. Also, packets originating from the wireless network can be classified and marked on the WLC before entering the wired network. It is interesting to consider the usefulness of marking upstream packets on the WLC as they are passed into the wired network. Although the WLC can hardly be considered the edge of the network, it does offer the possibility to look deeper into the packet than any edge switch can today, thus providing a powerful and granular tool for application classification.

With DSCP re-marking capabilities, the downstream QoS handling at the AP can be more methodically controlled by allowing different applications to access the correct WMM ACs on the AP. Because AVC operates only on the WLC (unlike the other WLAN QoS policy features that operate on both the WLC and AP), it has an effect only on the downstream direction into the wireless network (once an upstream packet is received on the WLC and is processed by AVC it has already traversed the wireless network, thus from a strictly wireless perspective, it can only influence downstream traffic).

It has already been mentioned, but it is worth restating this fundamental fact: AVC re-marking on the WLC has no concept of direction—meaning that whether packets are headed upstream to the wired LAN or are headed downstream toward the AP, AVC re-marks them in both directions.

The following summarizes the interaction of AVC and QoS in the WLC in both the upstream and downstream directions:

Upstream direction (from wireless client to wired network):

1. A wireless client begins to transmit traffic.
2. The DSCP and 802.11e UP values are set by the client, and the data transmitted to the AP through the correct AC.
3. The AP maps the UP value to a DSCP value on the CAPWAP tunnel, but does not modify the inner DSCP marking.
4. The WLC receives the incoming packed via CAPWAP.
5. The WLC, using NBAR2, examines the packet at the application layer and applies the AVC policy, re-marking the DSCP to the desired value.
6. The packet is sent to the wired network with the new DSCP value.

Downstream direction (from wired network to wireless client):

1. A packet is sent to the WLC from the wired network.
2. Using NBAR2, the WLC examines the packet, applies the AVC policy, and rewrites the DSCP into the *original* packet header.
3. The WLC compares this new DSCP value to the WLAN QoS profile. The new DSCP is mapped to the CAPWAP DSCP value. The AVC DSCP marking takes precedence over the upper DSCP limit established by the QoS profile.
4. When the AP receives the CAPWAP packet, it maps the DSCP value to the appropriate 802.11e UP value.
5. The packet is transmitted to the wireless client through the correct AC on the AP.

From a visibility and monitoring perspective, the WLC can collect and display various wireless performance metrics, such as bandwidth usage for individual clients and applications. This reporting information can be both displayed locally on the controller or exported through FNF to a management tool.

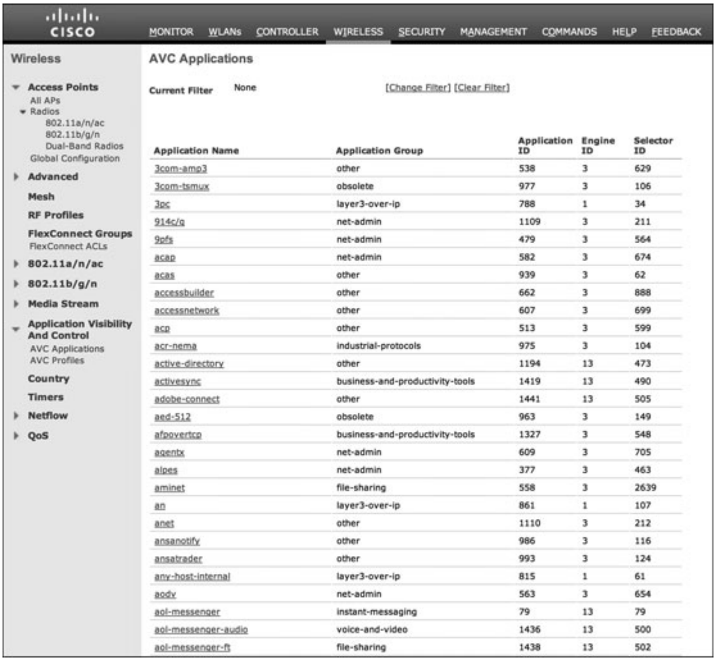
Because AVC uses the same NBAR2 DPI engine as IOS, through this technology the WLC has the capability to identify more than 1000 applications, including Oracle, SAP, Citrix, BitTorrent, Microsoft Exchange, Skype, Facebook, and many others. To view details of the list of supported AVC applications, navigate to **Wireless > Application Visibility and Control > AVC Applications**. Figure 19-11 shows a partial list of the applications supported by AVC in the WLC.

As with the AVC feature set available in IOS, the applications are collected together in application groups. If you are familiar with the IOS-based AVC feature (see Chapter 9 for more details on AVC), you will note that the list of available applications in the WLC is, for the most part, identical to the list of applications available in Cisco ISR G2 and ASR 1000 routers.

Within the application group, you can find the individual applications to match. Figure 19-12 shows an example of the voice and video applications that AVC is able to discover and control using NBAR2.

Consider an example of how to use AVC on the WLC. Assume you have configured a WLAN called Enterprise. This wireless WLAN serves a variety of clients that include many real-time application, including tablet PCs, smartphones, and laptops. Because the Enterprise WLAN is a multi-use WLAN, it is given the global QoS policy of Platinum. This basically means that all QoS markings are trusted up to a DSCP value of 46 by default, regardless of what is set on the inner IP packet.

Naturally, the challenge with trusting everything coming from the wired network is that you do not know if the DSCP values are truly set correctly. For example, many voice or video devices may not even set DSCP values, and are thus treated as best effort. Similarly, there may be best effort applications that are causing trouble on the wireless network because they mark all traffic as EF and need to be downgraded to the Silver or Bronze profiles, even though they are on WLAN configured as Platinum.

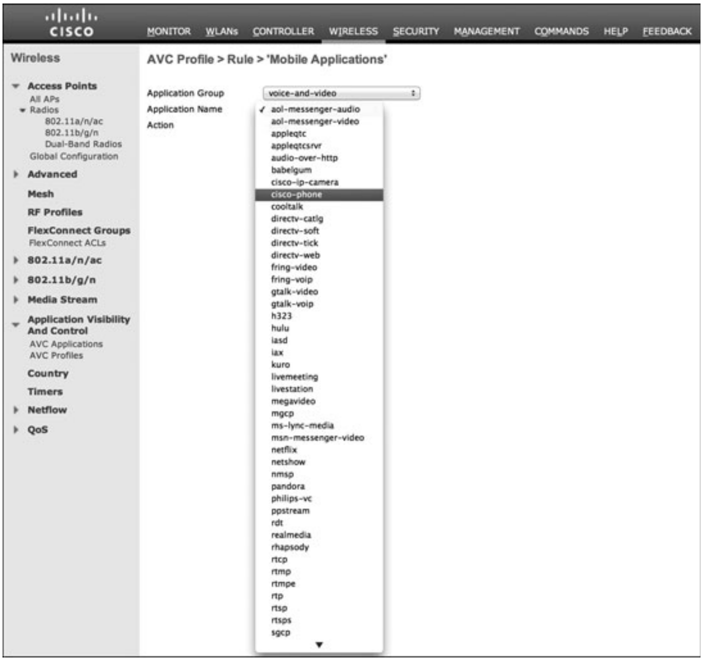


AVC Applications

Current Filter None

Application Name	Application Group	Application ID	Engine ID	Selector ID
3com-amp3	other	538	3	629
3com-tsmux	obsolete	977	3	106
3pc	layer3-over-ip	788	1	34
914c/g	net-admin	1109	3	211
9pfs	net-admin	479	3	564
acap	net-admin	582	3	674
acas	other	939	3	62
accessbuilder	other	662	3	888
accessnetwork	other	607	3	699
acp	other	513	3	599
acr-nema	industrial-protocols	975	3	104
active-directory	other	1194	13	473
activesync	business-and-productivity-tools	1419	13	490
adobe-connect	other	1441	13	505
aed-512	obsolete	963	3	149
afpovertcp	business-and-productivity-tools	1327	3	548
agentx	net-admin	609	3	705
alpes	net-admin	377	3	463
aminet	file-sharing	558	3	2639
an	layer3-over-ip	861	1	107
anet	other	1110	3	212
ansanotify	other	986	3	116
ansatrader	other	993	3	124
any-host-internal	layer3-over-ip	815	1	61
aodv	net-admin	563	3	654
aol-messenger	instant-messaging	79	13	79
aol-messenger-audio	voice-and-video	1436	13	500
aol-messenger-ft	file-sharing	1438	13	502

Figure 19-11 *Examining the AVC Application List*



AVC Profile > Rule > 'Mobile Applications'

Application Group voice-and-video

Application Name aol-messenger-audio

Action

Figure 19-12 *AVC Applications*

With this in mind, AVC can be used to protect a variety of different voice and video applications by elevating their QoS handling, ensuring that these applications are given the correct DSCP value and corresponding 802.11e UP value in the downstream direction.

An interesting design concept that allows you to enforce a trust boundary at the WLC is to use the wired switch to re-mark all downstream traffic to DSCP 0 before it even enters the WLC. AVC can then be used to selectively identify applications and re-mark them to the correct DSCP level as they are transmitted to the wireless client (this is not an actual design recommendation, but rather is used to illustrate how AVC might be used).

AVC can even drop unwanted traffic types that are considered as a nuisance to the network administrator. For example, perhaps Hulu traffic is considered problematic on your network, and it is determined that it should not be allowed on the Enterprise WLAN at all. Using AVC, Hulu can be easily indentifed and the action of "drop" can be enabled for this traffic type.

To configure AVC on the WLC, begin by creating a new AVC profile under the Wireless tab, then navigate to Application Visibility and Control, and finally click on **AVC Profiles.** Click on **New** and enter the AVC profile name. In this case, the policy name **Protect Real Time Apps** is entered.

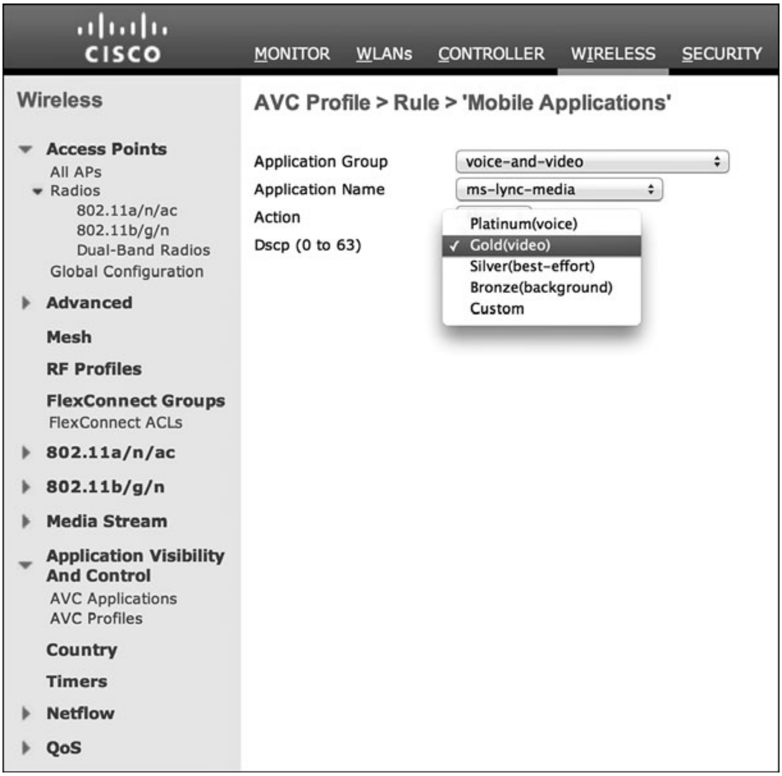
Note In the WLC, a total of 16 different AVC policies may be created, with a maximum of 32 rules per policy.

Figure 19-13 illustrates a simple example of how to create a new rule that re-marks Microsoft Lync traffic. This application is listed under the voice-and-video application group category. In this case, the action is taken to re-mark the DSCP of Lync to Gold (which re-marks it to DSCP 34).

If the Mark option is chosen, the WLC gives five marking options:

- Platinum (Packets are re-marked by default to DSCP 46.)
- Gold (Packets are re-marked by default to DSCP 34.)
- Silver (Packets are re-marked by default to DSCP 0.)
- Bronze (Packets are re-marked by default to DSCP 10.)
- Custom (You can re-mark to whatever DSCP you choose.)

If packets are re-marked with one of the precious metal profiles, these DSCP values will map to the 802.11e UP values discussed previously. If you choose a custom DSCP, this value is marked on both the original IP packet header and on the downstream CAPWAP tunnel, and is in turn mapped to the appropriate 802.11e UP value according to Tables 18-5 and 18-6 in Chapter 18.



AVC Profile > Rule > 'Mobile Applications'

Application Group	voice-and-video
Application Name	ms-lync-media
Action	
Dscp (0 to 63)	Gold(video)

Figure 19-13 *Re-marking DSCP Using AVC*

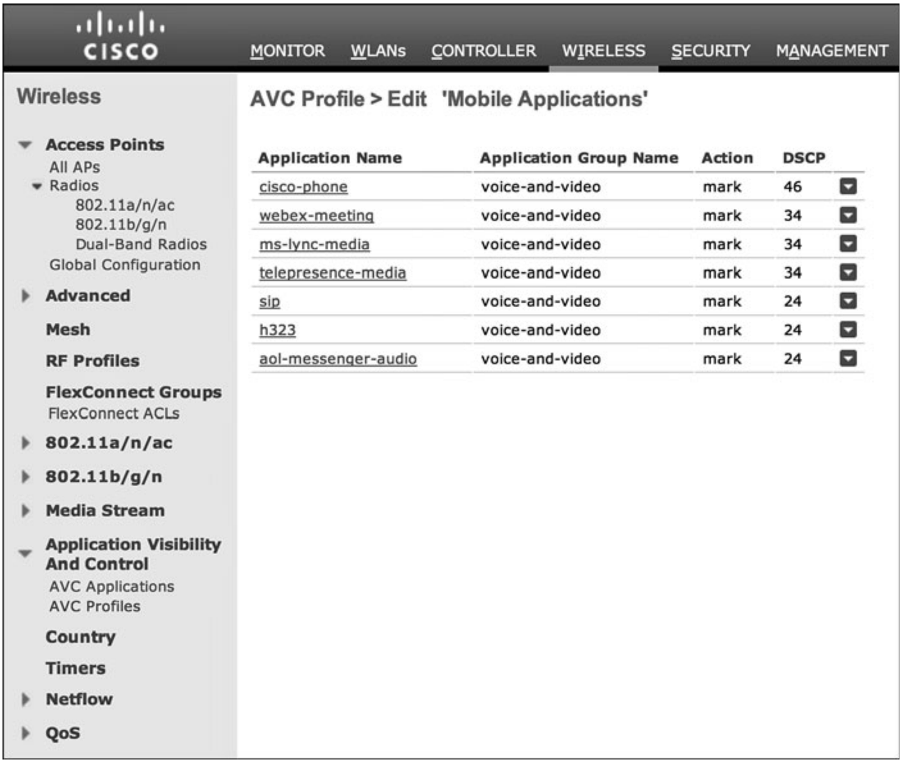
Figure 19-14 shows a more comprehensive AVC policy that considers a variety of applications that may run on a mobile device.

In this example, IP phone traffic is marked as EF (46). Other real-time applications are marked as AF41 (34), and signaling traffic is marked as CS3 (24).

After the AVC rules have been created, the policy can now be added to the QoS policy for the Enterprise WLAN. First, AVC needs to be enabled for the WLAN. AVC is never turned on by default (because there can be a significant CPU hit caused by the NBAR2 DPI engine during higher traffic periods). Navigate to WLANs, and then select the WLAN that you want to edit, and select the **QoS** tab.

Even though the Enterprise WLAN has been configured with the Silver QoS policy, AVC policy always take precedence over the WLAN QoS policy.

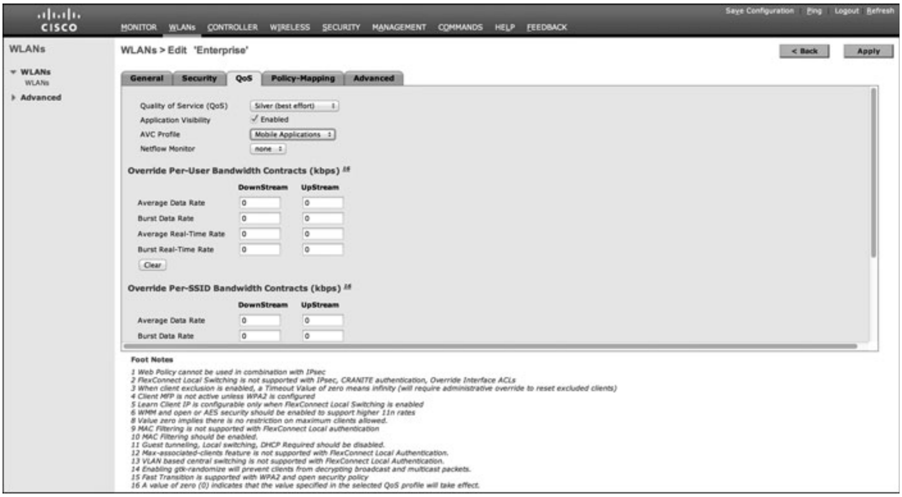
Figure 19-15 demonstrates how the AVC policy may be added to the WLAN QoS policy.



AVC Profile > Edit 'Mobile Applications'

Application Name	Application Group Name	Action	DSCP
cisco-phone	voice-and-video	mark	46
webex-meeting	voice-and-video	mark	34
ms-lync-media	voice-and-video	mark	34
telepresence-media	voice-and-video	mark	34
sip	voice-and-video	mark	24
h323	voice-and-video	mark	24
aol-messenger-audio	voice-and-video	mark	24

Figure 19-14 *A Comprehensive AVC Example*



WLANs > Edit 'Enterprise'

Quality of Service (QoS): Silver (best effort)
Application Visibility: ✓ Enabled
AVC Profile: Mobile Applications
Netflow Monitor: none

Override Per-User Bandwidth Contracts (kbps) [16]

	DownStream	UpStream
Average Data Rate	0	0
Burst Data Rate	0	0
Average Real-Time Rate	0	0
Burst Real-Time Rate	0	0

Override Per-SSID Bandwidth Contracts (kbps) [16]

	DownStream	UpStream
Average Data Rate	0	0
Burst Data Rate	0	0

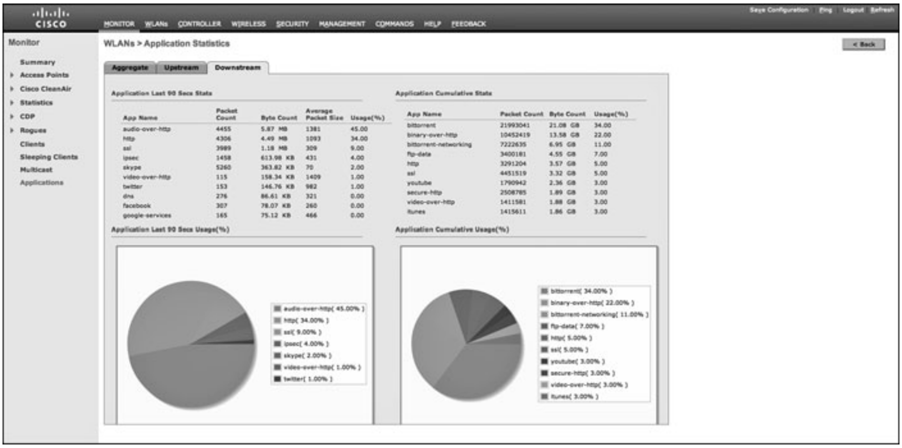
Foot Notes

1 Web Policy cannot be used in combination with IPsec
2 FlexConnect Local Switching is not supported with IPsec, CRANITE authentication, Override Interface ACLs
3 When client exclusion is enabled, a Timeout Value of zero means infinity (will require administrative override to reset excluded clients)
4 Client MFP is not active unless WPA2 is configured
5 Learn Client IP is configurable only when FlexConnect Local Switching is enabled
6 WMM and open or AES security should be enabled to support higher 11n rates
8 Value zero implies there is no restriction on maximum clients allowed.
9 MAC Filtering is not supported with FlexConnect Local authentication
10 MAC Filtering should be enabled.
11 Guest tunneling, Local switching, DHCP Required should be disabled.
12 Max-associated-clients feature is not supported with FlexConnect Local Authentication.
13 VLAN based central switching is not supported with FlexConnect Local Authentication.
14 Enabling gtk-randomize will prevent clients from decrypting broadcast and multicast packets.
15 Fast Transition is supported with WPA2 and open security policy
16 A value of zero (0) indicates that the value specified in the selected QoS profile will take effect.

Figure 19-15 *Applying the AVC QoS Policy to the Enterprise WLAN*

It is important to note that AVC inspection and control happens on the WLC, not the AP. This means that if a packet originated from the wireless side, the AVC re-marking only has an effect on the packets as they leave the controller and enter the wired side of the network. Conversely, if the traffic originated from the wired side of the network and flow to the wireless side, then re-marking happens on the WLC before being transmitted over the CAPWAP tunnel toward the AP (of course, the actual queuing of the packets happens on the AP, not the controller).

The WLC also offers some exceptional visibility capabilities based on AVC. Figure 19-16 illustrates the information that can be gleaned from the AVC feature set. Notice here that AVC is able to discover and report on application statistics in both the upstream and downstream directions, and the total aggregate. Figure 19-16 illustrates the AVC reporting function.



WLANs > Application Statistics

Application Last 90 Secs Stats

App Name	Packet Count	Byte Count	Average Packet Size	Usage(%)
audio-over-http	4455	5.87 MB	1381	45.00
http	4306	4.49 MB	1093	34.00
ssl	3989	1.18 MB	309	9.00
ipsec	1458	613.98 KB	431	4.00
skype	5260	363.82 KB	70	2.00
video-over-http	115	158.34 KB	1409	1.00
twitter	153	146.76 KB	982	1.00
dns	276	86.61 KB	321	0.00
facebook	307	78.07 KB	260	0.00
google-services	165	75.12 KB	466	0.00

Application Cumulative Stats

App Name	Packet Count	Byte Count	Usage(%)
bittorrent	21993041	21.08 GB	34.00
binary-over-http	10452419	13.58 GB	22.00
bittorrent-networking	7222635	6.95 GB	11.00
ftp-data	3400181	4.55 GB	7.00
http	3291204	3.57 GB	5.00
ssl	4451519	3.32 GB	5.00
youtube	1790942	2.36 GB	3.00
secure-http	2508785	1.89 GB	3.00
video-over-http	1411581	1.88 GB	3.00
itunes	1415611	1.86 GB	3.00

Application Last 90 Secs Usage(%)

Application Cumulative Usage(%)

Figure 19-16 *Examining the AVC Application Statistics Report*

Developing a QoS Strategy for the WLAN

This book presents three generic QoS class models: the 4-class, 8-class, and 12-class models. As was discussed in Chapter 18, 802.11e and WMM only offer a maximum of four ACs, or queues that may be used. Because these models don't easily line up with the available access categories, wireless networks present a unique challenge when attempting to incorporate a QoS policy that marries both the campus wired and wireless QoS designs. This section discusses how to build a consolidated QoS policy for the three generic class models.

Four-Class Model Design

In accord with the WMM standard, the 5500 series WLC supports a maximum of four QoS classes (corresponding to the four WMM ACs). On the surface this may seem to line up well with the standard four classes presented in this book. In fact, if a four-class model were deployed, a 1:1 mapping to the four WMM ACs would be the desired design. However, if careful attention is given to the standard DSCPs used by the four-class model and the corresponding mapping of DSCPs to 802.11e UP values, an unexpected result arises.

Figure 19-17 illustrates how the standard four-class model stacks up with the wireless WMM mode.

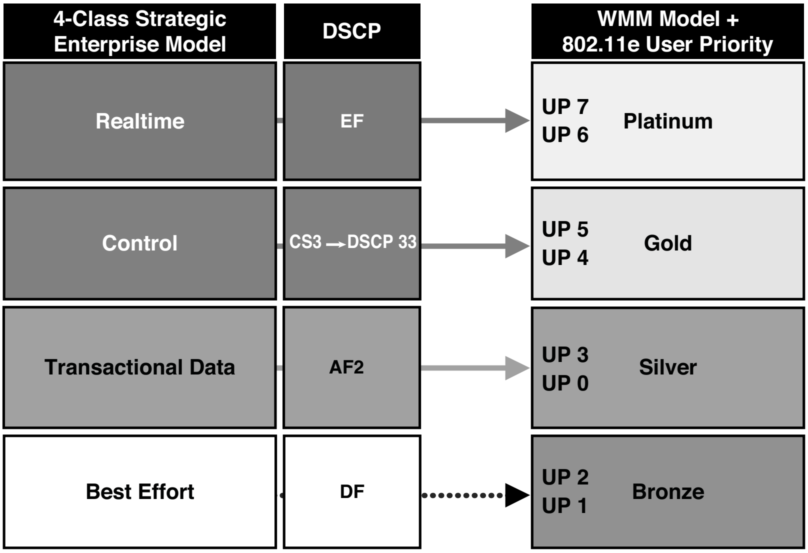


Figure 19-17 *A Design Based on the Four-Class Model*

As shown in Figure 19-17, the default mappings come out with a surprising result. While EF traffic is easily mapped to the Platinum queue, in the downstream direction the DSCP for network control (CS3 = DSCP 24) is mapped to UP 3 (see Table 18-6 in Chapter 18), thus ending up in the best effort AC. This mapping is set by default and is not adjustable in AireOS WLCs (the 5760 which is an IOS-based controller has the ability to configure this mapping table). The net result of this is that the four-class model in the campus network ends up being a two-class model over the wireless medium.

Tweaking the QoS Classification Downstream

Although this default mapping ensures correct handling of voice traffic, no other application really benefits from wireless QoS, as they all end up getting handled as best effort traffic in the Silver WMM AC. Of particular concern is that voice control and signaling traffic is not given any preferential treatment, and is also handled by the Silver WMM AC.

Consider an alternative approach where you perform re-marking at the LAN access switch connected to the wireless AP. On the egress interface in the outbound direction toward the AP you can re-mark the CAPWAP DSCP values such that the Control/Signaling application class is re-marked to a DSCP value that maps to an UP value that will—in turn—be assigned to the Gold WMM AC. Similarly, best effort traffic may be mapped to a DSCP value that maps to an UP value that will—in turn—be assigned to the Bronze WMM AC.

This remapping approach leaves the Silver WMM AC to exclusively service transactional data applications and thereby reflect the same overall relative priority of servicing as the original model.

Perhaps this question may arise: Rather than configuring mapping policies on the access switch connected to the AP, why not just configure an AVC policy on the WLC to match Control/Sgnaling traffic and re-mark it to a DSCP value that maps to the Gold WMM AC? For example, if you re-mark signaling traffic from CS3 to CS4/DSCP 32 it will ensure that this traffic gets the correct UP value and is handled by a the Gold AC)? Keep in mind, though, that such an AVC marking policy on the WLC, which operates both in the upstream and downstream directions, interferes with QoS policies on the rest of the wired network (which includes both the upstream network and the transit network between the WLC and the APs). For example, if a voice control packet were to be re-marked from CS3 to CS4 on the WLC just so it will land in the Gold AC when it arrives at the AP, then all the routers and switches in the middle, between the WLC and the AP that the CAPWAP tunnel goes over, would have an inconsistent classification scheme. So while this approach may put the packet into the correct downstream AC on the AP, it will break QoS in the transit network.

Therefore, a much simpler approach to achieve the same end result is to include an outbound re-marking policy on the final access switch that connects to the AP. However, rather than remapping signaling traffic to a code point that may be used for another application, it may be better to re-mark signaling to a nonstandard code point for this one-time operation, such as DSCP 33. DSCP 33 maps (by default) to the Gold WMM AC and would uniquely identify this remapped signaling traffic. This scheme ensures that signaling traffic is handled by the correct AC on the AP while at the same time not breaking QoS over the transit network.

Similarly, the default Best Effort class could be remapped on the network access switch to a nonstandard code point that would be assigned to the Bronze WMM AC (such as DSCP 9). However, some platforms, such as the Catalyst 3750, do not support egress marking policies and only support marking/re-marking/DSCP mutation policies on ingress. In such a case, re-marking best effort traffic on access switch ingress will affect BE marking on all interfaces—not just the interface connecting to the AP.

It bears noting that assigning the default class to the Background WMM category will marginally delay best effort traffic. However, this is the nature of QoS; there is always a tradeoff. In this scenario, this is the necessary cost of providing superior levels of service to the Signaling and Transactional Data application classes, thus providing the four levels of service required to meet their strategic business objectives of QoS.

Figure 19-18 shows the modified mapping of a four-class QoS model to WMM.

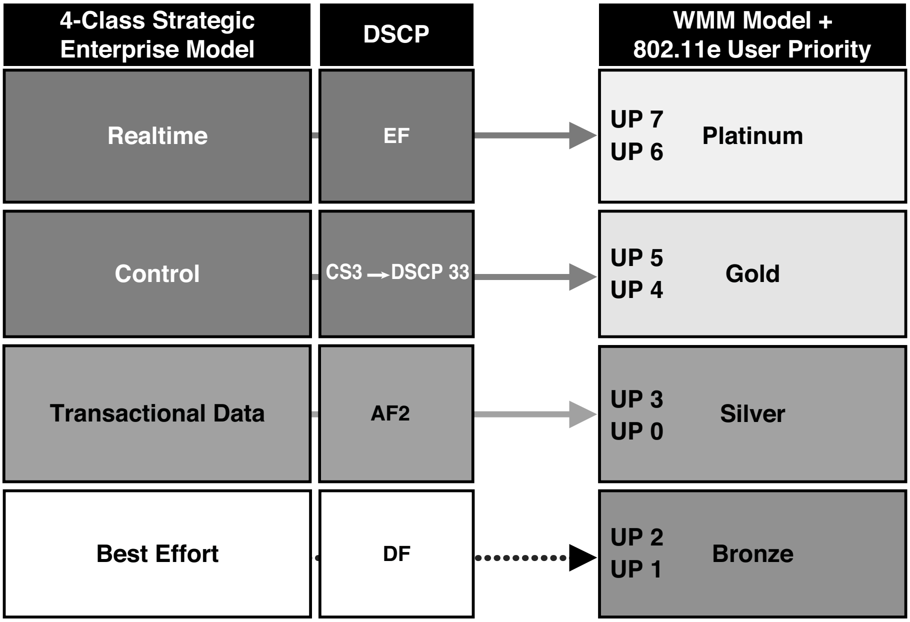


Figure 19-18 *Modified Downstream Four-Class Enterprise Model Mapping to WMM*

As noted, to achieve this QoS classification model, it is required to mutate the DSCP value of the CAPWAP header before it enters the AP. This requires switch-level configuration anywhere an AP is connected to the network. Naturally, each switching platform is configured in a slightly different way. However, the following is an example of how to configure the Catalyst 3750X switch to achieve this desired result.

As noted, the Catalyst 3750 does not support egress re-marking; only ingress marking or ingress DSCP mutation is supported, as shown in Figure 19-19.

The Catalyst 3750 employs Multi-Layer Switch QoS (MLS QoS), and therefore DSCP-remarking policies can be configured in one of two ways:

- **Class-based marking policy** (as shown in Example 19-1)
- **DSCP Mutation** (as shown in Example 19-2)

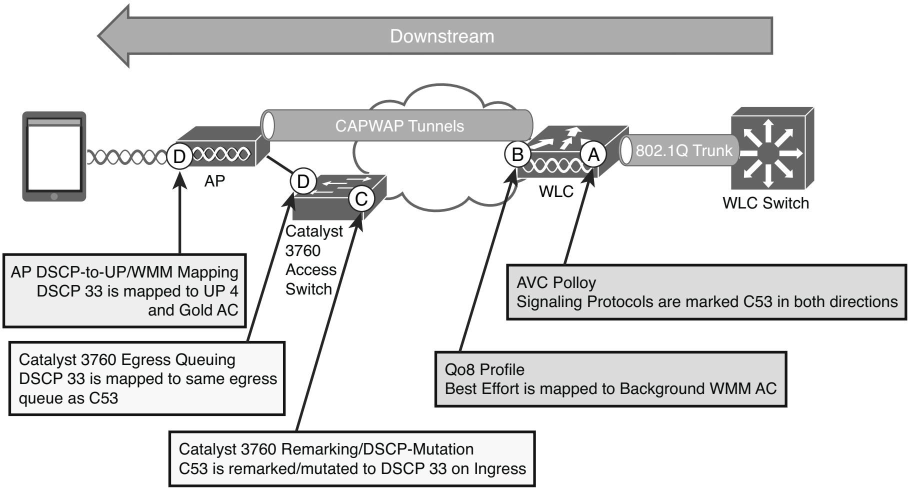

Downstream
CAPWAP Tunnels
D
AP
D
C
Catalyst
3760
Access
Switch
B
A
WLC
802.1Q Trunk
WLC Switch
AP DSCP-to-UP/WMM Mapping
DSCP 33 is mapped to UP 4
and Gold AC
Catalyst 3760 Egress Queuing
DSCP 33 is mapped to same egress
queue as C53
Catalyst 3760 Remarking/DSCP-Mutation
C53 is remarked/mutated to DSCP 33 on Ingress
AVC Polloy
Signaling Protocols are marked C53 in both directions
Qo8 Profile
Best Effort is mapped to Background WMM AC

Figure 19-19 *Mutating the CAPWAP DSCP to Align the QoS Classification Model*

Example 19-1 *Catalyst 3750 Downstream Four-Class Enterprise Model Mapping Policy to WMM via Class-Based Marking*

```
 ! This section configures the class-map
C3750-X(config-cmap)# class-map match-all SIGNALING
C3750-X(config-cmap)#  match ip dscp cs3
 ! Signaling traffic is matched on DSCP CS3

 ! This section configures the Network Downstream Remarking policy map
C3750-X(config-cmap)# policy-map DOWNSTREAM-WMM-REMARKING
C3750-X(config-pmap-c)# class SIGNALING
C3750-X(config-pmap-c)#  set dscp 33
 ! Signaling is re-marked DSCP 33 to map into WMM Gold downstream

 ! This section attaches the policy to the campus-side interface
C3750-X(config)# interface TenGigabitEthernet2/1/1
C3750-X(config-if)# mls qos trust dscp
 ! Configures the port to statically trust DSCP on ingress
C3750-X(config-if)# service-policy input DOWNSTREAM-WMM-REMARKING
 ! Attaches the Downstream DSCP re-marking policy to the interface on ingress
```

You can verify this configuration with the following commands:

- **show mls qos interface**
- **show class-map**

- **show policy-map**
- **show policy-map interface**

Example 19-2 *Downstream Four-Class Enterprise Model Mapping Policy to WMM via DSCP Mutation*

```
 ! This section configures the Downstream DSCP Mutation map
C3750-X(config)# mls qos map dscp-mutation DOWNSTREAM-WMM-MUTATION 24 to 33
 ! This section attaches the Downstream policy to the campus-side interface
C3750-X(config)# interface TenGigabitEthernet2/1/1
C3750-X(config-if)# mls qos trust dscp
 ! Configures the port to statically trust DSCP on ingress
C3750-X(config-if)# mls qos dscp-mutation DOWNSTREAM-WMM-MUTATION
 ! Attaches the Downstream DSCP mutation map to the interface on ingress
```

You can verify this configuration with the following commands:

- **show mls qos interface**
- **show mls qos maps dscp-mutation**

Tweaking the QoS Classification Upstream

The same issues just discussed related to mapping of DSCP values are also true in the reverse direction, although one interesting anomaly emerges. In the upstream direction, the 802.11e UP value is mapped to a DSCP value on the CAPWAP header (see Table 18-5 and Table 18-6 in Chapter 18). This poses an interesting problem for voice signaling and control. Most mobile applications automatically use UP 4 for voice signaling (contrast this with the standard DSCP value of CS3 / DSCP24). However, according to the way the AP maps UP to DSCP on the CAPWAP header, UP 4 becomes DSCP 26 (AF31), and not DSCP 24 (CS3) that you would expect. This mismatch in translating DSCP values means the underlying IP transit network that the CAPWAP packets pass over no longer recognize voice control traffic since the DSCP value has changed.

Similar to the downstream direction, one possible workaround is to use a manual re-marking procedure on the ingress port of the LAN switch that is connected to the AP. In this case, a QoS policy can be configured to convert all DSCP AF31 traffic into DSCP CS3. This will ensure that the IP transit network correctly manages voice control traffic in the CAPWAP tunnel.

Example 19-3 shows the upstream ingress configuration required on a Catalyst 3750 switch to remap signaling traffic from AF31 to the enterprise marking of CS3.

Example 19-3 *Catalyst 3750 Upstream Signaling-Marking Restoration*

```
 ! This section configures the class map
C3750-X(config-cmap)# class-map match-all AP-SIGNALING
C3750-X(config-cmap)#  match ip dscp af31
 ! Signaling traffic from the AP is matched on DSCP AF31

 ! This section configures the Network Upstream Remarking policy map
C3750-X(config-cmap)# policy-map UPSTREAM-WMM-REMARKING
C3750-X(config-pmap-c)# class AP-SIGNALING
C3750-X(config-pmap-c)#  set dscp cs3
 ! Signaling is re-marked to DSCP 24 to align to enterprise QoS model

! This section attaches the upstream policy to the AP-connected interface
C3750-X(config)# interface GigabitEthernet1/0/10
C3750-X(config-if)# mls qos trust dscp
 ! Configures the port to statically trust DSCP on ingress
C3750-X(config-if)# service-policy input UPSTREAM-WMM-REMARKING
 ! Attaches the upstream DSCP re-marking policy to the AP interface on ingress
```

You can verify this configuration with the following commands:

- **show mls qos interface**
- **show class-map**
- **show policy-map**
- **show policy-map interface**

Eight-Class Model Design

If the eight-class model is used in the campus network, there is more flexibility to use the four WMM ACs on the wireless network. Figure 19-20 illustrates the mappings of DSCP to 802.11e UP values that are used on the APs.

Similar to the four-class model, real-time voice traffic is mapped to the Platinum policy, but this time video traffic is added and is correctly applied to the Gold policy. In the eight-class model, network control (meaning Open Shortest Path First [OSPF], Border Gateway Protocol [BGP], Operations, Administration, and Maintenance [OAM], and so on) is not used, so this can be ignored. Like the four-class model, signaling (CS3), transactional data (AF21), and best effort traffic all end up getting mapped to the Silver AC. Again, this is not well suited to voice control traffic, but is nonetheless unavoidable due to the inherent inconsistencies between IETF and IEEE 802.11e QoS mappings.

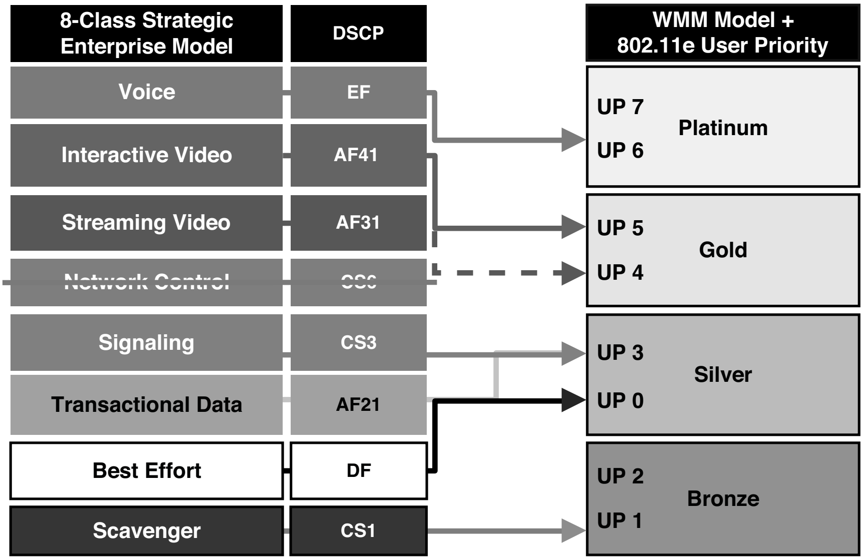


Figure 19-20 *A Design Based on the Eight-Class Model*

EF, AF41, AF31, and AF21 all have 1:1 mappings in the 802.11e UP to DSCP conversion table, so the upstream packets are handled as expected by the underlying IP transport network. Again, only CS3 traffic is incorrectly mapped to AF31 (DSCP 26). As demonstrated with the four-class model, signaling traffic can be manipulated into the desired AC using a system of DSCP mutation at the access switch connected to the AP.

Twelve-Class Model Design

If the 12-class model is used in the campus network, this opens up even more granularity of application management on the wireless interfaces. Figure 19-21 illustrates how the 12 classes are mapped to the respective 802.11e UP values.

In the 12-class model, it is not necessary to use either the Network Control (CS6) or OAM (CS2) classes, because these are not usually sent over wireless networks. As shown in Figure 19-21, all the video applications are mapped correctly to the Gold AC, and the same classes as the four- and eight-class models are mapped to the Silver AC. The new addition here is that bulk data is mapped to the Bronze AC.

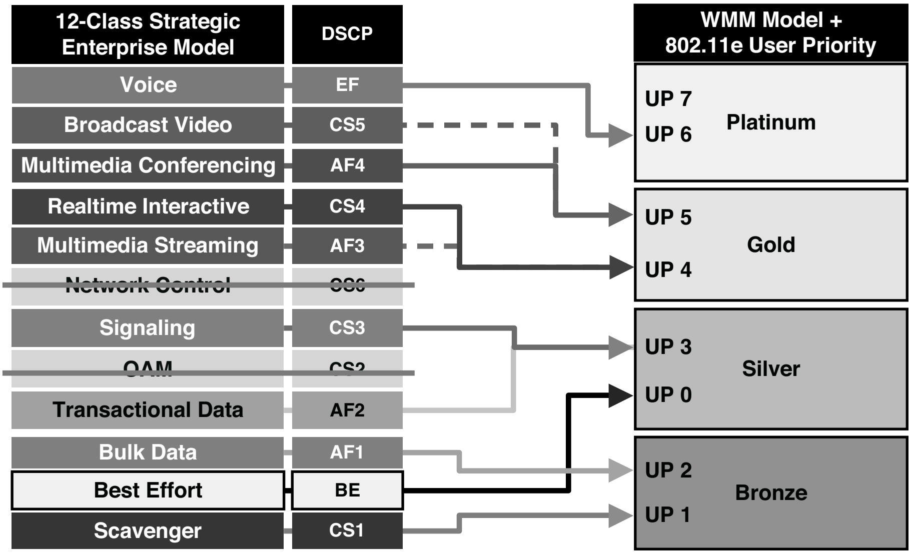


Figure 19-21 *A Design Based on the 12-Class Model*

Summary

This chapter discussed the role of QoS in an enterprise WLAN using the WLC 5500 series controller and Cisco wireless APs. Various QoS enforcement points in the WLAN architecture were examined, and the QoS tools that are used in these various points were considered.

Next, the QoS design of the WLC 5500 series controller was examined in detail, beginning with a discussion of for four QoS profiles and how they function in the WLC. Several aspects of the QoS design were discussed, including the recommendation to always use separate voice and video WLANs and to treat these with the appropriate QoS policy. Several key QoS features were discussed, including how to rate limit individual wireless clients and how to deploy a QoS policy for guest users.

The next section discussed some best practice design principles required to support real-time applications on the WLAN. This included ways to tweak the performance of EDCA in the WLAN, and how to apply CAC—a necessary and recommended QoS feature when deploying wireless networks.

The next section highlighted how to detect and apply special QoS policies to SIP traffic using Media Session Snooping. This feature is applied to both the AP and the WLC and can provide the correct QoS handling to SIP packets, even though they might not be marked correctly from the source. It was also shown that when using voice on a mixed-traffic WLAN, the Platinum profile should be used along with Media Session Snooping as a recommended best practice.

Although Media Session Snooping proves to be a useful feature, the discussion on AVC highlighted ways to leverage Cisco's NBAR2 technology to identify over a thousand applications and apply custom QoS policies to them. It was shown through a detailed example how the AVC feature set can be used whenever real-time services are being used on a mixed-use WLAN.

The final section examined how to work with the 4-class, 8-class, and 12-class reference designs that are used in the campus wired network. Several challenges were examined that have resulted from a mismatch in how the IETF and IEEE DSCP mappings are implemented. Some suggested workarounds were explored that promote the correct QoS handling of certain applications that suffer from this mismatch.

Further Reading

Voice over Wireless LAN Design Guide 4.1: http://www.cisco.com/en/US/partner/docs/solutions/Enterprise/Mobility/vowlan/41dg/vowlan41dg-book.html

Wireless LAN Controller Configuration Guide 7.3: http://www.cisco.com/en/US/partner/docs/wireless/controller/7.3/configuration/guide/b_cg73.html

Cisco Unified Wireless IP Phone 7921g Deployment Guides: http://www.cisco.com/en/US/products/hw/phones/ps379/prod_installation_guides_list.html

Cisco Compatible Extensions Program: http://www.cisco.com/go/ccx

This page intentionally left blank

Chapter 20

Converged Access (Cisco Catalyst 3850 and the Cisco 5760 Wireless LAN Controller) QoS Design

Thus far, Part IV of the book has covered the wireless overlay model deployed today with the current Cisco Unified Wireless Networking (CUWN) wireless solution based on the 5508 wireless LAN controller. This solution provides a centralized wireless controller that applies QoS policy to access points (APs) via a centralized graphical user interface (GUI). Application classification in this solution is based on Wi-Fi Multimedia (WMM) values set by endpoints at the access edge or via deep packet inspection (DPI) technology—such as Cisco's application visibility and control (AVC)—on the wireless LAN controller (WLC). The solution has been recently upgraded with several features detailed in Chapters 18, "Wireless LAN QoS Considerations and Recommendations," and 19, "Wireless LAN Controller QoS Design," but the overall architecture has remained the same.

This chapter defines the beginning of a new paradigm shift for wired and wireless applications, termed *converged access*. Though this marketing term appears to lend itself to the access PIN (place in the network), in reality, the converged access solution covers the Catalyst 3850 access switch and the Cisco 5760 WLC, which were released by Cisco in early 2013. The designs discussed within will apply equally to the 3650 and 3850.

The current CUWN centralized controller architecture is supported in the same manner using the Cisco 5760, with some differences that are discussed in detail later in this chapter. The primary difference is the quality of service (QoS) configuration. For instance, in the centralized controller mode, the Cisco 5760 offers fully compliant Modular Quality of Service command-line interface (MQC) QoS-based provisioning. In the same fashion as the WLC 5508, the Cisco 5760 provisions QoS policy per service set identifier (SSID) and the upstream policy is pushed to the APs over a Control and Provisioning of Wireless Access Points (CAPWAP) protocol tunnel negotiated between the controller and APs. It is up to the AP to enforce the upstream re-marking of the policies applied to it. From a QoS standpoint, this model is similar to the WLC 5508, although the QoS on the Cisco 5760 is configured via CLI in an MQC fashion.

The converged access architecture also offers a distributed mode where the Catalyst 3850 plays a major role. As part of the distributed architecture, the Catalyst 3850 provides visibility into applications via access control lists (ACLs) and other typical classification techniques. It does this by terminating the CAPWAP tunnel at the access edge, thereby allowing the switch to view the Layer 2 through Layer 7 packet information. Because of this, the Catalyst 3850 can establish a trust boundary for wired and wireless traffic just like the Catalyst 3750 does for wired traffic.

This new distributed architecture, illustrated in Figure 20-1, offers mobility by way of the Cisco 5760 controller in conjunction with the Catalyst 3850s at the edge. In this scenario, the Catalyst 3850s can be put into switch peer groups (SPGs). SPGs are just a group of switches that provide local roaming between clients, as opposed to the previous CUWN architecture where traffic was always terminated at the Cisco 5508 WLC. Each of the Catalyst 3850s or Catalyst 3850 switch stacks is defined as a mobility agent (MA). Within a single SPG, Catalyst 3850s form a full mesh of CAPWAP tunnels between MAs. Inside the SPG, two essential terms are used to explain client roaming:

- Point of presence (PoP), defined as the point in the network where the wired infrastructure first sees the wireless traffic
- Point of attachment (PoA) defined as the AP to which the user joins or roams

When a client roams between switches within a single SPG, traffic is sent from the PoA (also referred to as the foreign controller) back to the PoP (also referred to as the anchor controller) via the CAPWAP tunnel. The original switch being the PoP then sends traffic to its destination. However, if roaming occurs between SPGs, traffic will run through the Cisco 5760 controller, defined as the mobility controller (MC), as it does in the CUWN with a 5508 WLC today. With that in mind, having a large SPG may benefit the architecture so as to limit the instances when traffic must be sent to the Cisco 5760 controller.

Note In large deployments, the Cisco 5760 is positioned as the MC, but the Catalyst 3850 can be configured as the MC as well.

Note The network administrator should balance the desired size of the SPG with the number of tunnels resulting from the full mesh within the SPG, noting a more prudent approach to the design may be to consolidate switches in the network that will have several roaming clients. Examples of this might be a single floor of a building or smaller deployments in a single building.

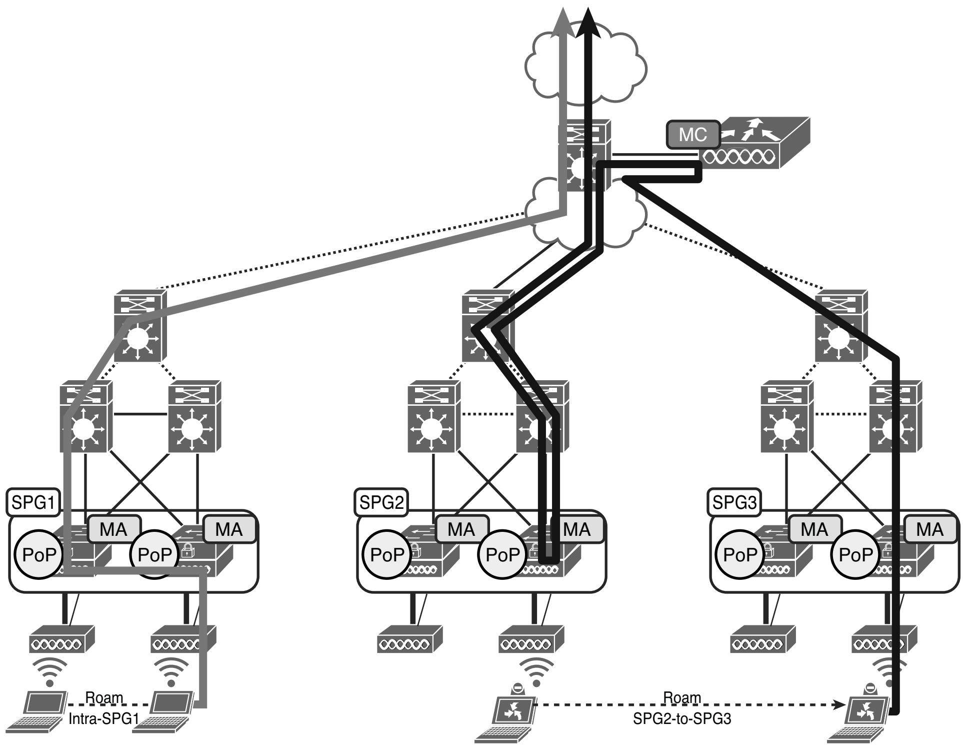


Figure 20-1 *Converged Access Architecture*

One of the major highlights of this new architecture is that it offers alignment between wired and wireless QoS classification/marking/policing polices used by other MQC-based platforms (Catalyst 4500 Sup 6/7, Catalyst 6500 Sup2T, ISRG2, ASR1000). Both the Cisco 5760 WLC and the Catalyst 3850 perform QoS in the same manner. Because they reside in different PINs, it is safe to say they may have different QoS roles. Chapter 21, "Converged Access QoS Design Case Study," highlights the role differences with detailed configurations.

Note QoS configuration is the same between the Cisco 5760 and the Catalyst 3850, so this chapter focuses on QoS in general and may describe the Catalyst 3850 in terms of configuration, but the Cisco 5760 configuration is just the same.

Note WMM User Priority (UP) markings are based on the network requirements of various types of traffic. They do not take into account customer business priorities, nor do they specify which applications on a particular mobile device should receive a particular marking. It is up to a combination of the application vendor, the vendor of the mobile device OS, and the mobile device hardware vendor to jointly ensure QoS markings are implemented correctly over the wireless medium. Hence, relying on WMM as a traffic classification method may sometimes lead to mismarked differentiated service code point (DSCP) values and thus mis-queued traffic. Keep in mind that the 5508 WLC relies on WMM for classification. Converged access controllers provide more granular capabilities at the access edge, which in some cases can resolve issues with mis-queued traffic.

Converged Access

The QoS role of the campus access switch is to differentiate and manage traffic at the edge and to manage packet loss. As discussed in Chapter 14, "Campus Access (Cisco Catalyst 3750) QoS Design," the role of the Cisco Catalyst 3750 is as an access switch focusing on wired access clients. This design chapter extends the access switch role by including both wired and wireless clients as part of converged access. Figure 20-2 illustrates the port-specific QoS roles of a converged campus access switch.

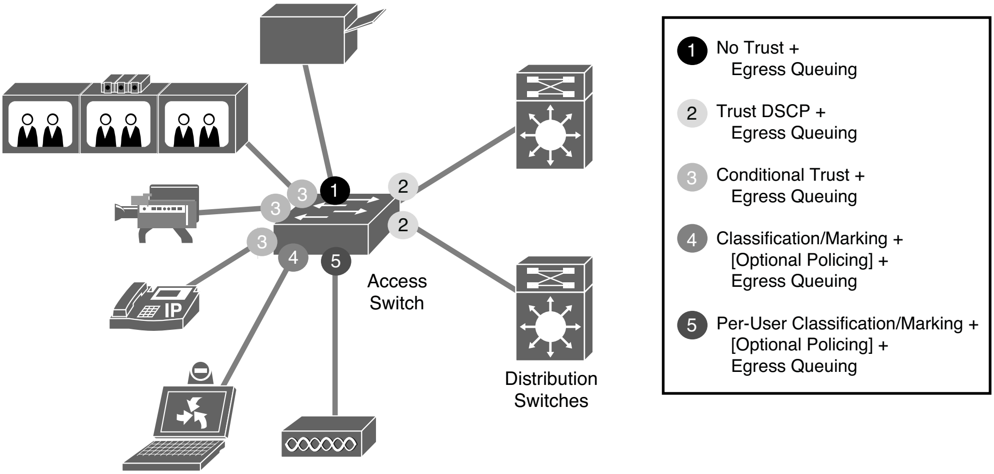


Figure 20-2 *Converged Access Switch Port QoS Roles*

Converged access surpasses the functionality of the Cisco Catalyst 3750 by adding wireless port roles into the equation. It promotes the unification of wired and wireless QoS by establishing a parallel between wired and wireless client roles. However, these lines are blurred when configuring the targets specific to either media (that is, SSID polices for wireless clients and port-based polices for wired clients).

The section that follows begins with a review of the platform's architecture, specifically as it relates to QoS.

Cisco Catalyst 3850 QoS Architecture

From a QoS perspective, the Cisco Catalyst 3850 (hereafter referred to simply as the Catalyst 3850) is a new breed of access switch. Though it resides in the 2k/3k family, it provides support for wireless clients via an internal wireless control module (referred to as WCM). The hardware architecture is similar to the Cisco Catalyst 3750 in that the Catalyst 3850 is stackable and provides a shared buffer pool. However, several key differences exist between these platforms, as summarized in Table 20-1.

Table 20-1 *Cisco Catalyst 3750-X, 3850:Major Feature and Functionality Matrix*

Switch	Catalyst 3750-X	Catalyst 3850
Layer 2/Layer 3 multilayer switch	Yes	Yes
Stackable?	Yes	Yes
Stacking technology	StackWise Plus	Ultra-StackWise
Total switching capacity	64 Gbps	480 Gbps
Wireless capable	No	Yes
Queue count	4 Queues	8 wired/4 wireless
Provisioning language	MLS	MQC

Figure 20-3 and Figure 20-4 illustrate the QoS architecture of this platform, which is split into two sections: wired-to-wireless and wireless-to-wired, respectively. Because of the inherent differences between wireless and wired technology, different touch points within QoS are highlighted in both models. In this section, keep in mind that each touch point mentioned might not be necessary, but it is important that all options be defined for a better understanding of the architecture.

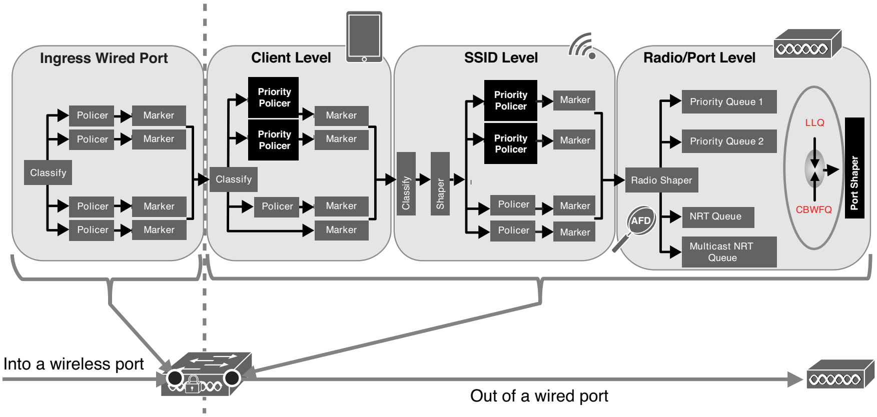


Figure 20-3 *Cisco Catalyst 3850 QoS Architectural Model: Wired to Wireless*

Traffic traveling from a wired-to-wireless port (connected to an AP) is explicitly classified based on class maps. Marking or policing policies can be applied to physical switch ports or switch virtual interfaces (SVIs), but at the time of this writing, per-port/per-VLAN policies are not supported. Ingress queuing, provided on the Catalyst 3750, is not configurable at this time on the Catalyst 3850 because of the increased size of the internal stack ring.

As traffic travels out of the wireless port, there are several QoS touch points to consider. A wireless port is defined as any port directly attached to an AP. The client-level traffic is classified on egress using class maps and provides strict-priority queuing for two separate traffic types. Egress marking and policing of priority and default traffic is also an option per client. This priority traffic bypasses any shaping and is sent directly to the physical port for scheduling.

SSID-level traffic is also classified on egress using class maps. In addition to policing and marking at this level, there is a shape command to limit the rate of traffic at the SSID per radio (**bssid**). Though identified as a shaper, this entity is set with a queue limit of 0, signifying that it does not buffer traffic and therefore is used only to limit traffic at the SSID level. A bandwidth for the SSID can also be configured to provide a ratio limit between SSIDs sharing the same radio. This helps in scenarios where multiple users on a guest SSID consume a majority of radio bandwidth while business users associated to another SSID on the same radio are stifled.

Note One difference between how the Catalyst 3850 and Cisco 5760 are configured is determined by the way the Cisco 5760 is positioned. If the Cisco 5760 is configured as a centralized controller like the Cisco 5508 wireless controller, the QoS policy at the SSID level is the one generally configured and pushed to the AP. This is similar to the configuration of the Cisco 5508. If, however, the Cisco 5760 is used in conjunction with the Catalyst 3850 as described earlier, the Cisco 5760 QoS policies at the SSID level are not be pushed to the APs. Instead, the Catalyst 3850 configuration is used to manage the QoS at the access edge.

Note SSID-level priority policers are performed on unicast priority traffic only. Physical-level priority policers are for multicast priority traffic only.

At the time of this writing, the radio level is not configurable. However, it does require explanation. The radio level shapes egress traffic toward the wireless port. The radio negotiates max rate with the associated AP to limit the amount of traffic being sent to the AP. Each AP negotiates these rates for every radio inside of the AP, and the subsequent policy is applied to all wireless ports by the WCM.

Port-level traffic is subject to four egress queues, two of which are strict priority. The non-real-time queue is effectively the default class, and the multicast-non-real-time queue is used for all non-real-time multicast traffic. Classification based on class maps and policing is also provided at this level. A port shaper that is nonconfigurable is generated based on the radio-level shaper negotiation. This shaper is there to provide a max limit to the amount of traffic the AP itself receives. After the APs negotiate the rate, the port shaper is set as the aggregate of all radio rates and the WCM installs the policy to the wireless interface.

The queuing scheduler is class-based weighted fair with priority queuing, and the bandwidth management algorithm is approximate fair drop (AFD), which provides fairness between users, both of which are discussed in more detail later in this chapter.

As illustrated in Figure 20-4, traffic in the opposite direction, from a wireless to a wired port, maintains the same basic structure.

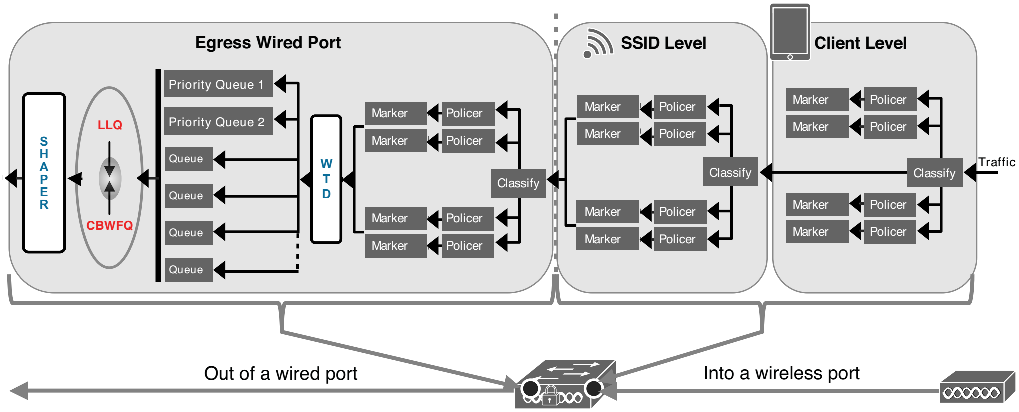


Figure 20-4 *Cisco Catalyst 3850 QoS Architectural Model: Wireless to Wired*

Traffic is classified on ingress for traffic from a wireless client through the AP into the Catalyst 3850. Marking or policing policies can be applied to individual clients or at the SSID as an aggregate.

Traffic classified on ingress can also be policed or marked at the SSID level. This level is not chained to the client level at the time of this writing and should be considered similar to VLAN-based polices. You may decide to provide both client-level and SSID-level policies to establish a cap for both user and group.

Note Chaining is defined as providing an overriding or aggregation point. For example, policers at the client level are not able to overrun the aggregate policer at the SSID level. If a single client policer is set to 12 Mbps and the SSID policer is set to 10 Mbps, the client policer allows the 12-Mbps rate. With chaining, the SSID level policer drops the additional 2 Mbps after the client level passes the 12 Mbps of traffic. Likewise, if marking is performed at the client and the SSID level, it provides precedence to the SSID level.

As traffic leaves the Catalyst 3850 out a wired port, classification is again accomplished via class maps. Marking and policing policies can be configured on the physical switch ports or on the SVIs.

The queuing scheduler is class-based weighted fair queuing (CBWFQ) and dual low-latency queuing (LLQ), and the dropping algorithm is weighted tail drop (WTD).

Egress hierarchical shaping is configurable to shape the port to a specific rate.

QoS Design Steps

QoS configuration on Cisco Catalyst 3850 series switch supports several device types, including both wired and wireless. Because of this versatile support, features like conditional trust are limited to specific devices (Cisco IP phones, Digital Media Players, TelePresence Systems, and so on), which are generally connected to a wired port. Also, egress queuing for the wireless port type is aligned with the WMM specification to include four queues, which differs from the wired port type aligned with other campus switches. These considerations have been taken into account when discussing the following design steps:

1. Configure ingress QoS models, including the following:
   - Wired ports or wireless clients: service policy models
   - Wired ports only: conditional trust
2. Configure egress queuing.

Each of these design steps is covered in turn.

Enabling QoS

While the Catalyst 3750 is an Multi-Layer Switch (MLS)-based QoS platform, it is worth noting at the start that the Catalyst 3850 is purely MQC based. This means that QoS is enabled by default and any QoS markings are sent through the platform untouched. At the time of this writing, one exception applies to this rule on the Catalyst 3850. If traffic passes from a wireless-to-wired port, or vice versa, the QoS values are re-marked to default (0). However, this is not the case for wired-to-wired traffic. This restriction can be lifted by disabling the default **untrust** command in the global configuration, thereby aligning the trust behavior of the Catalyst 3850 with that of other MQC-based devices. This is important to note because QoS marking policies will be overwritten by this boundary. Example 20-1 shows what is required at the SSID level to circumvent the untrusted behavior and what will be required in the future to provide trust alignment.

Example 20-1 *Enabling Trusted Behavior on the Catalyst 3850*

```
 ! This section enables default DSCP-to-DSCP copying
C3850(config)# table-map dscp2dscp
C3850(config-tablemap)# default copy
 ! Incoming DSCP values will be preserved 1:1

 ! This section enables default DSCP-to-UP copying
C3850(config)# table-map dscp2up
C3850(config-tablemap)# default copy
 ! DSCP values will be mapped to WMM UP values

 ! This section configures the downstream trust policy map
C3850(config)# policy-map TRUST-DOWN
C3850(config-pmap)# class class-default
C3850(config-pmap-c)#  set dscp dscp table dscp2dscp
 ! DSCP values will be preserved in the downstream policy
C3850(config-pmap-c)#  set wlan user-priority dscp table dscp2up
 ! WMM UP values will be set based on DSCP in the downstream policy

 ! This section configures the upstream trust policy map
C3850(config)# policy-map TRUST-UP
C3850(config-pmap)# class class-default
C3850(config-pmap-c)#  set wlan user-priority dscp table dscp2up
 ! WMM UP values will be set based on DSCP in the upstream policy

 ! This section attaches the policies to the WLAN
C3850(config)# wlan BRILEY-1
C3850(config-wlan)# service-policy out TRUST-UP
 ! Applies upstream trust policy to the WLAN
C3850(config-wlan)# service-policy out TRUST-DOWN
 ! Applies downstream trust policy to the WLAN
```

To align the trust behavior of the Catalyst 3850 to other MQC-based platforms the global configuration in Example 20-2 is required.

Example 20-2 *Enabling Trusted Behavior on the Catalyst 3850*

```
 ! This section enables global trust

C3850(config)# no qos wireless-default-untrust
```

You can verify these configurations with the following commands:

- **show run**
- **show policy-map interface wireless** (as shown in Example 20-3)

Note Only one example of a given verification command is usually presented (to minimize redundancy).

Example 20-3 *Verifying Applied Policy on Catalyst 3850:* ***show policy-map interface***

```
C3850# show policy-map interface wireless ssid name BRILEY-1

Remote SSID BRILEY-1 iifid: 0x0100B9C00000003B.0x00E2F44000000006.0x00F6208000000009

  Service-policy input: TRUST-UP

    Class-map: class-default (match-any)
      Match: any
      QoS Set
        dscp dscp table dscp2dscp
```

Ingress QoS Models

Ingress QoS models include the following:

- Wired-only Conditional Trust Model
- Wired port and Wireless Client Models
  - Classification and Marking Model
  - Optional Policing Model

Each of these ingress QoS models is discussed in turn.

Wired-Only Conditional Trust Model

There is no concept of explicit trust on MQC-based platforms like the Catalyst 3850 (because trust is implicit). However, this term is used loosely to indicate the classification and marking of QoS values similar to how trust was leveraged on classic MLS-based platforms. Conditional trust itself allows for dynamic verification of the following types of devices:

- **cisco-phone** to conditionally trust Cisco IP phones
- **cts** to conditionally trust Cisco TelePresence Systems
- **ip-camera** to conditionally trust Cisco IP video surveillance cameras
- **media-player** to conditionally trust Cisco Digital Media Players

Of these devices, all are verified via Cisco Discovery Protocol (CDP) to extend trust to the device. Conditional trust can be configured with the interface command **trust device**, which can be configured using the **cisco-phone**, **cts**, and other keywords from the preceding list to extend trust after each device has been verified via a CDP negotiation. Cisco phones do not re-mark DSCP values of traffic received from devices attached to the IP phones and therefore trusting class of service (CoS) is recommended. Devices that do not have this limitation should trust DSCP. The type of trust must be specified via a MQC-based policy, as illustrated in the configuration in Example 20-4. Example 20-5 shows verification of the conditional trust state.

Note The concept of extended trust is not provided to wireless devices at this time.

Example 20-4 *Enabling Conditional (CoS-Mode) Trust on the Catalyst 3850*

```
 ! This section configures the class maps
C3850(config-cmap)# class-map match-all VOICE
C3850(config-cmap)#  match cos 5
C3850(config-cmap)# class-map match-all SIGNALING
C3850(config-cmap)#  match cos 3


 ! This section defines the CoS-to-DSCP remarking policy map
C3850(config-cmap)# policy-map CISCO-IPPHONE
C3850(config-pmap)#  class VOICE
C3850(config-pmap-c)# set dscp ef
 ! Maps CoS 5 to DSCP EF
C3850(config-pmap-c)# class SIGNALING
C3850(config-pmap-c)#  set dscp cs3
 ! Maps CoS 3 to DSCP CS3
C3850(config-pmap-c)# class class-default
C3850(config-pmap-c)#  set dscp default

 ! This section attaches the policy to the interface(s)
C3850(config)# interface GigabitEthernet 1/0/13
C3850(config-if)# trust device cisco-phone
 ! This provides CDP-negotiation of device in question
C3850(config-if)# service-policy input CISCO-IPPHONE
 ! Attaches the CoS-to-DSCP mapping policy-map
```

Example 20-5 *Verifying Conditional Trust on Catalyst 3850:* ***show platform qos advanced qsb gigabitEthernet*** *1/0/13*

```
C3850# show platform qos advanced qsb gigabitEthernet 1/0/13
QoS Subblock contents...
def_cos = 0, def_cos_enable = 0, def_qos_label = 0
def_priority = 13, trust_enabled = 1
input_trans_index = 0, output_trans_index = 0
def_queuing = 1, trust_type = TRUST_DSCP, client_handle = 0
policy_trust = TRUST_DSCP
...
```

Classification and Marking Models

In many scenarios, trust models might not be available or sufficient to distinctly classify all types of traffic required by the end-to-end QoS strategic model. Therefore, explicit classification and marking policies may be needed at the access edge.

One goal of converged access is to simplify QoS deployments by aligning the CLI between wireless and wired clients. Wired and wireless clients can share basic ingress classification and marking policies, making deployments easier and management of polices less complex. To simplify, a configuration example provided to both wired and wireless clients based on Figure 11-5 (an eight-class QoS model) is shown in Example 20-6.

Note As discussed previously, not all application classes may be present at the access edge on ingress. For example, streaming video would likely not be present at the access edge on ingress (because these flows are not *sourced* from campus endpoints, but are likely *destined* to them), nor are network control flows sourced from campus endpoints. Therefore, these classes do not need to be included in the access-edge classification and marking policy map.

Note Referenced access lists are omitted from the policy examples for brevity.

Example 20-6 *Classification and Marking Policy Example on a Catalyst 3850*

```
! This section configures the class maps
C3850(config-cmap)# class-map match-all VOICE
C3850(config-cmap)#  match dscp ef
 ! Voice is matched on DSCP EF
C3850(config-cmap)# class-map match-all INTERACTIVE-VIDEO
C3850(config-cmap)#  match access-group name INTERACTIVE-VIDEO
```

```
 ! Associates INTERACTIVE-VIDEO access-list with class map
C3850(config-cmap)# class-map match-all SIGNALING
C3850(config-cmap)#  match dscp cs3
 ! Signaling is matched on DSCP CS3
C3850(config-cmap)# class-map match-all TRANSACTIONAL-DATA
C3850(config-cmap)#  match access-group name TRANSACTIONAL-DATA
 ! Associates TRANSACTIONAL-DATA access-list with class map
C3850(config-cmap)# class-map match-all SCAVENGER
C3850(config-cmap)#  match access-group name SCAVENGER
 ! Associates SCAVENGER access-list with class map


 ! This section configures the ingress marking policy map
C3850(config-cmap)# policy-map MARKING
C3850(config-pmap)# class VOICE
C3850(config-pmap-c)#  set dscp ef
 ! VoIP is marked EF
C3850(config-pmap-c)# class SIGNALING
C3850(config-pmap-c)#  set dscp cs3
 ! Signaling is marked CS3
C3850(config-pmap-c)# class INTERACTIVE-VIDEO
C3850(config-pmap-c)#  set dscp af41
 ! Interactive-Video is marked AF41
C3850(config-pmap-c)# class TRANSACTIONAL-DATA
C3850(config-pmap-c)#  set dscp af21
 ! Transactional Data is marked AF21
C3850(config-pmap-c)# class SCAVENGER
C3850(config-pmap-c)#  set dscp cs1
 ! Scavenger traffic is marked CS1
C3850(config-pmap-c)# class class-default
C3850(config-pmap-c)#  set dscp default


 ! This section attaches the service policy to the wired interface(s)
C3850(config)# interface range GigabitEthernet 1/0/1-48
C3850(config-if-range)# switchport access vlan 10
C3850(config-if-range)# switchport voice vlan 110
C3850(config-if-range)# spanning-tree portfast
C3850(config-if-range)# service-policy input MARKING
 ! Attaches the per-port marking policy to the wired interface(s)


 ! This section attaches the service policy to the wireless client(s)
C3850(config)# wlan BRILEY-1
C3850(config-wlan)# service-policy client input MARKING
```

The wireless portion of the Catalyst 3850 has the ability to attach a policy per client with the **client** keyword under the WLAN SSID. When the **client** keyword is used, as in Example 20-6, it is applied to each wireless client authorized into the SSID and is applied independently to each of the clients. When using the **service-policy** statement without this keyword, the policy applies to the SSID and treats all clients as an aggregate.

When a policy is applied to the SSID, it takes effect on the BSSID, meaning per AP or per SSID. For example, if the same SSID is used on multiple APs attached to a Catalyst 3850, the policy is applied to each AP/SSID pair separately.

Note Classification and marking models may be deployed on a per-port basis or on a per-VLAN basis for wired ports, per-client or per-SSID for wireless ports (as discussed in Chapter 21).

You can verify this configuration with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**
- **show policy-map interface wireless ssid**
- **show policy-map interface wireless client**
- **show platform qos dscp-cos counters**

Note The **show policy-map interface** command on the Catalyst 3850 does not show classification counters (like the corresponding command on other devices does). The **show platform qos dscp-cos counters** command though long and unformatted provides similar statistics as the **show mls qos interface statistics** command on the Catalyst 3750. The output of these commands is not shown for brevity.

Classification, Marking, and Policing Model

Optionally, policing policies may also be effectively deployed at the access edge of the campus to manage application traffic and to drop unwanted traffic.

In Example 20-7, a 12-class strategic QoS policy (based on Figure 11-7) is adapted to the converged access edge (application classes that are not present at the campus access edge, such as network control and streaming video are pruned from the policy). Also, a distinction is made for some flows sourced from the voice VLAN versus the data VLAN. This distinction is only applicable to the wired ports and not wireless ports, because there may be no separate voice and data VLAN associated to the wireless clients. The same class map names are used to demonstrate the ability to apply the same policy maps and class maps for both wired and wireless devices.

In this example, VoIP and signaling traffic from the voice VLAN can be policed to drop at 128 Kbps and 32 Kbps, respectively (as any excessive traffic matching this criterion would be indicative of network abuse). Similarly, multimedia conferencing, signaling, and scavenger traffic from the data VLAN can be policed to drop. In contrast, data plane policing policies can be applied to transactional, bulk, and best effort data traffic, such that these flows are subject to being re-marked (but not dropped at the ingress edge) when severely out of profile. Re-marking is performed by configuring a table map with the global configuration command **table-map MARKDOWN-TABLE**, which specifies from/to DSCP values to be re-marked for out-of-profile traffic (which in the case of Data Plane Policing/Scavenger class QoS policies the value is CS1/DSCP 8).

Note Referenced class maps are omitted from the policy example for brevity and to minimize redundancy. Also recognize that VVLAN-* class maps only classify based on DSCP values, not access lists as the others do. These class maps are used for both wired and wireless to limit the number of class maps required, though the voice and data VLAN separation for wireless might not exist.

Example 20-7 *Classification, Marking, and Policing Policy Example on a Catalyst 3850*

```
 ! This section configures the global DSCP markdown table map
C3850(config)# table-map MARKDOWN-TABLE
C3850(config-tablemap)# map from 0 to 8
C3850(config-tablemap)# map from 10 to 8
C3850(config-tablemap)# map from 18 to 8



 ! This section configures the policing policy map
C3850(config)# policy-map POLICING
C3850(config-pmap)# class VVLAN-VOIP
C3850(config-pmap-c)# set dscp ef
C3850(config-pmap-c)# police 128k conform-action transmit exceed-action drop
 ! VoIP is marked EF and policed to drop at 128 Kbps
C3850(config-pmap-c)# class VVLAN-SIGNALING
C3850(config-pmap-c)# set dscp cs3
C3850(config-pmap-c)# police 32k conform-action transmit exceed-action drop
 ! (VVLAN) Signaling is marked CS3 and policed to drop at 32 Kbps
C3850(config-pmap-c)# class MULTIMEDIA-CONFERENCING
C3850(config-pmap-c)# set dscp af41
C3850(config-pmap-c)# police 5m conform-action transmit exceed-action drop
 ! Multimedia-conferencing is marked AF41 and policed to drop at 5 Mbps
C3850(config-pmap-c)# class SIGNALING
```

```
C3850(config-pmap-c)# set dscp cs3
C3850(config-pmap-c)# police 32k conform-action transmit  exceed-action drop
 ! (DVLAN) Signaling is marked CS3 and policed to drop at 32 Kbps
C3850(config-pmap-c)# class TRANSACTIONAL-DATA
C3850(config-pmap-c)# set dscp af21
C3850(config-pmap-c)# police 10m conform-action transmit  exceed-action set-dscp-
transmit dscp table MARKDOWN-TABLE
 ! Trans-data is marked AF21 and policed to remark (to CS1) at 10 Mbps
C3850(config-pmap-c)# class BULK-DATA
C3850(config-pmap-c)# set dscp af11
C3850(config-pmap-c)# police 10m conform-action transmit  exceed-action set-dscp-
transmit dscp table MARKDOWN-TABLE
 ! Bulk-data is marked AF11 and policed to remark (to CS1) at 10 Mbps
C3850(config-pmap-c)# class SCAVENGER
C3850(config-pmap-c)# set dscp cs1
C3850(config-pmap-c)# police 10m conform-action transmit  exceed-action drop
  ! Scavenger traffic is marked CS1 and policed to drop at 10 Mbps
C3850(config-pmap-c)# class class-default
C3850(config-pmap-c)# set dscp default
C3850(config-pmap-c)# police 10m conform-action transmit  exceed-action set-dscp-
transmit dscp table MARKDOWN-TABLE
  ! All other traffic is marked 0 and policed to drop at 10 Mbps


 ! This section attaches the service policy to the wired interface(s)
C3850(config)# interface range GigabitEthernet 1/0/1-48
C3850(config-if-range)# switchport access vlan 10
C3850(config-if-range)# switchport voice vlan 110
C3850(config-if-range)# spanning-tree portfast
C3850(config-if-range)# service-policy input POLICING
 ! Attaches the per-port policing policy to the wired interface(s)


 ! This section attaches the service-policy to the wireless client(s)
C3850(config)# wlan BRILEY-1
C3850(config-wlan)# service-policy client input POLICING
```

Note The Catalyst 3850 IOS Software allows for policing rates to be entered using the postfixes **k** (for kilobits), **m** (for megabits), and **g** (for gigabits), as shown in Example 20-7. In addition, decimal points are allowed in conjunction with these postfixes. For example, a rate of 10.5 Mbps could be entered with the policy map command **police 10.5m**. Although these policing rates are converted to their full bits per second values within the configuration, it makes the entering of these rates more user friendly and less error prone (as could easily be the case when having to enter up to 10 zeros to define the policing rate).

Note Although technically it is possible to apply an aggregate policing policy such as this on a per-VLAN basis, it is not advisable to do so. This is because the number of endpoints in a given VLAN can dynamically vary, yet the policing rates are statically fixed at an aggregate level, resulting in unpredictable bandwidth allotments per endpoint. With that said, in the instance of guest access, per-VLAN policing can help to aggregate and limit the influence of this traffic on the shared corporate WAN. In this case, protection is focused on the enterprise traffic and not the individual guest traffic.

Note Per-client wireless policies may require tighter restrictions on policing because of the nature of the available bandwidth. Also consider using aggregate-based policies on the SSID in cases where per-client policies are not necessary (guest access). Though the number of wireless clients may vary as clients within a VLAN do, the wireless medium justifies an aggregate SSID policy because it is paired with the per-user fairness algorithm AFD, which is explained in a later section, "Wireless Queuing."

You can verify the configuration in Example 20-7 with the following commands:

- **show class-map**
- **show policy-map**
- **show table-map** (as shown in Example 20-8)
- **show platform qos policy target** (as shown in Example 20-9)
- **show platform qos policy hw_state target** (as shown in Example 20-10)
- **show policy-map interface brief** (as shown in Example 20-11)
- **show policy-map interface** (as shown in Example 20-12)

Example 20-8 *Verifying Global Policing Markdown Table Maps on a Catalyst 3850:* ***show table-map***

```
C3850# show table-map
 Table Map MARKDOWN-TABLE
    from 0 to 8
    from 10 to 8
    from 18 to 8
    default copy
```

In Example 20-8, the markdown table is shown. The **from** value describes the incoming value, and the **to** value represents the markdown. **default copy** represents trusting or leaving the ingress marking in the packet without modification. Table maps at the SSID level are used exclusively as the markdown tool to re-mark traffic; other levels provide the set command as well.

Example 20-9 *Verifying Interface Policers on a Catalyst 3850:* ***show platform qos policy target gigabitEthernet 1/0/10***

```
C3850# show platform qos policy target gi 1/0/10
Input policy :
--------------
POLICY: POLICING Num Classes:8 Targets:0
...
  Target:Gi1/0/10 DIR:IN STATE:SET IN HW LE_LABEL:2, client_policy_type:LE
    IF:0x010351800000001f TCCGs:8 TCG:0x4a54adfc Next:(nil) Prev:(nil)
    CLASS: VVLAN-VOIP
      Action flags:0x5
      AG Policer Action Num:0
      Rate:128000 Peak:0
      Limit:0 ExtLimit:0
      BC:4000 Peak:0
      Color Mode:Color Blind Type:1R2C
      Mark DSCP Action:46
    CLASS: VVLAN-SIGNALING
      Action flags:0x5
      AG Policer Action Num:1
      Rate:32000 Peak:0
      Limit:0 ExtLimit:0
      BC:1500 Peak:0
      Color Mode:Color Blind Type:1R2C
      Mark DSCP Action:24
    CLASS: MULTIMEDIA-CONFERENCING
      Action flags:0x5
      AG Policer Action Num:2
      Rate:5000000 Peak:0
      Limit:0 ExtLimit:0
      BC:156250 Peak:0
      Color Mode:Color Blind Type:1R2C
      Mark DSCP Action:34
...

Additional details for policy: POLICING
        Number of targets attached to : 1
        Number of targets policy successfully programmed in HW: 1
```

In Example 20-9, the policy is shown in greater detail. Important to note are the number of targets to which the policy is attached and if this policy was indeed programmed in hardware correctly.

Example 20-10 *Verifying Policy Map Installed in Hardware on a Catalyst 3850:* ***show platform qos policy hw_state target gigabitEthernet 1/0/10***

```
C3850# show platform qos policy hw_state target gi 1/0/10
Input policy :POLICING
H/W programming State:INSTALLED IN HW
Output policy :Not attached
```

In Example 20-10, the QoS policy for interface GigabitEthernet 1/0/10 is shown as installed in hardware. In certain instances, this policy may not be specifically installed (for example, if the incorrect number of classes is applied or if the marking method is used incorrectly). In these cases, you will receive an information message, but it is always preferable to validate that policies are installed with this command.

Example 20-11 *Verifying Policy Map Attachment to Interfaces on a Catalyst 3850:* ***show policy-map interface brief***

```
C3850# show policy-map interface brief
Service-policy output: def-11gn
Radio dot11b iifid: 0x0100B9C00000003B.0x00E2F44000000006
...
Service-policy input: POLICING
 GigabitEthernet1/0/10
```

In Example 20-11, the policy is shown to be attached to specific ports. This is a simple way to reference which policy is attached to multiple ports before troubleshooting QoS or validating that the policy is actually attached.

Example 20-12 *Verifying policy-map configuration on a Catalyst 3850:* ***show policy-map interface***

```
C3850# show policy-map interface gi 1/0/10
 GigabitEthernet1/0/10

  Service-policy input: POLICING

    Class-map: VVLAN-VOIP (match-any)
      Match: ip dscp ef (46)
      QoS Set
        dscp ef
      police:
          cir 128000 bps, bc 4000 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          drop
        conformed 0000 bps, exceed 0000 bps

    Class-map: VVLAN-SIGNALING (match-any)
      Match: ip dscp cs3 (24)
      QoS Set
        dscp cs3
      police:
          cir 32000 bps, bc 1500 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          drop
        conformed 0000 bps, exceed 0000 bps
```

Example 20-12 shows the QoS policy for interface GigabitEthernet 1/0/10, but note that classification counters are not displayed. Policing statistics are the only statistics available on ingress of the Catalyst 3850. Also note the additional keywords for **wireless** and **ssid** under **show policy-map interface**. These keywords as illustrated will show the attached policy to individual wireless clients by MAC address or the SSID specified.

Queuing Models

Because of the nature of converged access, there are separate queuing models for wired and wireless ports. A wireless port, defined as any switch port on the Catalyst 3850 attached to a Cisco AP, provides four independent queues. To align with these queues, a port that is determined to be wireless on the wireless control module internal to the Catalyst 3850 will transition to a four-queue model. Wired ports, in contrast, provide up to eight queues, which tie in well with other campus-based switches and thus make it much easier to align QoS policies. These queuing models are described separately in subsequent sections of the chapter.

Ingress queuing is provided on the internal stack interfaces, but at the time of this writing is not configurable nor is it likely to be necessary. The basic architecture of the platform provides 24 x 1G access ports and 2 x 10G uplinks per application specific integrated circuit (ASIC) to a 120G stack connection. The architecture provides two separate internal queues over the stack ring, giving access to priority traffic and nonpriority traffic.

Wired Queuing

The wired queuing model is an eight-queue structure where two of the queues can be configured as strict. The priority queues are optional, but there are instances where a single or dual priority queue model has its benefits. If alignment is important between queuing models, the Catalyst 4500 and Catalyst 6500 both share a 1P7Q model that easily aligns with the Catalyst 3850 1P7Q3T model. However, this model does not consider separation between voice and video priority traffic. The 2P6Q3T model provides voice and video separation, but reuse of the policy does not align well with the Catalyst 4500.

Priority queuing is defined identically for wired as it was previously in wireless. Of the two queues provided, **priority level 1** is scheduled strictly before **priority level 2**. This simply means that level 1 will always take precedence over level 2 though both are low-latency queues.

Shaping is configurable on egress in a hierarchical fashion to limit the overall bandwidth of a link while providing for traffic granularity. When using shaping at the physical port, the shaped traffic will not use more than the allocated bandwidth, even if the link is idle. Shaping provides a more even flow of traffic over time and reduces the peaks and valleys of bursty traffic. With shaping, either an absolute value or a percentage of each weight is used to compute the bandwidth available for the queues. Shaping is configured with the **shape average** or **shape average percent** command.

With respect to scheduling on the Catalyst 3850, the priority queues are serviced ahead of the other wired queues, which are then scheduled by CBWFQ. The weighted percentages are considered a minimum guarantee for the specific class and can expand above the configured rate in an effort to make use of available bandwidth. These queues are scheduled as a minimum guarantee and can exceed this rate if bandwidth is available.

The Catalyst 3850 also supports the weighted tail drop (WTD) congestion avoidance mechanism. WTD is implemented on queues to manage the queue lengths and to provide drop preferences for different traffic classes. As a packet is enqueued to a particular egress queue, WTD uses the frame's DSCP value to subject it to different drop thresholds. If the threshold is exceeded for a given DSCP value (in other words, the space available in the destination queue is less than the size of the packet), the switch drops the packet. Each queue has three threshold values, and packets are mapped to queues and thresholds on the Catalyst 3850 by standard MQC-based classification. The classification parameters include, but are not limited to, DSCP, CoS, access group, and QoS group. Based on this classification, packets are sent to the correct queue and threshold for treatment.

Ordinarily in each platform-specific design chapter, separate queuing models are presented for the 4-class, 8-class, and 12-class strategic end-to-end QoS models presented in Chapter 11, "QoS Design Principles and Strategies." However, because the Catalyst 3850 provides for wired and wireless queuing models, both of these models are presented as a 12-class model with the option of using either a single or dual priority queue on the wired side.

Wired 1P7Q3T Egress Queuing Model

Egress wired queuing on the Catalyst 3850 family of switches can be configured as 8Q3T, 1P7Q3T, or 2P6Q3T, with either of the latter two being the recommended configuration (because it supports the RFC 3246 EF PHB).

Egress queuing, classification, marking, and other functions on the Catalyst 3850 are provisioned by MQC. This is a significant change from the Catalyst 3750, which used MLS, although a few similarities exist between these switches.

Like the Catalyst 3750, the Catalyst 3850 supports flexible buffer allocations (a.k.a. dynamic thresholding and scaling [DTS]) to hardware queues, which may be dynamically loaned against or borrowed from (as needed). However, provisioning this additional buffer space is quite different. The first difference is that the hardware allocation of buffers is provided only to the priority queues on both the wired and wireless queues. The other queues are allocated buffers on an as-needed basis, making it more flexible within the system to support several queues at once. Each queue (class) can leverage additional buffering capacity from the shared buffer pool based on a **queue-buffers ratio**. This command sets the ratio of buffers received by the class and these buffers are allocated, as previously stated, on an as-needed basis. In general, the max rate for the class is up to four times the configured value. The **queue-buffers ratio** command skews this value based on the overall ratio of buffers provided to the queue. By default, these use the ratio of 1:1:1... for a wired port, where a wireless port is allocated on the order of 10:20:50:20. The recommended buffer allocations for wired interface queues 7 through 1 are 10 percent, 10 percent, 10 percent, 10 percent, 10 percent, 10 percent, and 25 percent, respectively, and the defaults should be used for the wireless queues. Unlike the Catalyst 3750, the Catalyst 3850 does not provide reserve thresholds, and maximum (overload) thresholds are based on the queue buffers' ratio for nonpriority queues.

Once the primary queuing set has been configured for 1P7Q3T egress queuing, WTD thresholds can be defined on Q2 through Q6 to provide intraqueue QoS. Specifically, Q2T1 can be explicitly set at 80 percent queue depth, Q2T2 can be explicitly set at 90 percent queue depth, and Q2T3 can be set at 100 percent queue depth. Note that this example provides threshold settings based on the AF numbering scheme. AFx1 has the lowest drop probability, and AFx3 has the highest drop probability. In addition, the shared buffer is configured using the **queue-buffers ratio** command. This setting allows for most queues to benefit from the extended buffering capabilities of this platform. You might need to take additional consideration with the extended buffering for classes such as Bulk and Scavenger. This buffering will cover the traffic from these classes given they are less-desirable flows and can be held for an extended period of time.

Figure 20-5 shows the 1P7Q3T egress queue mappings for a Catalyst 3850 using an eight-class model.

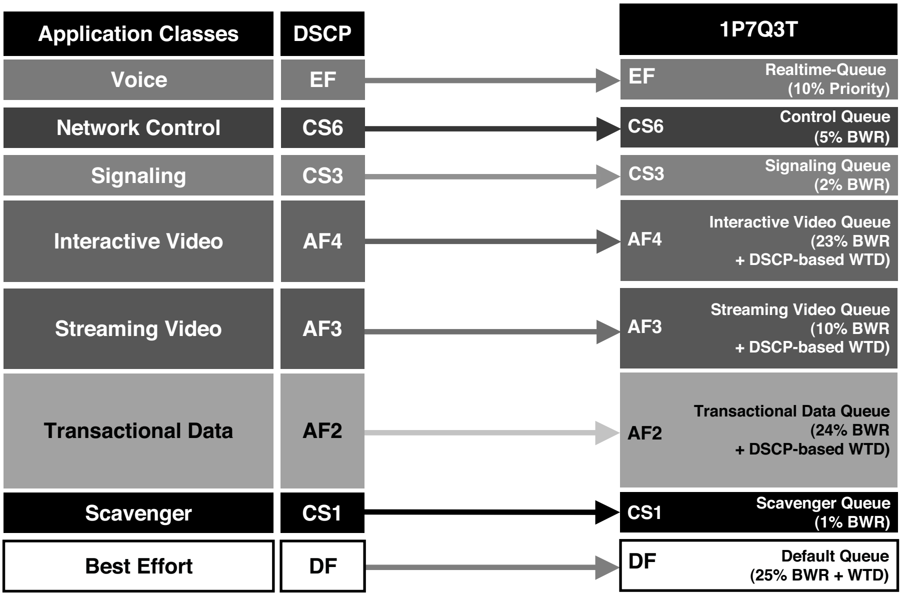


Figure 20-5 *Catalyst 3850 Eight-Class (1P7Q3T) Egress Queuing Model*

Example 20-13 shows the corresponding configuration for an eight-class 1P7Q3T egress queuing on the Catalyst 3850.

Example 20-13 *Catalyst 3850 Eight-Class (1P7Q3T) Egress Queuing Configuration Example*

```
! This section configures buffers, thresholds, and a single priority
 ! queue
C3850(config-pmap-c)# policy-map 1P7Q3T
C3850(config-pmap)# class REALTIME-QUEUE
C3850(config-pmap-c)# priority level 1
C3850(config-pmap-c)# police rate percent 10
 ! REALTIME is enabled as a strict priority queue and policed to drop at
 ! 10 percent of physical rate
C3850(config-pmap-c-police)# class CONTROL-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 5
C3850(config-pmap-c)# queue-buffers ratio 10
 ! CONTROL is allocated 5 percent of bandwidth not allocated to the
 ! priority queue and allowed to consume 10% of additional buffers from
 ! global buffer pool
C3850(config-pmap)# class SIGNALING-QUEUE
C3850(config-pmap-c)#  bandwidth remaining percent 2
 ! SIGNALING is allocated 2 percent of bandwidth not allocated to the
 ! priority queue
```

```
C3850(config-pmap-c)# class INTERACTIVE-VIDEO-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 23
C3850(config-pmap-c)# queue-buffers ratio 10
C3850(config-pmap-c)# queue-limit dscp af43 percent 80
C3850(config-pmap-c)# queue-limit dscp af42 percent 90
C3850(config-pmap-c)# queue-limit dscp af41 percent 100
 ! INTERACTIVE is allocated 23 percent of bandwidth not allocated to the
 ! priority queue, allowed to consume 10% of additional buffers from
 ! global buffer pool, and provided Tail Drop thresholds for 3 AF
 ! classes at rates 80,90,100
C3850(config-pmap-c)# class STREAMING-VIDEO-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 10
C3850(config-pmap-c)# queue-buffers ratio 10
C3850(config-pmap-c)# queue-limit dscp af33 percent 80
C3850(config-pmap-c)# queue-limit dscp af32 percent 90
C3850(config-pmap-c)# queue-limit dscp af31 percent 100
 ! STREAMING is allotted 10 percent of the bandwidth and
 ! allowed to consume 10 percent of additional buffers from the
 ! global buffer pool, and provided Tail Drop thresholds for 3 AF
 ! classes at rates 80,90,100
C3850(config-pmap-c)# class TRANSACTIONAL-DATA-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 24
C3850(config-pmap-c)# queue-buffers ratio 10
C3850(config-pmap-c)# queue-limit dscp af23 percent 80
C3850(config-pmap-c)# queue-limit dscp af22 percent 90
C3850(config-pmap-c)# queue-limit dscp af21 percent 100
 ! TRANSACTIONAL is allotted 24 percent of the bandwidth and
 ! allowed to consume 10 percent of additional buffers from the
 ! global buffer pool, and provided tail drop thresholds for three AF
 ! classes at rates 80,90,100
C3850(config-pmap-c)# class SCAVENGER-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 1
C3850(config-pmap-c)# queue-buffers ratio 10
 ! SCAVENGER is allotted 1 percent of the bandwidth and
 ! allowed to consume 10 percent of additional buffers from the
 ! global buffer pool
C3850(config-pmap-c)# class class-default
C3850(config-pmap-c)# bandwidth remaining percent 25
C3850(config-pmap-c)# queue-buffers ratio 25
 ! class-default is allotted 25 percent of the bandwidth and
 ! allowed to consume 25 percent of additional buffers from the
 ! global buffer pool

 ! This section configures interface egress queuing parameters
C3850(config)# interface gigabitethernet 1/0/13
C3850(config-if)# service-policy out 1P7Q3T
```

You can verify the configuration in Example 20-13 with the following commands (of which sample outputs are provided in the following section):

- **show platform qos hw_state target gigabit x/y**
- **show platform qos policy target gigabit x/y**
- **show policy-map interface**
- **show platform qos queue stats**
- **show platform dscp-cos counters**
- **show policy-map interface wireless**
- **show platform qos queue config**

Wired 2P6Q3T Egress Queuing Model

The 2P6Q3T model only differs slightly from the wired 1P7Q3T egress queuing model as it has been extended to cover a 12-class model. The addition of a second priority queue provides separation for voice and video traffic and still provides for the majority of Assured Forwarding classes. When using this queuing strategy for a 12-class model, the only loss is that of the separate Scavenger class queue, which is now combined with bulk data in the lowest threshold Q1T3 along with AF13.

Figure 20-6 shows these 2P6Q3T egress queue mappings for the Catalyst 3850.

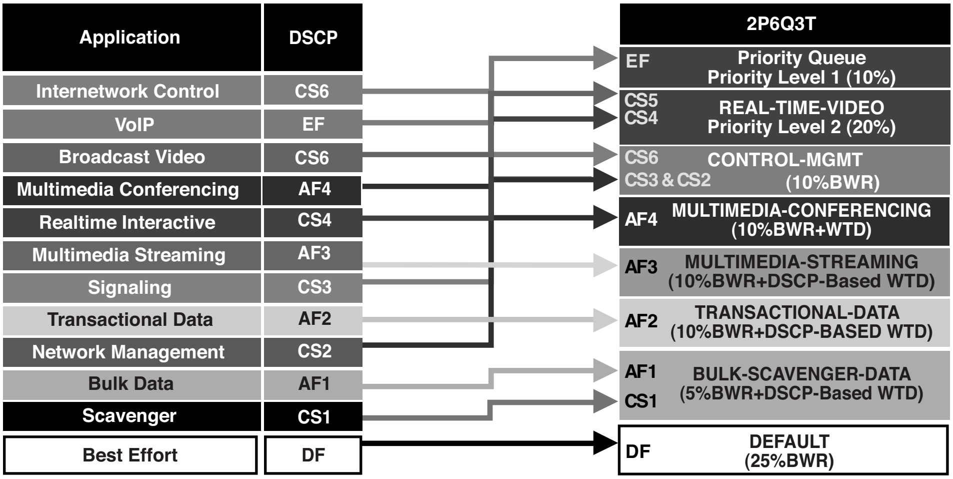


Figure 20-6 *Catalyst 3850 12-Class (2P6Q3T) Egress Queuing Model*

Example 20-14 shows the corresponding configuration for 12-class 2P6Q3T egress queuing on the Catalyst 3850.

Example 20-14 *Catalyst 3850 12-Class (2P6Q3T) Egress Queuing Configuration Example*

```
! This section configures buffers, thresholds and a single PQ
C3850(config-cmap)# policy-map 2P6Q3T
C3850(config-pmap)# class PRIORITY-QUEUE
C3850(config-pmap-c)# priority level 1
C3850(config-pmap-c)# police rate percent 10
! PRIORITY is set to level 1 and serviced before any other class
! It is policed to 10 percent of the physical interface rate
C3850(config-pmap-c-police)# class REAL-TIME-VIDEO-QUEUE
C3850(config-pmap-c)# priority level 2
C3850(config-pmap-c)# police rate percent 20
! REAL-TIME is set to priority level 2 and serviced before any other
! class, it is policed to 20 percent of the physical interface rate
C3850(config-pmap-c-police)# class CONTROL-MGMT-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 10
C3850(config-pmap-c)# queue-buffers ratio 10
! CONTROL is allotted 10 percent of the bandwidth and
! allowed to consume 10 percent of additional buffers from the
! global buffer pool
C3850(config-pmap-c)# class MULTIMEDIA-CONFERENCING-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 10
C3850(config-pmap-c)# queue-buffers ratio 10
C3850(config-pmap-c)# queue-limit dscp af43 percent 80
C3850(config-pmap-c)# queue-limit dscp af42 percent 90
C3850(config-pmap-c)# queue-limit dscp af41 percent 100
! MULTIMEDIA is allotted 10 percent of the bandwidth,
! allowed to consume 10 percent of additional buffers from the
! global buffer pool and provided tail drop thresholds for three AF
! classes at rates 80,90,100
C3850(config-pmap-c)# class MULTIMEDIA-STREAMING-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 10
C3850(config-pmap-c)# queue-buffers ratio 10
C3850(config-pmap-c)# queue-limit dscp af33 percent 80
C3850(config-pmap-c)# queue-limit dscp af32 percent 90
C3850(config-pmap-c)# queue-limit dscp af31 percent 100
! TRANSACTIONAL is allotted 10 percent of the bandwidth,
! allowed to consume 10 percent of additional buffers from the
! global buffer pool, and provided tail drop thresholds for three AF
! classes at rates 80,90,100
C3850(config-pmap-c)# class TRANSACTIONAL-DATA-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 10
C3850(config-pmap-c)# queue-buffers ratio 10
C3850(config-pmap-c)# queue-limit dscp af23 percent 80
```

```
C3850(config-pmap-c)# queue-limit dscp af22 percent 90
C3850(config-pmap-c)# queue-limit dscp af21 percent 100
! TRANSACTIONAL is allotted 10 percent of the bandwidth,
! allowed to consume 10 percent of additional buffers from the
! global buffer pool, and provided tail drop thresholds for three AF
! classes at rates 80,90,100
C3850(config-pmap-c)# class BULK-SCAVENGER-DATA-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 5
C3850(config-pmap-c)# queue-buffers ratio 10
C3850(config-pmap-c)# queue-limit dscp values af13 cs1 percent 80
C3850(config-pmap-c)# queue-limit dscp values af12 percent 90
C3850(config-pmap-c)# queue-limit dscp values af11 percent 100
! TRANSACTIONAL is allotted 5 percent of the bandwidth and
! allowed to consume 10 percent of additional buffers from the
! global buffer pool, and provided tail drop thresholds for three AF
! classes at rates 80,90,100

C3850(config-pmap-c)# class class-default
C3850(config-pmap-c)# bandwidth remaining percent 25
C3850(config-pmap-c)# queue-buffers ratio 25
! class-default is allotted 25 percent of the bandwidth and
! allowed to consume 25 percent of additional buffers from the
! global buffer pool

 ! This section configures interface egress queuing parameters
C3850(config)# int gi 1/0/13
C3850(config-if)# service-policy out 2P6Q3T
```

Note Queue buffer ratios are set very loosely in the preceding examples and need to be adjusted based on the traffic content and desired latency in the specific network. The values above do not establish much delineation for the bulk-scavenger queue, which may require additional buffering because of the low bandwidth allocation.

You can verify the configuration in Example 20-14 with the following commands:

- **show platform qos hw_state target gigabit x/y** (as shown in Example 20-15)
- **show platform qos policy target gigabit x/y** (as shown in Example 20-16)
- **show policy-map interface** (as shown in Example 20-17)
- **show platform qos queue stats** (as shown in Example 20-18)
- **show platform dscp-cos counters** (as shown in Example 20-19)

- **show policy-map interface wireless**
- **show platform qos queue config** (as shown in Example 20-20)

Example 20-15 *Verifying Application of Egress Queuing policy on a Catalyst 3850:* ***show platform qos hw_state*** *target*

```
C3850# show platform qos policy hw_state target gigabitethernet 1/0/2
Input policy : Not attached
Output policy :2P6Q3T
H/W programming State:INSTALLED IN HW
```

Example 20-15 shows an example of the state of the attached policy. This is an important verification step when troubleshooting specific configuration caveats. If the policy is attached to the interface but not installed in hardware, there might be a capacity issue or the policy itself might be invalid.

Example 20-16 *Verifying Egress Queuing on a Catalyst 3850:* ***show platform qos policy target gigabit*** *x/y*

```
C3850# show platform qos policy target gi 1/0/2
Input policy :
--------------
         Not attached
Output policy :
--------------
POLICY: 2P6Q3T Num Classes:8 Targets:1
  PMAP:0x120724b8 NextPMAP:0x120998d0 PrevPMAP:(nil)
  UP Mask: 0, Lookup Type:216
  COS Mask: 0, dscp mask:72410903774319872
  Filter flags: 0x2, Action Flags:0x14, num_classmaps 8 policy_type:Invalid nfl_req_
pending_cnt:0 pmap_qsize:0
    CLASS: PRIORITY-QUEUE
      CMAP:0x4a54c88c Next:0x4a5447fc Prev:(nil)
      Masks:- UP:0, CoS: 0, Dscp:70368744177664
      Filter flags 0x2
      Match DSCP:46
      Negate: NO Next:(nil)
    CLASS: REAL-TIME-VIDEO-QUEUE
      CMAP:0x4a5447fc Next:0x4a54a604 Prev:0x4a54c88c
      Masks:- UP:0, CoS: 0, Dscp:1103806595072
      Filter flags 0x2
      Match DSCP:32 40
      Negate: NO Next:(nil)
    CLASS: CONTROL-MGMT-QUEUE
      CMAP:0x4a54a604 Next:0x4a544664 Prev:0x4a5447fc
      Masks:- UP:0, CoS: 0, Dscp:72339069031481344
      Filter flags 0x2
      Match DSCP:16 24 48 56
      Negate: NO Next:(nil)
    CLASS: MULTIMEDIA-CONFERENCING-QUEUE
      CMAP:0x4a544664 Next:0x4a5453ac Prev:0x4a54a604
      Masks:- UP:0, CoS: 0, Dscp:360777252864
      Filter flags 0x2
      Match DSCP:34 36 38
      Negate: NO Next:(nil)
    CLASS: MULTIMEDIA-STREAMING-QUEUE
      CMAP:0x4a5453ac Next:0x4a54aacc Prev:0x4a544664
      Masks:- UP:0, CoS: 0, Dscp:1409286144
      Filter flags 0x2
      Match DSCP:26 28 30
      Negate: NO Next:(nil)
    CLASS: TRANSACTIONAL-DATA-QUEUE
      CMAP:0x4a54aacc Next:0x4a54ae84 Prev:0x4a5453ac
      Masks:- UP:0, CoS: 0, Dscp:5505024
      Filter flags 0x2
      Match DSCP:18 20 22
      Negate: NO Next:(nil)
    CLASS: BULK-SCAVENGER-DATA-QUEUE
      CMAP:0x4a54ae84 Next:0x4a545874 Prev:0x4a54aacc
      Masks:- UP:0, CoS: 0, Dscp:21760
      Filter flags 0x2
      Match DSCP:8 10 12 14
      Negate: NO Next:(nil)
    CLASS: class-default
      CMAP:0x4a545874 Next:(nil) Prev:0x4a54ae84
      Masks:- UP:0, CoS: 0, Dscp:0
      Filter flags 0
      Not Supported
      Negate: NO Next:(nil)
  Target:Gi1/0/2 DIR:OUT STATE:SET IN HW LE_LABEL:1, client_policy_type:LE
    IF:0x01047d800000000a TCCGs:8 TCG:0x4a544dd4 Next:(nil) Prev:(nil)
    CLASS: PRIORITY-QUEUE
      Action flags:0x14
      AG Policer Action Num:0
      Rate:10000000 Peak:0
      Limit:0 ExtLimit:0
      BC:312500 Peak:0
      Color Mode:Color Blind Type:1R2C
      Queue Action
      Buffer ratio:0 Limit Type:Queue Limit Single
    CLASS: REAL-TIME-VIDEO-QUEUE
      Action flags:0x14
      AG Policer Action Num:1
      Rate:20000000 Peak:0
      Limit:0 ExtLimit:0
      BC:625000 Peak:0
      Color Mode:Color Blind Type:1R2C
      Queue Action
      Buffer ratio:0 Limit Type:Queue Limit Single
    CLASS: CONTROL-MGMT-QUEUE
      Action flags:0x10
      Queue Action
      Buffer ratio:10 Limit Type:Queue Limit Single
    CLASS: MULTIMEDIA-CONFERENCING-QUEUE
      Action flags:0x10
      Queue Action
      Buffer ratio:10 Limit Type:Queue Limit DSCP
    CLASS: MULTIMEDIA-STREAMING-QUEUE
      Action flags:0x10
      Queue Action
      Buffer ratio:10 Limit Type:Queue Limit DSCP
    CLASS: TRANSACTIONAL-DATA-QUEUE
      Action flags:0x10
      Queue Action
      Buffer ratio:10 Limit Type:Queue Limit DSCP
    CLASS: BULK-SCAVENGER-DATA-QUEUE
      Action flags:0x10
      Queue Action
      Buffer ratio:10 Limit Type:Queue Limit DSCP
    CLASS: class-default
      Action flags:0x10
      Queue Action
      Buffer ratio:25 Limit Type:Queue Limit Single
  Hardware States ****************
      MAC LE Label:1
      IPV4 LE Label:1
      Priority: 13
      Translation Index:0
      Trust:TRUST_DSCP
  STATS ARE IN BYTE_COUNT FORMAT...
   PORT 20 RT1 Policer Accept Stats 0
   PORT 20 RT2 Policer Accept Stats 0
   PORT 20 RT1 Policer Drop Stats 0
   PORT 20 RT2 Policer Drop Stats 0
```

```
Additional details for policy: 2P6Q3T
        Number of targets attached to : 1
        Number of targets policy successfully programmed in HW: 1
```

Example 20-16 shows the detail behind what is installed in hardware for the policy attached to the interface. This provides detailed information about the number of classes and number of interfaces this policy is assigned to and how many have been successfully programmed in the system's hardware. This command is similar to the **show policy-map interface** command but provides details around the actual programming of the interfaces.

Example 20-17 *Verifying Egress Queuing on a Catalyst 3850:* ***show policy-map interface***

```
C3850# show policy-map int gi 1/0/25
 GigabitEthernet1/0/25

  Service-policy output: 1P7Q3T

    queue stats for all priority classes:
      Queueing
      priority level 1

      (total drops) 0
      (bytes output) 9243213

    Class-map: PRIORITY-QUEUE (match-any)
      Match:  dscp ef (46)
      Priority: Strict,

      Priority Level: 1
      police:
         rate 30 %
         rate 300000000 bps, burst 9375000 bytes
          conformed 8892522 bytes; actions:
            transmit
          exceeded 0 bytes; actions:
            drop
          conformed 86000 bps, exceeded 0 bps

    Class-map: CONTROL-MGMT-QUEUE (match-any)
      Match:  dscp cs2 (16) cs3 (24) cs6 (48) cs7 (56)
      Queueing
      (total drops) 0
      (bytes output) 11333
      bandwidth remaining 10%
      queue-buffers ratio 10

    Class-map: MULTIMEDIA-CONFERENCING-QUEUE (match-any)
      Match:  dscp af41 (34) af42 (36) af43 (38)
      Queueing
      queue-limit dscp 34 percent 100
      queue-limit dscp 36 percent 90
      queue-limit dscp 38 percent 80

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%

      queue-buffers ratio 10

    Class-map: MULTIMEDIA-STREAMING-QUEUE (match-any)
      Match:  dscp af31 (26) af32 (28) af33 (30)
      Queueing
      queue-limit dscp 26 percent 100
      queue-limit dscp 28 percent 90
      queue-limit dscp 30 percent 80

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%

      queue-buffers ratio 10

    Class-map: TRANSACTIONAL-DATA-QUEUE (match-any)
      Match:  dscp af21 (18) af22 (20) af23 (22)
      Queueing
      queue-limit dscp 18 percent 100
      queue-limit dscp 20 percent 90
      queue-limit dscp 22 percent 80

      (total drops) 0
      (bytes output) 46844266
      bandwidth remaining 10%

      queue-buffers ratio 10
    Class-map: BULK-DATA-QUEUE (match-any)
      Match:  dscp af11 (10) af12 (12) af13 (14)
      Queueing
      queue-limit dscp 10 percent 100
      queue-limit dscp 12 percent 90
      queue-limit dscp 14 percent 80

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 4%

      queue-buffers ratio 10

    Class-map: SCAVENGER-QUEUE (match-any)
      Match:  dscp cs1 (8)
      Queueing

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 1%
      queue-buffers ratio 10

    Class-map: class-default (match-any)
      Match: any
      Queueing

      (total drops) 0
      (bytes output) 64193461
      bandwidth remaining 25%
      queue-buffers ratio 25
```

Example 20-18 *Verifying Egress Queuing on a Catalyst 3850:* ***show platform qos queue stats***

```
C3850# show platform qos queue stats gi 1/0/25
DATA Port:21 Enqueue Counters
-------------------------------
Queue Buffers Enqueue-TH0 Enqueue-TH1 Enqueue-TH2
----- ------- ----------- ----------- -----------
    0       0           0           0     8452618
    1       0           0           0       11333
    2       0           0           0           0
    3       0           0           0           0
    4       0           45844266    0           0
    5       0           0           0           0
    6       0           0           0           0
    7       0           0           0    64193115
DATA Port:21 Drop Counters
-------------------------------
Queue Drop-TH0    Drop-TH1    Drop-TH2    SBufDrop    QebDrop
----- ----------- ----------- ----------- ----------- -----------
    0           0           0           0           0           0
    1           0           0           0           0           0
    2           0           0           0           0           0
    3           0           0           0           0           0
    4           0           0           0           0           0
    5           0           0           0           0           0
    6           0           0           0           0           0
    7           0           0           0           0           0
 AQM Broadcast Early WTD COUNTERS(In terms of Bytes)
--------------------------------------------------
  PORT TYPE          ENQUEUE             DROP
--------------------------------------------------
 UPLINK PORT-0        N/A               0
 UPLINK PORT-1        N/A               0
 UPLINK PORT-2        N/A               0
 UPLINK PORT-3        N/A               0
 NETWORK PORTS           0              0
 RCP PORTS               0              0
 CPU PORT                0              0
! Note: Queuing stats are in bytes
```

Examples 20-17 and 20-18 can be used together to show the mapping of traffic to queues and thresholds. You can see that the CONTROL-MGMT-QUEUE in Example 20-17 has 11,333 packets. From Example 20-18, that same number of packets can be seen as enqueued to the tail of the queue (Q1T2). Traffic is also being marked AF23 from another access port, as shown in Q4T0 or the TRANSACTIONAL-QUEUE.

Example 20-19 *Verifying Egress Queuing on a Catalyst 3850:* ***show platform qos dscp-cos counters***

```
C3850# show platform qos dscp-cos counters gigabitethernet 1/0/25
Ingress DSCP0 504638        0
Ingress DSCP1 0             0
Ingress DSCP2 0             0
Ingress DSCP3 0             0
Ingress DSCP4 0             0
Ingress DSCP5 0             0
Ingress DSCP6 0             0
Ingress DSCP7 0             0
Ingress DSCP8 0             0
Ingress DSCP9 0             0
Ingress DSCP10 0            0
Ingress DSCP11 0            0
Ingress DSCP12 0            0
Ingress DSCP13 0            0
...
```

Example 20-19 illustrates the ingress/egress QoS values of traffic. This command helps to provide an idea of what values may be arriving at an interface.

Example 20-20 *Verifying Egress Queuing on a Catalyst 3850:* ***show platform qos queue config***

```
C3850# show platform qos queue config gi 1/0/12
DATA Port:6 GPN:12 AFD:Disabled QoSMap:1 HW Queues: 48 - 55
  DrainFast:Disabled PortSoftStart:1 - 600
----------------------------------------------------------
  DTS Hardmax   Softmax  PortSMin GlblSMin  PortStEnd
  --- --------  -------- -------- --------- ---------
 0   1  6    30 11   120  0    80   0     0   0   800
 1   1  4     0 10   360  5   240   5    90   5   800
 2   1  4     0 11   120  6    80   6    30   6   800
 3   1  4     0 11   120  6    80   6    30   6   800
 4   1  4     0 11   120  6    80   6    30   6   800
 5   1  4     0 11   120  6    80   6    30   6   800
 6   1  4     0 11   120  6    80   6    30   6   800
 7   1  4     0 11   120  6    80   6    30   6   800
 Priority   Shaped/shared   weight  shaping_step
 --------   ------------   ------  ------------
 0      1     Shared           50           0
 1      7     Shared           50           0
 2      7     Shared          100           0
 3      7     Shared          100           0
 4      7     Shared          100           0
 5      7     Shared          100           0
 6      7     Shared           50           0
 7      7     Shared          100           0

Weight0 Max_Th0 Min_Th0 Weigth1 Max_Th1 Min_Th1 Weight2 Max_Th2 Min_Th2
------- -------  ------  ------  ------  ------  ------  ------ ------
0     0     119       0       0     133       0       0     150       0
```

```
1      0     286       0       0     320       0       0     360       0
2      0      95       0       0     106       0       0     120       0
3      0      95       0       0     106       0       0     120       0
4      0      95       0       0     106       0       0     120       0
5      0      95       0       0     106       0       0     120       0
6      0      95       0       0     106       0       0     120       0
7      0      95       0       0     106       0       0     120       0
```

Example 20-20 illustrates what port-based buffering is configured. This example shows a single priority (Q0) with hard buffers set to 30 and the first CBWFQ set to a **queue-buffers ratio 30** allocating a soft maximum of 360 buffers for the class. Because hard buffers are not set for nonpriority queues, these are values available from the common pool.

Wireless Queuing

The wireless queuing model is a four-queue structure, of which two are strict priority. These priority queues are to be used for voice and video separation and are policed to a specific rate. The other queues are defined as the **class-default** "catchall" queue and a new statically defined queue non-client-nrt-class. The non-client-nrt-class queue is specific to multicast traffic that is not matched by the real time or priority queues. Though multicast traffic marked CS5 (which is destined to the priority queue) will go to the priority queue, if multicast traffic is not marked as real-time traffic, it is placed into the non-client-nrt-class queue.

Scheduling on the wireless ports, as illustrated in Figure 20-7, is accomplished by class-based weighted fair queuing (CBWFQ), but this occurs on only two of the four queues because strict-priority scheduling is used on the other two.

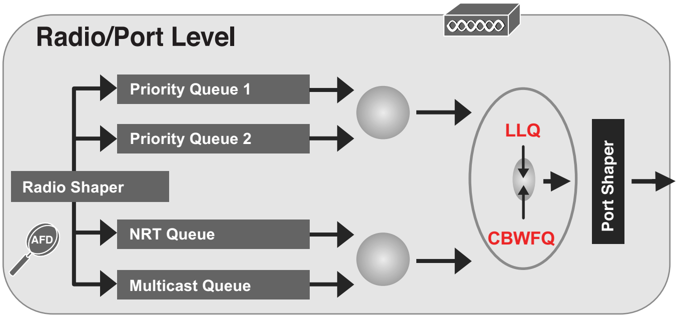


Figure 20-7 *Catalyst 3850 Egress Wireless Queuing Model (2P2Q)*

Strict-priority queues are fully serviced ahead of all other queues until empty. When configuring more than one priority queue, designated by level 1 or 2, scheduling is performed in strict fashion between the priority queues. For example, after the first priority queue has been fully serviced, the scheduler then services the second priority queue, and then begins to service the nonpriority queues, according to CBWFQ. A strict-priority queue is enabled with the **priority level** *x* interface command and may require a policer to limit the size.

Note Policers associated with the priority queue at this level are for multicast real-time traffic only. Multicast packets marked with a priority-based DSCP value (that is, EF, CS5) will cause the packet to be queued to one of the priority queues. If the multicast packet is not marked with such a DSCP value, it will be classified **non-client-nrt-class** and destined for that specific multicast non-real-time queue.

Approximate fair drop (AFD) is the bandwidth control algorithm used to control bandwidth allocation among classes that share the class-default queue of wireless interfaces. AFD provides fairness between clients by calculating virtual queue lengths at the radio, SSID, and client levels. These virtual queue lengths trigger probabilistic drops at the client level for clients that are consuming greater than their fair share of bandwidth. This algorithm is relatively similar to a per-client policer function. Both AFD and policers drop packets upon arrival, but policers drop deterministically, and AFD drops probabilistically and based on a variable used to calculate fairness. As a result, AFD drops traffic in a way that is more TCP friendly, allowing TCP windowing to assist in the bandwidth control that AFD is performing.

Figure 20-8 helps to explain AFD. A queue reference based on the amount of allocated bandwidth given to the default class (Data Queue) provides the foundation of queue reference calculations necessary to provide the fairness between clients. The amount of radio bandwidth, number of SSIDs, and number of clients are used to dynamically calculate the fair share at the client level. Once it is determined that a client has reached the reference threshold, the drop probability calculation is used to enforce fair packet drops for the client flows.

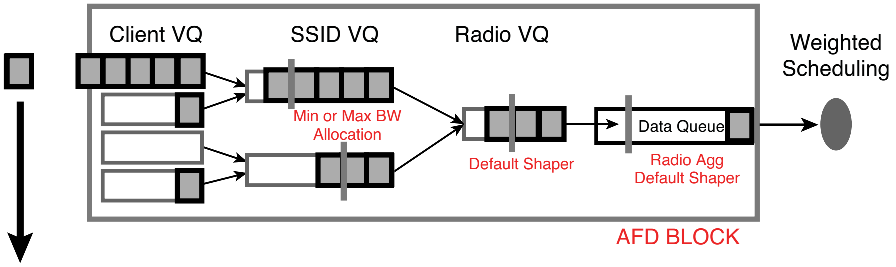


Figure 20-8 *Cisco Catalyst 3850 AFD*

A simple example can be surmised from the diagram in Figure 20-8. At the radio level, if 120 Mbps of bandwidth is available and two SSIDs are configured, each SSID is allocated 60 Mbps of bandwidth and provides a fair share of the bandwidth to each client. If a policy were to be applied at the SSID level that modified the bandwidth-remaining ratio allocation for an SSID, the reference rate would be based on the SSID-level configuration. Assuming no SSID policy in this example, the 60 Mbps would be further divided based on the number of clients. Each client would be again given its fair share based on the available bandwidth, taking into consideration the priority traffic being sent from the client into the physical port. Priority traffic bypasses the AFD block but is taken into consideration at the SSID level for accounting purposes. After the priority allocation is removed at the SSID level, the rest of the available bandwidth is split between each of the clients.

Wireless 2P2Q Egress Queuing Model

Egress wireless queuing on the Catalyst 3850 family of switches can be configured as 2P2Q.

Buffering on the wireless side is the same as the wired side. The difference lies in the default ratios and the limited number of queues, as was detailed in the previous section. It is not recommended to modify the buffering on the wireless side because of the number of queues and the defaults being adjusted for the best possible result.

With the default queuing parameters in place for 2P2Q egress queuing, bandwidth allocation between the multicast-NRT and class-default queue need to be addressed. Specifically, you first limit multicast traffic destined for the priority queues as provided by policers at the wireless port level. Priority level 1 is considered the voice class and therefore limited to 10 percent, and video is limited to 20 percent, adhering to the “33 percent LLQ rule.” In this case, the physical policers only limit multicast traffic, not unicast. To limit unicast traffic, a policy must be applied in the downstream direction to the SSID. Because this is not part of the queuing policy, it is not shown here.

Next, allocating bandwidth to the remaining two classes (non-client-nrt-class and class-default) is determined by what is left of the remaining 70 percent of available bandwidth. Class-default can be allocated the lion’s share at 63 percent, and non-client-nrt can be given the remaining 7 percent. This allocation is flexible enough to limit NRT multicast while providing for other applications in the default class during congestion. Thresholding is not available in this queuing model because of the fairness provided per client by AFD. In the class-default, each wireless client adheres to AFD drops based on calculated probability to achieve per-client fairness, as discussed previously in the section “Wireless Queuing,” which covered AFD.

Figure 20-9 shows the 2P2Q egress queuing mappings for the Catalyst 3850 wireless ports.

Example 20-21 shows the corresponding configuration for an eight-class 2P2Q wireless egress queuing on the Catalyst 3850.

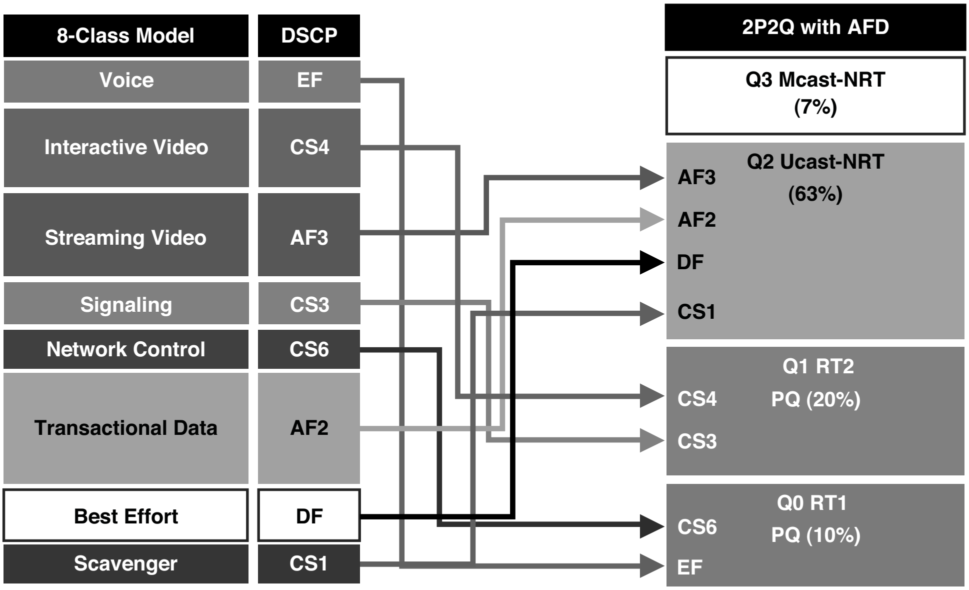


Figure 20-9 *Catalyst 3850 Eight-Class (2P2Q) Wireless Egress Queuing Model*

Example 20-21 *Catalyst 3850 Eight-Class (2P2Q) Wireless Egress Queuing Configuration Example*

```
! This section configures egress wireless queuing and a dual PQ
C3850(config)# policy-map port_child_policy
C3850(config-pmap)# class non-client-nrt-class
C3850(config-pmap-c)# bandwidth remaining ratio 7
 ! non-client-nrt-class is a static class for multicast non-real-time
 ! traffic allocated 7 percent of bandwidth not allocated to the
 ! priority queue
C3850(config-pmap-c)# class RT1
C3850(config-pmap-c)# priority level 1
C3850(config-pmap-c) police rate percent 10 conform-action transmit     exceed-
action drop
 ! RT1 is set to level 1 and serviced before any other class
 ! Multicast traffic in this queue is policed to 10 percent of the
 ! physical interface rate
C3850(config-pmap-c-police)# class RT2
C3850(config-pmap-c)# priority level 2
C3850(config-pmap-c) police rate percent 20 conform-action transmit exceed-action
drop
 ! RT2 is set to level 2 and serviced before any other class
 ! Multicast traffic in this queue is policed to 20 percent of the
 ! physical interface rate
```

```
C3850(config-pmap-c-police)# class class-default
C3850(config-pmap-c)# bandwidth remaining ratio 63

 ! This section configures wireless interface egress queuing parameters
C3850(config)# This policy is applied automatically by the WCM to all wireless ports
- no policy attachment necessary
```

Note The egress wireless queuing configuration is applied automatically by the WCM based on a wireless port negotiation. When it is determined that a port is wireless in nature, the static policy map **port_child_policy** will be applied along with the radio-level shaper. If the priority levels are not turned on at the wireless port level, they will not be usable at the SSID or client-level policies.

You can verify the configuration in Example 20-21 with the following commands:

- **show platform qos hw_state target gigabit x/y**
- **show platform qos policy target gigabit x/y**
- **show policy-map interface**
- **show policy-map interface wireless**

Summary

This chapter detailed best-practice recommendations for QoS design on the Cisco Catalyst 3850 series switch deployed in the role of a campus access layer switch supporting both wired and wireless network access. A majority of these designs are also compatible with the Catalyst 4500 Sup6/7 switch family when used in the access layer.

The first step to deploying QoS on this platform is to globally enable trust, if applicable. If this small but important step is not performed, any and all QoS marking configured will be invalidated. Once the untrusted command is disabled, the need for trust policies at the SSID level goes away.

This chapter next discussed how to define ingress QoS policies, which may include conditional trust (based on Cisco devices) and classification and marking (with optional policing policies).

Egress queuing policies for both wired and wireless port types were also detailed, showing how an 8- and 12-class strategic model can all be provisioned on this platform. The differences in wired and wireless queuing models and details as to these differences were also explored in detail.

Additional platform-specific design options and considerations are discussed in the case study in the next chapter, including how to configure per-SSID and per-client QoS.

Additional Reading

Cisco Enterprise Medianet Campus QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html

Medianet Campus QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampusaag.html

Medianet Catalyst 3560/3750 QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampuscat3xxxaag.html

Medianet Campus AutoQoS At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/autoqosmediacampus.pdf

Cisco Catalyst 3850 and Catalyst 3560-X Switch Software Configuration Guide, Cisco IOS Release 15.0(2)SE: QoS Configuration Guide: http://www.cisco.com/en/US/docs/switches/lan/catalyst3750x_3560x/software/release/15.0_2_se/configuration/guide/swqos.html

This page intentionally left blank

Chapter 21

Converged Access QoS Design Case Study

Having defined Tifosi Software's strategic eight-class end-to-end QoS model (in Chapter 12, "Strategic QoS Design Case Study," and as illustrated in Figure 12-2), the networking team is now ready to apply these policies on a per-platform basis to their converged access campus network infrastructure.

Tifosi has a multitier campus network defined in Chapter 17, "Campus QoS Design Case Study." In addition to this design, the network also provides for wired and wireless convergence via the Cisco Catalyst 3850 access switch and CT5760 wireless LAN controller (WLC).

Tifosi has multiple types of endpoints connecting to the access layer of the campus, including the following:

- Desktop PCs and Macs, in addition to UNIX/Linux workstations (abbreviated simply as PCs)
- Cisco IP phones
- Cisco wireless access points
- Cisco TelePresence Systems
- Printers
- Mobile wireless devices

The networking team wants to ensure that all endpoints have corresponding trust or explicit classification and marking policies applied to them. They also want to extend these policies to mobile wireless clients with the least amount of policy modifications.

Tifosi has two main service set identifiers (SSIDs) used within the campus: guest and enterprise. The networking team has often noticed that a disproportionate amount of traffic runs across the guest SSID, which inhibits internal users on the enterprise SSID. They would like to eliminate this issue by providing 70 percent of the available bandwidth to enterprise users while limiting the guest users to 30 percent of that same available bandwidth. In addition, the networking team wants to ensure fairness between users in the SSIDs while limiting the users to 1 Mbps each bidirectionally.

Wireless users are currently authenticated via Cisco's Identity Services Engine (ISE). As part of this platform, user credentials are used to provide QoS policy modifications when the users are authorized into the network. QoS policy name information will be passed from ISE to the Catalyst 3850 based on these credentials, providing user-based QoS treatment.

Finally, the networking team wants to ensure that eight-class egress queuing policies are applied on all wired and uplink ports, while mapping into a four-queue model for access points (APs).

Figure 21-1 shows the QoS policies to be applied to Tifosi's access network. Logical placement of policies for wireless devices is on the Catalyst 3850.

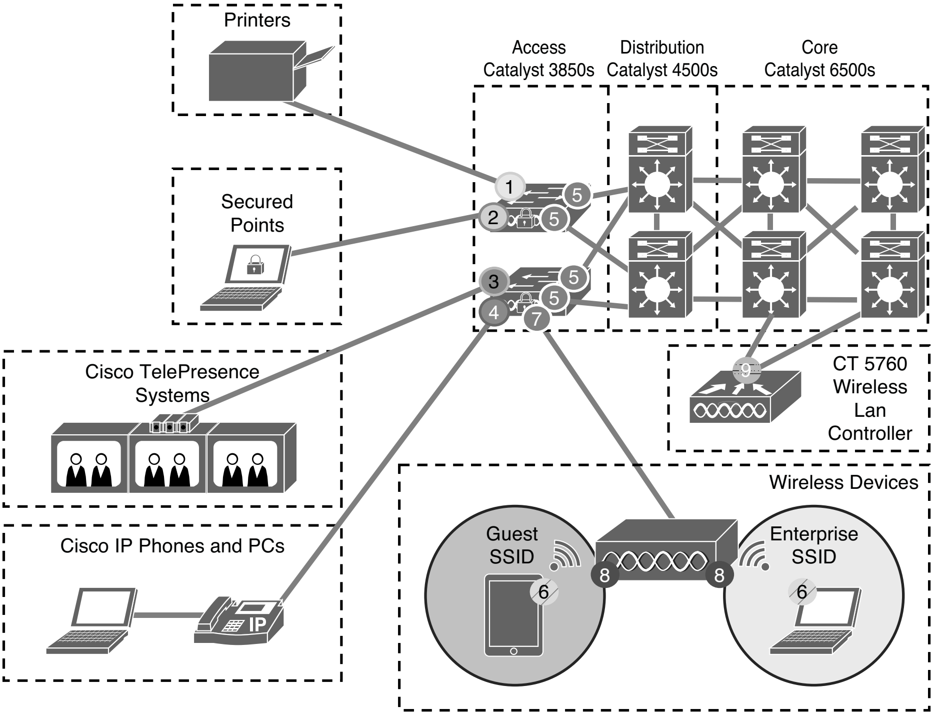


Figure 21-1 *Tifosi Software Case Study: Campus Network QoS Design*

These campus QoS policies are detailed as follows:

Wired Policies

- **Policy 1: Untrusted wired endpoints (Catalyst 3850)**
  - Reset QoS packet marking to 0.
  - Enable 1P7Q3T egress queuing for Tifosi's eight-class QoS model.
- **Policy 2: Trusted wired endpoints (Catalyst 3850)**
  - Trust DSCP.
  - Enable 1P7Q3T egress queuing for Tifosi's eight-class QoS model.
- **Policy 3: Conditionally trusted wired endpoints: TelePresence (Catalyst 3850)**
  - Enable conditional DSCP trust for a Cisco TelePresence System.
  - Enable 1P7Q3T egress queuing for Tifosi's eight-class QoS model.
- **Policy 4: Conditionally trusted wired endpoints: IP phones + PCs (Catalyst 3850)**
  - Enable conditional CoS trust for a Cisco IP phone.
  - Map CoS 5 to DSCP EF.
  - Classify and mark traffic according to Table 21-1.
  - Enable 1P7Q3T egress queuing for Tifosi's eight-class QoS model.
- **Policy 5: Queuing wired ports (wired 3850)**
  - Trust DSCP.
  - Enable 1P7Q3T egress queuing for Tifosi's eight-class QoS model.

Wireless Policies

- **Policy 6: Authorized wireless endpoint - per-client QoS (Catalyst 3850)**
  - Enable DOT1X authentication.
  - Classify and mark traffic according to Table 21-1.
  - Push QoS policy name from RADIUS to Catalyst 3850 for authorized clients.
  - Enable 2P2Q egress queuing for Tifosi's eight-class QoS model.
- **Policy 7: Queuing wireless ports (Wireless 3850)**
  - Trust DSCP.
  - Enable 2P2Q egress queuing for Tifosi's eight-class QoS model.
- **Policy 8: Bandwidth differentiation between SSIDs**
  - Trust DSCP (trust needed unless **untrusted** command disabled).
  - Modify bandwidth ratio at service set identifier SSID level.

- **Policy 9: CT 5760 Wireless LAN controller uplink ports**
  - Trust DSCP (trust enabled for wired to wired traffic by default).
  - Enable 1P7Q3T egress queuing for an Tifosi's eight-class QoS model.

Note Table 21-1 presents *example* values for TCP/UDP port-based classification that are being used by Tifosi. These values may change and may not be exhaustive. Refer to application-specific documentation to verify TCP/UDP ports to be used for application classification.

Table 21-1 *Tifosi's Eight-Class Classification and Marking Policy Table for Cisco IP Phone + PC Endpoints*

Application Class	Classification Criteria	Marking
Voice	Conditionally trusted from IP phones	EF
Real-Time Interactive	Conditionally trusted from Cisco TelePresence Systems	CS4
Signaling	Conditionally trusted (VVLAN) or SCCP (DVLAN-TCP 2000) or SIP (DVLAN-TCP 5060-5061)	CS3
Multimedia Conferencing	Cisco Jabber (UDP/RTP 16384-32767) or Microsoft Lync (TCP 50000-59999)	AF4
Transactional Data	HTTPS (TCP 443) or Citrix (TCP 3389, 5985, 8080) or Oracle (TCP 1521, 1527, 1575, 1630, 6200)	AF2
Bulk Data	FTP (TCP 20 & 21) or Secure FTP (TCP 22) or SMTP (TCP 25) or Secure SMTP (TCP 465) or IMAP (TCP 143) or Secure IMAP (TCP 993) or POP3 (TCP 11) or Secure POP3 (TCP 995) or Connected PC Backup (TCP 1914)	AF1
Scavenger	BitTorrent (TCP 6881-6999) or Apple iTunes (TCP/UDP 3689) or Microsoft Direct X Gaming (TCP/UDP 2300-2400)	CS1
Best Effort	Default	DF

Tifosi Converged Access QoS Design: Wired

Tifosi's access layer consists of wired Cisco Catalyst 3750s and converged Cisco Catalyst 3850s. Policies outlined in the sections that follow are in reference to the wired ports of the Catalyst 3850 and include combinations of the following policies:

- Disabling trust for printers and similar devices
- Enabling trust for secured devices
- Enabling conditional trust for Cisco TelePresence Systems
- Enabling conditional trust for Cisco IP phones + PCs
- Enabling 1P7Q3T egress queuing for Tifosi's eight-class QoS model

The configuration for each of these campus access policies is shown in turn. However, to better clarify wired and wireless configuration, the first section is associated with wired clients and the second to wireless. Note that the client policies for wired and wireless are the same, but queuing differs somewhat.

Policy 1: Access-Edge Design for Wired Printer Endpoints (No Trust)

The QoS policy to be used on all ports connecting to untrusted endpoints is detailed in Example 21-1.

Example 21-1 *Tifosi Access-Edge Design for Untrusted Endpoints (Printer Examples) on a Catalyst 3850*

```
C3850(config-cmap)# policy-map client-default
C3850(config-pmap-c)# class class-default
C3850(config-pmap-c)# set dscp default
C3850(config)# interface GigabitEthernet 1/0/46
C3850(config-if)# switchport access vlan 10
C3850(config-if)# service-policy in client-default
! Port is configured to be statically untrusted!
Queuing configurations detailed in Examples 21-6 and 21-7
```

Policy 2: Access-Edge Design for Wired Access Endpoints (DSCP Trust)

The QoS policy to be used on all ports connecting to DSCP trusted endpoints is shown in Example 21-2.

Example 21-2 *Tifosi Case Study: Access-Edge Design for Trusted Endpoints (Secured PC or IP Conferencing Station Endpoint Example) on a Catalyst 3850*

```
C3850(config)# interface GigabitEthernet 1/0/47
C3850(config-if)# switchport access vlan 10
C3850(config-if)# [NO ADDITIONAL CONFIGURATION NECESSARY]
! The Catalyst 3850 trusts DSCP by default without configuration
! Wired Queuing configurations detailed in Examples 21-5
```

Policy 3: Access-Edge Design for Cisco TelePresence Endpoints (Conditional Trust)

The QoS policy to be used on all ports connecting to Cisco TelePresence Systems (CTS) is presented in Example 21-3.

Example 21-3 *Tifosi Case Study: Access-Edge Design for Conditionally Trusted Endpoints (Cisco TelePresence System Endpoint Example) on a Catalyst 3850*

```
C3850(config)# interface GigabitEthernet 1/0/48
C3850(config-if)# switchport access vlan 10
C3850(config-if)# switchport voice vlan 110
C3850(config-if)# spanning-tree portfast
C3850(config-if)# trust device cts
! Port is configured to conditionally trust the CTS endpoint
! The Catalyst 3850 trusts DSCP by default without configuration
! Queuing configurations detailed in Examples 21-5
```

Policy 4: Access-Edge Design for Cisco IP Phones and PCs (Conditional Trust and Classification and Marking)

The QoS policy to be used on all ports connecting to Cisco IP Phones + PCs (based on Table 21-1) is detailed in Example 21-4.

Example 21-4 *Tifosi Case Study: Access-Edge Design for Conditionally Trusted Endpoints (Cisco IP Phones and PCs Example) on a Catalyst 3850*

```
! This section configures the classification extended ACLs
C3850(config)# ip access-list extended SIGNALING
C3850(config-ext-nacl)# remark SCCP
C3850(config-ext-nacl)# permit tcp any any eq 2000
C3850(config-ext-nacl)# remark SIP
C3850(config-ext-nacl)# permit tcp any any range 5060 5061
```

```
C3850(config)# ip access-list extended MULTIMEDIA-CONFERENCING
C3850(config-ext-nacl)# remark CISCO-JABBER-RTP
C3850(config-ext-nacl)# permit udp any any range 16384 32767
C3850(config-ext-nacl)# remark MICROSOFT-LYNC
C3850(config-ext-nacl)# permit tcp any any range 50000 59999

C3850(config)# ip access-list extended TRANSACTIONAL-DATA
C3850(config-ext-nacl)# remark HTTPS
C3850(config-ext-nacl)# permit tcp any any eq 443
C3850(config-ext-nacl)# remark CITRIX
C3850(config-ext-nacl)# permit tcp any any eq 3389
C3850(config-ext-nacl)# permit tcp any any eq 5985
C3850(config-ext-nacl)# permit tcp any any eq 8080
C3850(config-ext-nacl)# remark ORACLE
C3850(config-ext-nacl)# permit tcp any any eq 1521
C3850(config-ext-nacl)# permit tcp any any eq 1527
C3850(config-ext-nacl)# permit tcp any any eq 1575
C3850(config-ext-nacl)# permit tcp any any eq 1630
C3850(config-ext-nacl)# permit tcp any any eq 6200

C3850(config)# ip access-list extended BULK-DATA
C3850(config-ext-nacl)# remark FTP
C3850(config-ext-nacl)# permit tcp any any eq ftp
C3850(config-ext-nacl)# permit tcp any any eq ftp-data
C3850(config-ext-nacl)# remark SSH/SFTP
C3850(config-ext-nacl)# permit tcp any any eq 22
C3850(config-ext-nacl)# remark SMTP/SECURE SMTP
C3850(config-ext-nacl)# permit tcp any any eq smtp
C3850(config-ext-nacl)# permit tcp any any eq 465
C3850(config-ext-nacl)# remark IMAP/SECURE IMAP
C3850(config-ext-nacl)# permit tcp any any eq 143
C3850(config-ext-nacl)# permit tcp any any eq 993
C3850(config-ext-nacl)# remark POP3/SECURE POP3
C3850(config-ext-nacl)# permit tcp any any eq pop3
C3850(config-ext-nacl)# permit tcp any any eq 995
C3850(config-ext-nacl)# remark CONNECTED PC BACKUP
C3850(config-ext-nacl)# permit tcp any eq 1914 any

C3850(config)# ip access-list extended SCAVENGER
C3850(config-ext-nacl)# remark BITTORRENT
C3850(config-ext-nacl)# permit tcp any any range 6881 6999
C3850(config-ext-nacl)# remark APPLE ITUNES MUSIC SHARING
C3850(config-ext-nacl)# permit tcp any any eq 3689
C3850(config-ext-nacl)# permit udp any any eq 3689
C3850(config-ext-nacl)# remark MICROSOFT DIRECT X GAMING
```

```
C3850(config-ext-nacl)# permit tcp any any range 2300 2400
C3850(config-ext-nacl)# permit udp any any range 2300 2400

C3850(config)# ip access-list extended DEFAULT
C3850(config-ext-nacl)# remark EXPLICIT CLASS-DEFAULT
C3850(config-ext-nacl)# permit ip any any


 ! This section configures the class maps
C3850(config-cmap)# class-map match-all VOICE
C3850(config-cmap)# match cos 5
 ! VoIP is trusted (from the VVLAN)- cos classification for IP phone

C3850(config-cmap)# class-map match-any SIGNALING
C3850(config-cmap)# match cos 3
 ! Signaling is trusted (from the VVLAN) - cos classification for IP phone
C3850(config-cmap)# match access-group name SIGNALING
 ! Signaling is classified by ACL (from the DVLAN)

C3850(config-cmap)# class-map match-all MULTIMEDIA-CONFERENCING
C3850(config-cmap)# match access-group name MULTIMEDIA-CONFERENCING
 ! Associates MULTIMEDIA-CONFERENCING access list with class map

C3850(config-cmap)# class-map match-all TRANSACTIONAL-DATA
C3850(config-cmap)# match access-group name TRANSACTIONAL-DATA
 ! Associates TRANSACTIONAL-DATA access list with class map

C3850(config-cmap)# class-map match-all BULK-DATA
C3850(config-cmap)# match access-group name BULK-DATA
 ! Associates BULK-DATA access list with class map

C3850(config-cmap)# class-map match-all SCAVENGER
C3850(config-cmap)# match access-group name SCAVENGER
 ! Associates SCAVENGER access list with class map

 ! This section configures the PC-MARKING policy map
C3850(config-cmap)# policy-map PC-MARKING
C3850(config-pmap-c)# class VOICE
C3850(config-pmap-c)# set dscp ef
 ! VoIP is marked EF
C3850(config-pmap-c)# class SIGNALING
C3850(config-pmap-c)# set dscp cs3
 ! Signaling is marked CS3
C3850(config-pmap-c)# class MULTIMEDIA-CONFERENCING
C3850(config-pmap-c)# set dscp af41
```

```
 ! Multimedia conferencing is marked AF41
C3850(config-pmap-c)# class TRANSACTIONAL-DATA
C3850(config-pmap-c)# set dscp af21
 ! Transactional data is marked AF21
C3850(config-pmap-c)# class BULK-DATA
C3850(config-pmap-c)# set dscp af11
 ! Bulk data is marked AF11
C3850(config-pmap-c)# class SCAVENGER
C3850(config-pmap-c)# set dscp cs1
 ! Scavenger traffic is marked CS1
C3850(config-pmap-c)# class class-default
C3850(config-pmap-c)# set dscp default
 ! class-default marks all other IP traffic to 0


 ! This section configures conditional trust on the interfaces
 ! and applies the PC-MARKING service policy
C3850(config)# interface range GigabitEthernet 1/0/1-45
C3850(config-if-range)# switchport access vlan 10
C3850(config-if-range)# switchport voice vlan 110
C3850(config-if-range)# trust device cisco-phone
 ! The interface is set to conditionally-trust Cisco IP Phones
C3850(config-if-range)# service-policy input PC-MARKING
 ! Attaches the PC-MARKING policy to the interface(s)
 ! Queuing configurations detailed in Examples 21-5
```

Policy 5: Access-Edge Wired Queuing Design

The QoS policy applied to all wired ports (based on Table 21-1) is detailed in Example 21-5. Figure 21-2 shows the 1P7Q3T queuing model used on all wired ports in graphical format.

Example 21-5 shows the corresponding configuration for 1P7Q3T egress queuing on the Catalyst 3850.

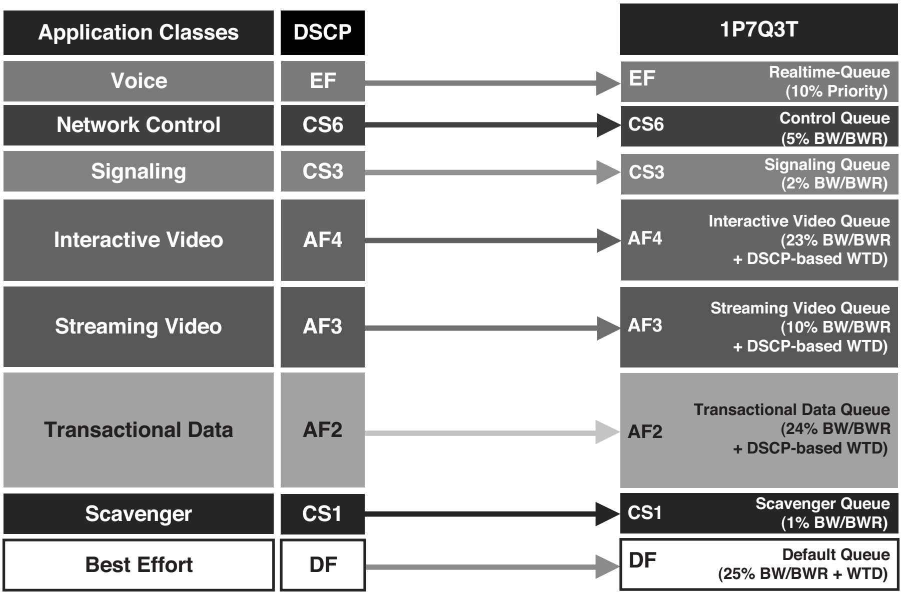

Application Classes
DSCP
1P7Q3T
Voice
EF
EF
Realtime-Queue (10% Priority)
Network Control
CS6
CS6
Control Queue (5% BW/BWR)
Signaling
CS3
CS3
Signaling Queue (2% BW/BWR)
Interactive Video
AF4
AF4
Interactive Video Queue (23% BW/BWR + DSCP-based WTD)
Streaming Video
AF3
AF3
Streaming Video Queue (10% BW/BWR + DSCP-based WTD)
Transactional Data
AF2
AF2
Transactional Data Queue (24% BW/BWR + DSCP-based WTD)
Scavenger
CS1
CS1
Scavenger Queue (1% BW/BWR)
Best Effort
DF
DF
Default Queue (25% BW/BWR + WTD)

Figure 21-2 *Tifosi Case Study: Catalyst 3850 Eight-Class (1P7Q3T) Egress Queuing Model*

Example 21-5 *Tifosi Case Study: 1P7Q3T Egress Wired Queuing Configuration Example on a Catalyst 3850*

```
! This section configures buffers, thresholds, and a single PQ
C3850(config-pmap-c)# policy-map 1P7Q3T
C3850(config-pmap)# class REALTIME-QUEUE
C3850(config-pmap-c)# priority level 1
C3850(config-pmap-c)# police rate percent 10
 ! PRIORITY is assigned strict priority with a policer rate set to 10
 ! percent of line rate
C3850(config-pmap-c-police)# class CONTROL-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 5
C3850(config-pmap-c)# queue-buffers ratio 10
 ! CONTROL is assigned 5 percent bandwidth remaining after the
 ! priority queue is serviced, and allowed up to an additional 1/10th of
 ! buffer pool
C3850(config-pmap)# class SIGNALING-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 2
 ! SIGNALING is assigned 2 percent bandwidth remaining after the
 ! priority queue is serviced
C3850(config-pmap-c)# class INTERACTIVE-VIDEO-QUEUE
```

```
C3850(config-pmap-c)# bandwidth remaining percent 23
C3850(config-pmap-c)# queue-buffers ratio 10
C3850(config-pmap-c)# queue-limit dscp af43 percent 80
C3850(config-pmap-c)# queue-limit dscp af42 percent 90
C3850(config-pmap-c)# queue-limit dscp af41 percent 100
 ! INTERACTIVE is assigned 23 percent bandwidth remaining after the
 ! priority queue is serviced, allowed up to an additional 1/10th of
 ! buffer pool, and provides for three thresholds for three DSCP values
C3850(config-pmap-c)# class STREAMING-VIDEO-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 10
C3850(config-pmap-c)# queue-buffers ratio 10
C3850(config-pmap-c)# queue-limit dscp af33 percent 80
C3850(config-pmap-c)# queue-limit dscp af32 percent 90
C3850(config-pmap-c)# queue-limit dscp af31 percent 100
 ! STREAMING is assigned 10 percent bandwidth remaining after the
 ! priority queue is serviced, allowed up to an additional 1/10th of
 ! buffer pool, and provides for three thresholds for three DSCP values
C3850(config-pmap-c)# class TRANSACTIONAL-DATA-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 24
C3850(config-pmap-c)# queue-buffers ratio 10
C3850(config-pmap-c)# queue-limit dscp af23 percent 80
C3850(config-pmap-c)# queue-limit dscp af22 percent 90
C3850(config-pmap-c)# queue-limit dscp af21 percent 100
 ! TRANSACTIONAL is assigned 24 percent bandwidth remaining after the
 ! priority queue is serviced, allowed up to an additional 1/10th of
 ! buffer pool, and provides for three  thresholds for three DSCP values
C3850(config-pmap-c)# class SCAVENGER-QUEUE
C3850(config-pmap-c)# bandwidth remaining percent 1
C3850(config-pmap-c)# queue-buffers ratio 10
 ! SCAVENGER is assigned 1 percent bandwidth remaining after the
 ! priority queue is serviced, and allowed up to an additional 1/10th of
 ! buffer pool
C3850(config-pmap-c)# class class-default
C3850(config-pmap-c)# bandwidth remaining percent 25
C3850(config-pmap-c)# queue-buffers ratio 25
 ! DEFAULT is assigned 25 percent bandwidth remaining after the
 ! priority queue is serviced, and is allowed up to an additional 1/4th
 ! of the buffer pool


 ! This section configures interface egress queuing parameters
C3850(config)# interface gigabitEthernet 1/0/13
C3850(config-if)# service-policy out 1P7Q3T
```

Note The eight-class model 1P7Q3T configuration in Example 21-5 can easily be expanded to cover two priority queues, as illustrated in Chapter 20, "Converged Access (Cisco Catalyst 3850 and the CT 5760 Wireless LAN Controller) QoS Design." The main difference is the separation of voice and video traffic and the condensing of Bulk and Scavenger classes. However, these are relatively minor modifications.

Note The numeric value used in the **queue-buffer ratio** command in the preceding example can be somewhat misleading. The given value is not the numeric denominator; it is, in fact, a ratio. For instance, if the queue buffer were to be 1/4 or 25 percent, the command would be **queue-buffer ratio 25**.

Tifosi Converged Access QoS Design: Wireless

Tifosi's mobile campus access layer consists of Cisco Catalyst 3850 switches, a CT5760 WLC, and several wireless clients.

Wireless clients will adhere to a marking/policing policy that is applied to all wireless clients. This policy follows the wireless clients when they roam between APs, but if a client roams outside of its switch peer group (SPG—also known as their domain/building) the traffic from the client will be directed to a mobility controller (MC—that is, CT 5760) which provides roaming between SPGs. In other words, if a client roams outside of its SPG the traffic is always sent to the MC; in other words, back to its original point of attachment, as if it originated there.

Due to this traffic flow, an additional queuing policy is configured at the CT5760 to maintain the prioritization of traffic.

Wireless clients and the Wireless Controller will be addressed in this section with the following mix of policies:

- Dynamically enabling a classification, marking and policing policy for clients
- Enabling 2P2Q+AFD egress queuing for Tifosi's eight-class QoS model
- Bandwidth separation between Guest and Enterprise SSIDs
- Enabling 1P7Q3T egress queuing for Tifosi's eight-class QoS model on the CT 5760
- Enabling AAA (ISE) to authenticate wireless clients

The configuration of these access QoS policies is shown in turn; also a description of ISE configuration is shown, but not in any significant detail. To find detailed information on setting up Cisco's Identity Services Engine (ISE): http://www.cisco.com/en/US/products/ps11640/index.html

The wireless QoS design to be applied to Tifosi's access network is shown in Figure 21-3. The diagram illustrates the position of the Catalyst 3850s, CT 5760 and the ISE.

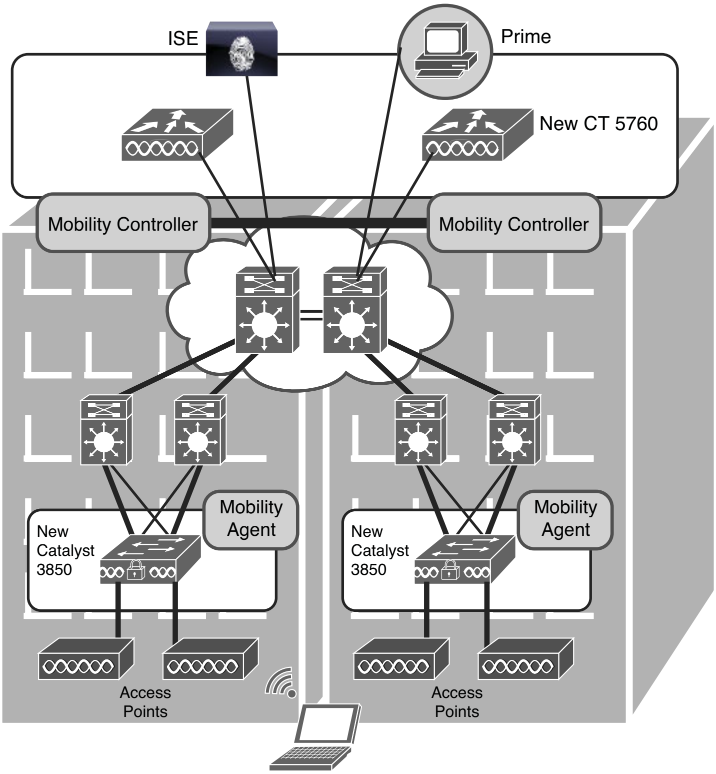


Figure 21-3 *Tifosi Software Case Study: Mobile Network QoS Design*

Policy 6: Access-Edge Design for Mobile Wireless Clients (Dynamic Policy with and Classification & Marking)

The QoS policy will be dynamically applied via ISE to wireless clients (based on Table 21-1) as detailed in Example 21-6.

Example 21-6 *Tifosi Case Study: Access-Edge Design for Mobile Wireless Clients (Wireless PCs or BYOD device Example) on a Catalyst 3850*

```
C3850(config-cmap)# class-map match-all VOICE
C3850(config-cmap)# match dscp ef
 ! VoIP is trusted (from the VVLAN)

C3850(config-cmap)# class-map match-any SIGNALING
```

```
C3850(config-cmap)# match dscp cs3
 ! Signaling is trusted (from the VVLAN)
C3850(config-cmap)# match access-group name SIGNALING
 ! Signaling is classified by ACL (from the DVLAN)

C3850(config-cmap)# class-map match-all MULTIMEDIA-CONFERENCING
C3850(config-cmap)# match access-group name MULTIMEDIA-CONFERENCING
 ! Associates MULTIMEDIA-CONFERENCING access list with class map

C3850(config-cmap)# class-map match-all TRANSACTIONAL-DATA
C3850(config-cmap)# match access-group name TRANSACTIONAL-DATA
 ! Associates TRANSACTIONAL-DATA access list with class map

C3850(config-cmap)# class-map match-all BULK-DATA
C3850(config-cmap)# match access-group name BULK-DATA
 ! Associates BULK-DATA access list with class map

C3850(config-cmap)# class-map match-all SCAVENGER
C3850(config-cmap)# match access-group name SCAVENGER
 ! Associates SCAVENGER access list with class map

 ! This section configures the PC-MARKING policy-map
C3850(config-cmap)# policy-map PC-MARKING
C3850(config-pmap-c)# class VOICE
C3850(config-pmap-c)# set dscp ef
 ! VoIP is marked EF
C3850(config-pmap-c)# class SIGNALING
C3850(config-pmap-c)# set dscp cs3
 ! Signaling is marked CS3
C3850(config-pmap-c)# class MULTIMEDIA-CONFERENCING
C3850(config-pmap-c)# set dscp af41
 ! Multimedia-conferencing is marked AF41
C3850(config-pmap-c)# class TRANSACTIONAL-DATA
C3850(config-pmap-c)# set dscp af21
 ! Transactional Data is marked AF21
C3850(config-pmap-c)# class BULK-DATA
C3850(config-pmap-c)# set dscp af11
 ! Bulk Data is marked AF11
C3850(config-pmap-c)# class SCAVENGER
C3850(config-pmap-c)# set dscp cs1
 ! Scavenger traffic is marked CS1
C3850(config-pmap-c)# class class-default
C3850(config-pmap-c)# set dscp default
 ! class-default marks all other IP traffic to 0
```

Policy 7: Access-Edge Wireless Queuing Design

The QoS policy is dynamically applied via WCM to wireless ports after the global policy has been modified (based on Table 21-1) as detailed in Example 21-7. Figure 21-4 shows the graphical representation of the wireless 2P2Q model applied dynamically to all wireless ports.

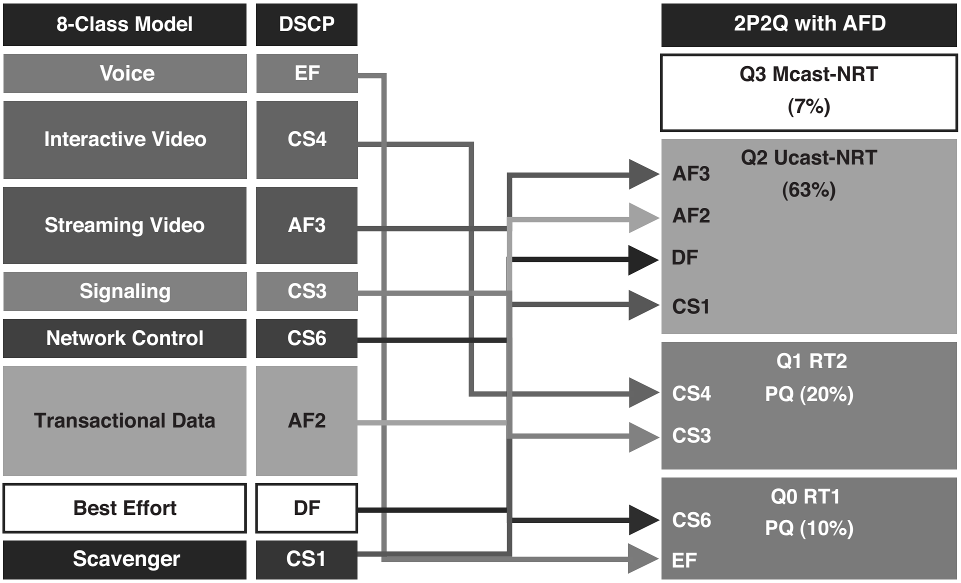


Figure 21-4 *Catalyst 3850 Egress Wireless Queuing Model 2P2Q*

Example 21-7 shows the corresponding configuration for 2P2Q egress queuing on the Catalyst 3850.

Example 21-7 *2P2Q Egress Wireless Queuing Configuration Example on a Catalyst 3850*

```
! This section configures egress wireless queuing and a dual priority queue
C3850-2(config)# policy-map port_child_policy
C3850-2(config-pmap)# class non-client-nrt-class
C3850-2(config-pmap-c)# bandwidth remaining ratio 7
 ! Non-real-time multicast traffic receives 7 percent of the remaining
 ! bandwidth after the priority queue has been serviced
C3850-2(config-pmap-c)# class RT1
C3850-2(config-pmap-c)# priority level 1
C3850-2(config-pmap-c) police rate percent 10 conform-action transmit     exceed-
action drop
 ! RT1 is set to priority level 1 and policed to drop at 10 percent
C3850-2(config-pmap-c-police)# class RT2
```

```
C3850-2(config-pmap-c)# priority level 2
C3850-2(config-pmap-c)# police rate percent 20 conform-action transmit exceed-action
drop
 ! RT1 is set to priority level 2 and policed to drop at 20 percent
C3850-2(config-pmap-c-police)# class class-default
C3850-2(config-pmap-c)# bandwidth remaining ratio 63
 ! DEFAULT receives 63 percent of the remaining
 ! bandwidth after the priority queue has been serviced
```

Note The *egress wireless queuing configuration is applied automatically* by the WCM based on a wireless port negotiation. When it is determined that a port is wireless in nature, the static system-defined policy map port_child_policy (as is the non-client-nrt-class system policy map) is applied along with the radio-level shaper. If the priority levels are not turned on at the wireless port level, they will not be usable at the SSID-level or client-level policies.

Policy 8: SSID Bandwidth Allocation Between Guest and Enterprise SSIDs (SSID Policy to Separate Bandwidth Distribution)

The QoS policy will be applied downstream at the SSID, as detailed in Example 21-8.

Example 21-8 *Tifosi Case Study: Access-Edge Design for Differentiated Bandwidth Distribution*

```
! This section configures a bandwidth ratio of 30 for the guest SSID
C3850(config)# Policy-map TRUST-BW-GUEST
C3850(config-pmap)# class class-default
 ! This configures TRUST of DSCP and marks UP toward the AP – only necessary if
global trust has not been enabled
C3850(config-pmap-c)# set dscp dscp table dscp2dscp
C3850(config-pmap-c)# set wlan user-priority dscp table dscp2up
C3850(config-pmap-c)# bandwidth remaining ratio 30

 ! This section configures a bandwidth ratio of 70 for the enterprise SSID
C3850(config)#Policy-map TRUST-BW-ENTERPRISE
C3850(config-pmap)# class class-default
 ! This configures TRUST of DSCP and marks UP toward the AP – only necessary if
global trust has not been enabled
C3850(config-pmap-c)# set dscp dscp table dscp2dscp
C3850(config-pmap-c)# set wlan user-priority dscp table dscp2up
C3850(config-pmap-c)# bandwidth remaining ratio 70

 ! Applies policy to wireless lan guest
```

```
C3850(config)# wlan guest 33
C3850(config-wlan)# service-policy out TRUST-BW-GUEST

 ! Applies policy to wireless lan enterprise
C3850(config)# wlan enterprise 34
C3850(config-wlan)# service-policy out TRUST-BW-ENTERPRISE
```

Policy 9: CT 5760 Wireless LAN Controller Uplink Ports

The QoS policy will be applied to the CT 5760 wired ports and is detailed in Example 21-9. Figure 21-5 shows the graphical representation of the wireless eight-class queuing model.

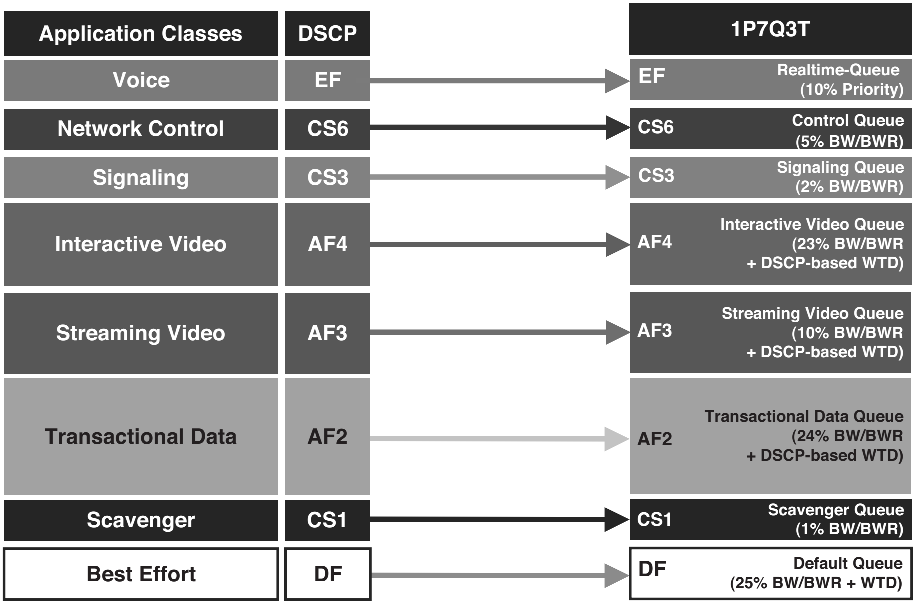

Application Classes
DSCP
1P7Q3T
Voice
EF
EF
Realtime-Queue (10% Priority)
Network Control
CS6
CS6
Control Queue (5% BW/BWR)
Signaling
CS3
CS3
Signaling Queue (2% BW/BWR)
Interactive Video
AF4
AF4
Interactive Video Queue (23% BW/BWR + DSCP-based WTD)
Streaming Video
AF3
AF3
Streaming Video Queue (10% BW/BWR + DSCP-based WTD)
Transactional Data
AF2
AF2
Transactional Data Queue (24% BW/BWR + DSCP-based WTD)
Scavenger
CS1
CS1
Scavenger Queue (1% BW/BWR)
Best Effort
DF
DF
Default Queue (25% BW/BWR + WTD)

Figure 21-5 *Tifosi Case Study: CT 5760 Eight-Class (1P7Q3T) Egress Queuing Model*

The corresponding configuration for 1P7Q3T egress queuing on the Cisco 5760 WLC is shown in Example 21-9.

Example 21-9 *1P7Q3T Egress Wired Queuing Configuration Example on a Cisco 5760 WLC*

```
 ! This section configures buffers, thresholds and a single priority queue
CT5760(config-pmap-c)# policy-map 1P7Q3T
CT5760(config-pmap)# class REALTIME-QUEUE
```

```
CT5760(config-pmap-c)# priority level 1
CT5760(config-pmap-c)# police rate percent 10
 ! PRIORITY is assigned strict priority with a policer rate set to 10
 ! percent of line rate
CT5760(config-pmap-c-police)# class CONTROL-QUEUE
CT5760(config-pmap-c)# bandwidth remaining percent 5
CT5760(config-pmap-c)# queue-buffers ratio 10
 ! CONTROL is assigned 5 percent bandwidth remaining after the
 ! priority queue is serviced, and allowed up to an additional 1/10th of
 ! buffer pool
CT5760(config-pmap)# class SIGNALING-QUEUE
CT5760(config-pmap-c)# bandwidth remaining percent 2
 ! SIGNALING is assigned 2 percent bandwidth remaining after the
 ! priority queue is serviced
CT5760(config-pmap-c)# class INTERACTIVE-VIDEO-QUEUE
CT5760(config-pmap-c)# bandwidth remaining percent 23
CT5760(config-pmap-c)# queue-buffers ratio 10
CT5760(config-pmap-c)# queue-limit dscp af43 percent 80
CT5760(config-pmap-c)# queue-limit dscp af42 percent 90
CT5760(config-pmap-c)# queue-limit dscp af41 percent 100
 ! INTERACTIVE is assigned 23 percent bandwidth remaining after the
 ! priority queue is serviced, allowed up to an additional 1/10th of
 ! buffer pool, and provides for 3 thresholds for 3 DSCP values
CT5760(config-pmap-c)# class STREAMING-VIDEO-QUEUE
CT5760(config-pmap-c)# bandwidth remaining percent 10
CT5760(config-pmap-c)# queue-buffers ratio 10
CT5760(config-pmap-c)# queue-limit dscp af33 percent 80
CT5760(config-pmap-c)# queue-limit dscp af32 percent 90
CT5760(config-pmap-c)# queue-limit dscp af31 percent 100
 ! STREAMING is assigned 10 percent bandwidth remaining after the
 ! priority queue is serviced, allowed up to an additional 1/10th of
 ! buffer pool, and provides for 3 thresholds for 3 DSCP values
CT5760(config-pmap-c)# class TRANSACTIONAL-DATA-QUEUE
CT5760(config-pmap-c)# bandwidth remaining percent 24
CT5760(config-pmap-c)# queue-buffers ratio 10
CT5760(config-pmap-c)# queue-limit dscp af23 percent 80
CT5760(config-pmap-c)# queue-limit dscp af22 percent 90
CT5760(config-pmap-c)# queue-limit dscp af21 percent 100
 ! TRANSACTIONAL is assigned 24 percent bandwidth remaining after the
 ! priority queue is serviced, allowed up to an additional 1/10th of
 ! buffer pool, and provides for 3 thresholds for 3 DSCP values
CT5760(config-pmap-c)# class SCAVENGER-QUEUE
CT5760(config-pmap-c)# bandwidth remaining percent 1
CT5760(config-pmap-c)# queue-buffers ratio 10
 ! SCAVENGER is assigned 1 percent bandwidth remaining after the
 ! priority queue is serviced, and allowed up to an additional 1/10th of
 ! buffer pool
CT5760(config-pmap-c)# class class-default
CT5760(config-pmap-c)# bandwidth remaining percent 25
CT5760(config-pmap-c)# queue-buffers ratio 25
 ! DEFAULT is assigned 25 percent bandwidth remaining after the
 ! priority queue is serviced, and is allowed up to an additional 1/4th
 ! of the buffer pool


 ! This section configures interface egress queuing parameters
CT5760(config)# interface gigabitEthernet 1/0/13
CT5760(config-if)# service-policy out 1P7Q3T
```

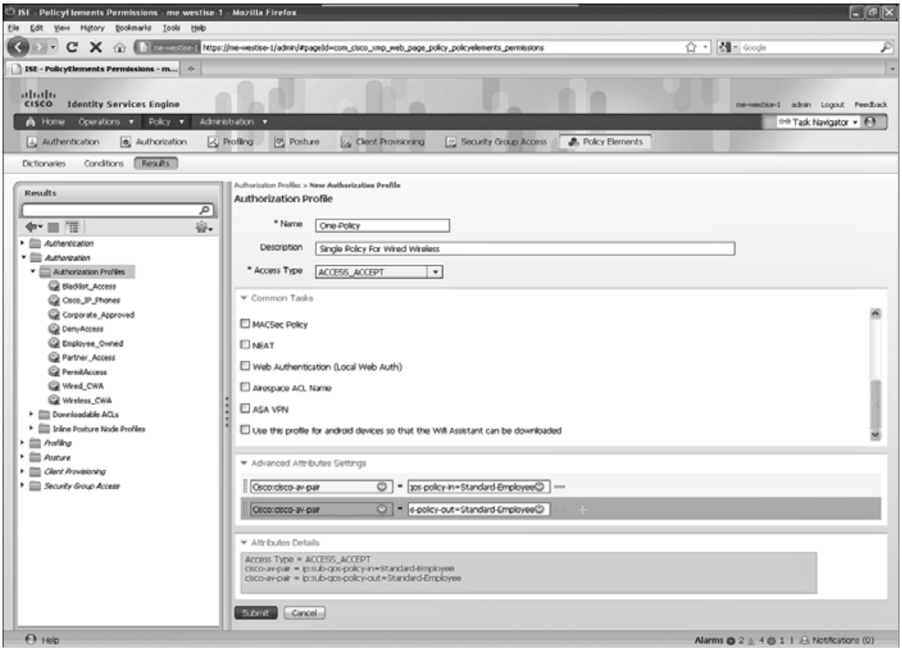
Note The eight-class model 1P7Q3T configuration in Example 21-9 can easily be expanded to cover two priority queues, as illustrated in Chapter 20. The main difference is the separation of voice and video traffic and the condensing of Bulk and Scavenger classes. However, these are relatively minor modifications.

Note Wired uplinks on the 5760 are not limited to four queues because these are not directly attached to wireless APs. The same wired queuing policy is leveraged here as on the wired ports of the Catalyst 3850 for simplicity.

Cisco Identity Services Engine

Tifosi's management tool of choice is Cisco ISE. As shown in Figure 21-6, you can use the cisco-av-pair: ip:sub-qos-policy-in/out=<policy name> to provide QoS treatment to individuals that are authorized. The full QoS policy is configured in global CLI on the Catalyst 3850. The RADIUS server (ISE) pushes only the policy name to the Catalyst 3850, where this policy will be applied to the specific client. Figure 21-6 shows the GUI of ISE. This illustrates the setup of the Cisco-av-pair necessary to push the MQC policy name to the Catalyst 3850 for an individual user or user group.

If a client decides to roam and mobility is configured, the policy migrates with the client. This migration takes place by passing policy name information associated with the client to the converged access device to which the client is roaming.



Authorization Profiles > New Authorization Profile

Authorization Profile

* Name: One-Policy

Description: Single Policy For Wired Wireless

* Access Type: ACCESS_ACCEPT

Common Tasks

□ MACSec Policy

□ NEAT

□ Web Authentication (Local Web Auth)

□ Airespace ACL Name

□ ASA VPN

□ Use this profile for android devices so that the Wifi Assistant can be downloaded

Advanced Attributes Settings

Cisco:cisco-av-pair = qos-policy-in=Standard-Employee

Cisco:cisco-av-pair = s-policy-out=Standard-Employee

Attributes Details

```
Access Type = ACCESS_ACCEPT
cisco-av-pair = ip:sub-qos-policy-in=Standard-Employee
cisco-av-pair = ip:sub-qos-policy-out=Standard-Employee
```

Figure 21-6 *Tifosi Case Study: ISE QoS Policy Name Screenshot*

Summary

This chapter continued the case study example of Tifosi Software and applied their strategic eight-class end-to-end QoS model to the converged access switching platforms (Catalyst 3850, CT 5760) in their multitier enterprise campus network.

Campus access layer policies were presented for untrusted, trusted, and conditionally trusted endpoints, as were explicit classification and marking policies, along with egress queuing policies for the Catalyst 3850 and CT 5760.

Additional Reading

Cisco Enterprise Medianet Campus QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html

Medianet Campus QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampusaag.html

Medianet Catalyst 3560/3850 QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampuscat3xxxaag.html

Cisco Catalyst 3850 and Catalyst 3560-X Switch Software Configuration Guide, Cisco IOS Release 15.0(2)SE: QoS Configuration Guide: http://www.cisco.com/en/US/docs/switches/lan/catalyst3850x_3560x/software/release/15.0_2_se/configuration/guide/swqos.html

Medianet Catalyst 4500 QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampuscat4500aag.html

Cisco Catalyst 4500 Series Switch Software Configuration Guide, Release IOS XE 3.3.0SG and IOS 15.1(1)SG: QoS Configuration Guide: http://www.cisco.com/en/US/docs/switches/lan/catalyst4500/15.1/XE_330SG/configuration/guide/qos_mrg.html

Medianet Catalyst 6500 QoS Design At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoscampuscat6500aag.html

Cisco Catalyst 6500 Series Switch IOS Release 15.1SY Supervisor Engine 2T Software Configuration Guide: QoS Configuration Guide: http://www.cisco.com/en/US/partner/docs/switches/lan/catalyst6500/ios/15.1SY/config_guide/sup2T/qos_overview.html

This page intentionally left blank

Chapter 22

Data Center QoS Design Considerations and Recommendations

The primary role of quality of service (QoS) in data center networks is—first and foremost—to manage packet loss. In a Gigabit Ethernet (GE)/10GE/40GE/100GE data center network, it takes only a few milliseconds of congestion to cause instantaneous buffer overruns resulting in severe packet drops. Furthermore, hardware buffering and queuing alone cannot meet the needs of specific data center protocols—such as Remote Direct Memory Access over Converged Ethernet (RoCE) or Fibre Channel over Ethernet (FCoE)—which demand zero-drop/lossless service. To meet such requirements, additional data center bridging (DCB) technologies must also be worked into the data center QoS designs.

A secondary role of QoS in the data center is to classify, mark, and manage application protocols at the physical/virtual access edge.

Therefore, several of the strategic QoS design principles discussed in Chapter 11, "QoS Design Principles and Strategies," apply to data center QoS designs, including the following:

- **Always perform QoS in hardware rather than software when a choice exists:** Cisco Nexus switches (with the exception of the Nexus 1000V virtual switch) perform QoS in dedicated hardware application-specific integrated circuits (ASICs) and therefore do not tax their main CPUs while administering QoS policies. You can therefore apply and efficiently enforce even complex QoS policies at GE/10GE/40GE and 100GE line rates on these platforms.
- **Classify and mark applications as close to their sources as technically and administratively feasible:** This principle promotes end-to-end differentiated services/per-hop behaviors. Some application servers mark class of service (CoS) or differentiated services code point (DSCP) values on their network interface cards (NICs), and therefore (for the most part) these markings may be trusted. This is especially true when considering that a degree of approval by the IT department has already been met to physically locate these application servers within the data center. However, cases

may exist where the server-based markings are incongruent with the strategic end-to-end QoS models in place and therefore may need to be re-marked.

- **Police unwanted traffic flows as close to their sources as possible:** The networking department may want to place limits on specific application flows, and therefore these may be policed at their sources—either to re-mark these in accordance with standards-based rules or to drop them outright.
- **Enable queuing policies at every node where the potential for congestion exists, regardless of how rarely this in fact may occur:** This principle applies to data center designs where oversubscription ratios create the potential for instantaneous congestion. Because of the multigigabit speeds of data center networks, queuing buffers can fill almost instantaneously, and therefore it is recommended to enable ingress and egress queuing policies throughout such data center designs.
- **Protect the control plane by enabling control plane policing:** This is an optional step to use QoS tools to improve the security posture of the data center network infrastructure.

Data Center Architectures

Recommendation:

- Understand various data center architectures and their respective QoS requirements.

There is no one-size-fits-all data center design. In addition, the overall architecture of the data center will significantly influence the QoS designs associated with it. Let's consider a few types of data center architecture and their respective QoS requirements, including the following:

- High-performance trading (HPT) data center architecture
- Big data architectures, including high-performance computing (HPC), high-throughout computing (HTC), and grid data center architectures
- Virtualized multiservice data center (VMDC) architecture
- Secure multitenant data center (SMDC) architecture
- Massively scalable data center (MSDC) architecture

The sections that follow provide a brief overview of and highlight the QoS design implications for each of these types of data center designs.

High-Performance Trading Data Center Architectures

Recommendation:

- Recognize why HPT data center architectures have minimal QoS requirements.

HPT data centers are used by financial enterprises because their focus is on increasing order execution speeds so that they can gain competitive advantages and capture opportunities during periods of market volatility. Performance in HPT environments is measured not in milliseconds, but in microseconds and nanoseconds.

Therefore, HPT data centers often feature low-latency platforms, such as the Cisco Nexus 3000 series data center switch, and their entire data center architectural approach is focused on minimizing or eliminating oversubscription. Figure 22-1 shows an example of HPT data center architecture.

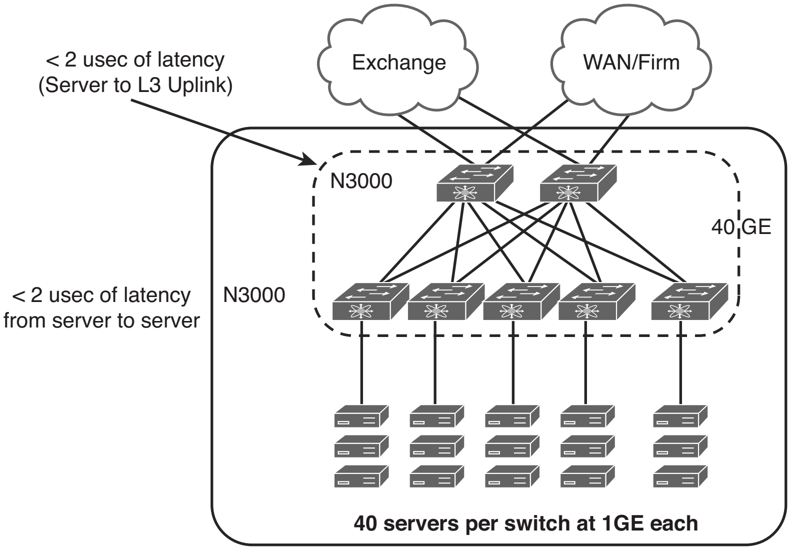


Figure 22-1 *An HPT Data Center Architecture*

In a low-blocking/nonblocking HPT data center architecture, QoS plays a minimal role (because the design goal of the architecture is essentially to render the role of QoS unnecessary).

Big Data (HPC/HTC/Grid) Architectures

Recommendation:

- Recognize why big data architectures use similar QoS tools as campus architectures.

The goal of such big data architectures is to process large and complex data sets that are too difficult to handle by traditional data processing applications.

The challenges addressed by these data centers include the capturing, indexing, storing, searching, sharing, transferring, and analyzing of massive data sets. The trend to larger data sets is due to the additional information derivable from the analysis of a single large set of related data. This allows for correlations to be found for such diverse challenges as preventing diseases, combating crime, predicting road-traffic conditions, or spotting market trends. Therefore, these data centers are used by universities, research and development organizations, governments, and the private sector.

Big data architectures can also be broken down further into the following categories:

- **High-performance computing (HPC):** HPC tasks are characterized by large amounts of computing power used for short periods of time. HPC environments are often measured in terms of floating-point operations per second (FLOPS).
- **High-throughput computing (HTC):** Like HPC, HTC tasks also require large amounts of computing power, but for much longer periods of time (months and years rather than hours and days), and is therefore more focused—not on operations per second—but rather operations per month or year.
- **Grid:** Grid computing is the federation of computer resources from multiple locations to reach a common goal. The grid can be thought of as a distributed system with noninteractive workloads that involve a large number of files. What distinguishes grid computing from conventional HPC systems is that grids tend to be more loosely coupled, heterogeneous, and geographically dispersed. Although a single grid can be dedicated to a particular application, commonly a grid is used for a variety of purposes.

Divide-and-conquer strategies can prove quite effective for several kinds of workloads that deal with massive amounts of data. For example, a single large workload can be divided or mapped into smaller subworkloads, and the results from the subworkloads can be merged, condensed, and reduced to obtain the final result. This is the approach for big data analysis used by Google's patented MapReduce methodology.

This approach has also been adapted into Apache's open source Hadoop methodology, but with the emphasis on using large clusters of inexpensive nodes built with general-purpose hardware for processing subworkloads.

With either approach, handling massive amounts of data requires storing massive amounts of data; therefore, distributed, cluster-based file systems are also required to store data storage.

Figure 22-2 illustrates an example of a big data data center architecture, highlighting cluster-based design.

In cluster-based implementations of big data architectures, both compute and storage reside within individual servers, which then form clusters. Therefore, from a QoS perspective, the tools and designs required to support big data architectures are similar to campus networks—because these focus mainly on extending trust and performing hardware queuing at every node.

However, big data data centers also benefit from throughput optimization, as is discussed in the "Data Center Transmission Control Protocol" section to follow.

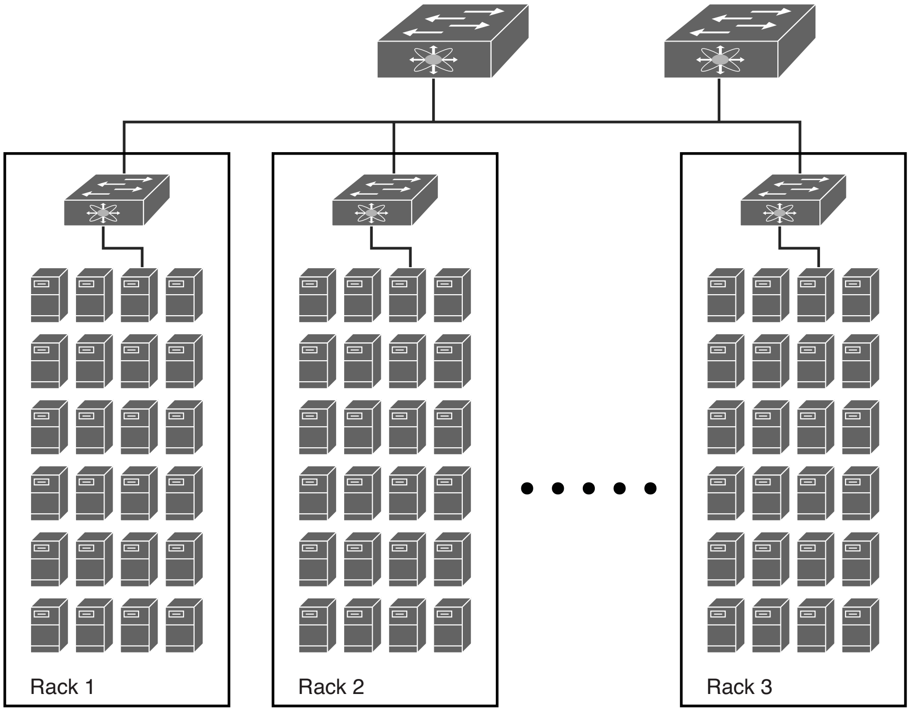


Figure 22-2 *A (Cluster-Based) Big Data Data Center Architecture*

Virtualized Multiservice Data Center Architectures

Recommendation:

- Understand the unique QoS requirements of compute and storage virtualization, including provisioning a lossless Ethernet service.

Virtualization and cloud technologies have shifted the overall architecture of the data center in the following ways:

- Applications no longer map to a physical server (or cluster of servers).
- Storage no longer is tied to a local physical disk (or array).
- Network infrastructure is no longer tied to hardware.

In contrast to the preceding list, virtualization is often the first step at leveraging pools of compute, storage, and networking resources to optimize the underlying infrastructure. Cloud technologies can thus deliver such infrastructures and platforms as a service.

Virtualized multiservice data center (VMDC) architectures leverage these cloud-based technologies to unifying compute, storage, networking, virtualization, and management resources into a single fabric-based platform designed to increase operating efficiencies, simplify operations, and provide business agility.

Figure 22-3 illustrates an example of VMDC architecture.

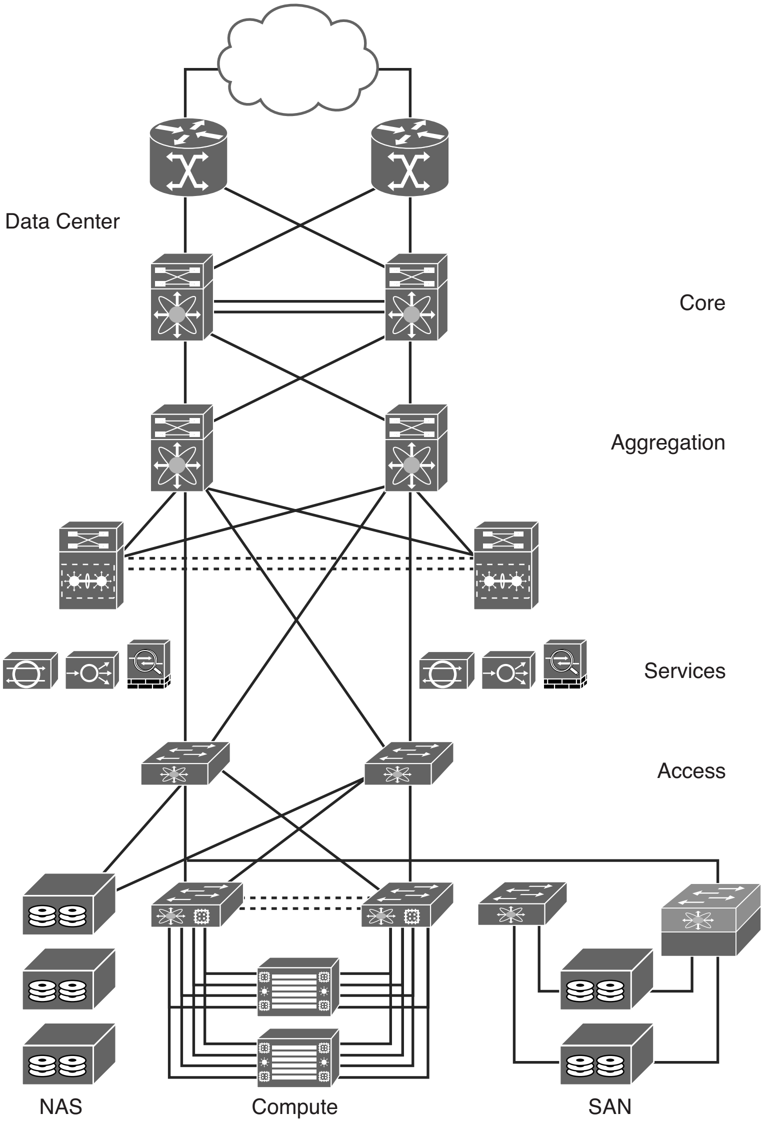


Figure 22-3 *A Virtualized Multiservice Data Center Architecture*

The VMDC presents a major paradigm shift in QoS design, as lossless compute and storage virtualization protocols—such as RoCE and FCoE—are required to be supported, as are server virtualization protocols, such as Live Migration/vMotion.

Because VMDCs present the newest, most demanding, and most widely deployed challenges for QoS in the data center, these are the focus of the design chapters in this part of the book. In addition, VMDC serves as a foundation to the data center architectures that follow (that is, SMDC and MSDC).

Secure Multitenant Data Center Architectures

Recommendation:

- Realize how virtualization can be leveraged to support multitenants over a common infrastructure and how this affects QoS designs.

The Secure Multitenant Data Center (SMDC) architecture leverages virtualization technologies with the additional objective of leveraging these to support multiple tenants via a single underlying compute, storage, and network infrastructure.

In the enterprise context, tenants can be departments of an organization that are operating as a private cloud. Each department may have different computational needs, applications, or storage requirements, while at the same time all the departments need to be managed and maintained under a single unified operational domain.

In the service provider context, tenants can be different enterprises that lease or use the service provider's cloud infrastructure in a shared public cloud environment. As in the private cloud case, each of these enterprises might have specific computation, application, and storage needs. In addition, the security and service level agreement (SLA) conformance of each tenant needs to be strictly enforced, so as to meet both the leasing enterprise's requirements and the hosting service provider's business requirements.

Figure 22-4 shows an example of SMDC architecture.

The QoS design requirements of SMDC are largely the same as that of VMDC, with the main exception being the marking models used: SMDC marking models identify not only various application/traffic classes but also various levels of customer tiers of service (for example, gold/silver/bronze), as discussed later in this chapter.

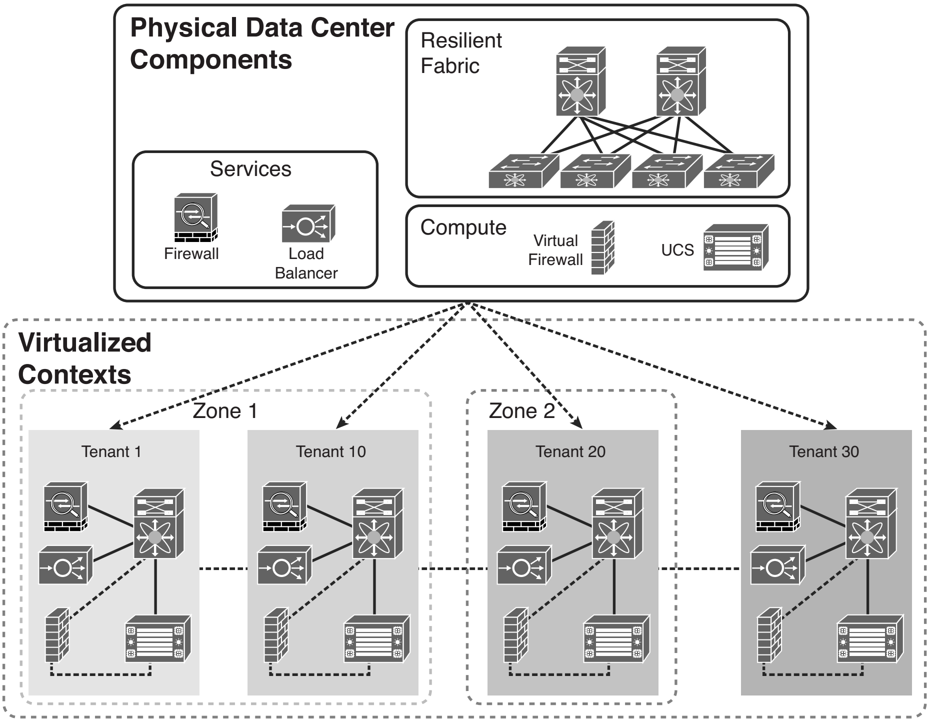


Figure 22-4 *A Secure Multitenant Data Center Architecture*

Massively Scalable Data Center Architectures

Recommendation:

- Understand how virtualized data centers can scale and how this affects the need to improve overall data transfer efficiencies.

Cisco's massively scalable data center (MSDC) is a framework that data center architects can use to build elastic data centers that host a few applications that are distributed across thousands of servers and that scale from department to Internet-scale audiences. Unlike a traditional data center, which hosts applications deployed in clusters, the MSDC data center is characterized by a few very large applications that are distributed over geographically distributed homogenous pools of compute and storage. In essence, these data centers behave less like hosting environments and more like highly optimized computers.

Figure 22-5 shows an example of MSDC architecture.

The MSDC is a cloud-based architecture, and so has the same QoS requirements as VMDC, but with the added requirement of maximizing throughput, because inefficiencies in this area are magnified in proportion to the (massive) scale of the data center.

Throughput—and more relevantly "goodput"—is discussed in the "Data Center Transmission Control Protocol" later in this chapter.

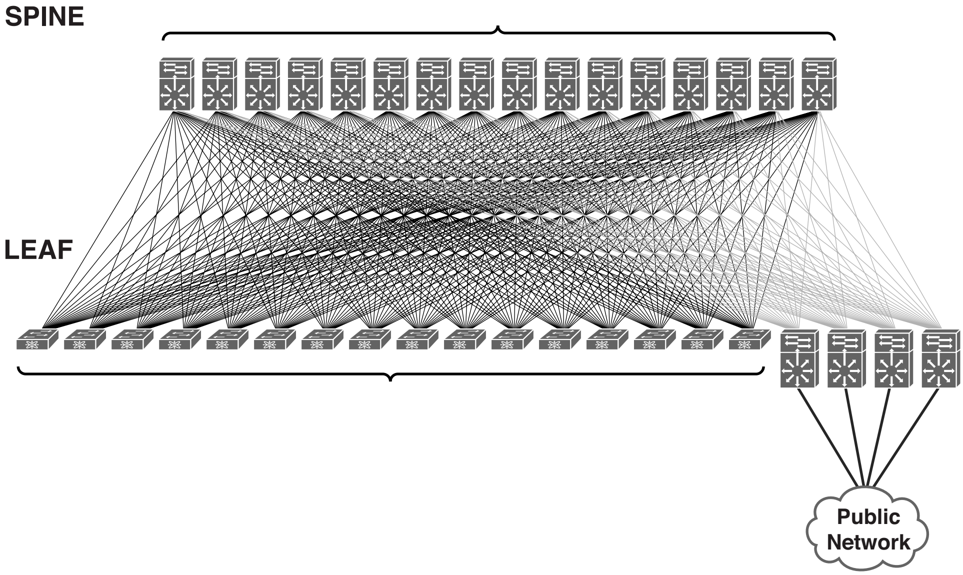


Figure 22-5 *A Massively Scalable Data Center Architecture*

Data Center QoS Tools

Recommendation:

- Understand the data center QoS toolset so that you can effectively leverage these in data center QoS designs.

Like campus switches, data center switches require QoS operations—including classification, marking, queuing, and dropping—to be performed in hardware to perform these operations at data center speeds.

In addition, virtualized data center models also have the unique challenge of converging storage-area networks (SANs) and Ethernet LANs into a single fabric. Such converged network architectures save costs by

- Reducing the number of switches required to support both types of network traffic
- Reducing the number of interfaces required
- Reducing cable complexity
- Reducing administration, operational complexity, and activities

Some SAN protocols, particularly Fibre Channel, require lossless transport. Although hardware QoS mechanisms go a long way to reducing drops due to instantaneous buffer congestion (as discussed in the chapters in Part III, "Campus QoS Design"), standard Ethernet cannot *guarantee* a completely lossless transport. To meet this unique challenge, the IEEE has developed extensions to Ethernet that fall under the data center bridging (DCB) toolset, as discussed next.

Following this DCB toolset discussion, an emerging goodput/throughput optimization algorithm that leverages the IP explicit congestion notification (ECN) QoS mechanism and which is gaining traction in big data and in MSDC contexts is discussed, namely Data Center Transmission Control Protocol (DCTCP).

Data Center Bridging Toolset

Recommendation:

- Become familiar with DCB tools such as PFC, ETS, congestion notification, and DCBX so that so that you can effectively leverage these to converge the SAN and LAN fabrics in the data center.

The IEEE formed a special working group called the (IEEE 802.1) Data Center Bridging Task Group to define enhancements to Ethernet to support the requirements of converged data center networks. These enhancements include the following:

- Priority flow control (IEEE 802.1Qbb)
- Enhanced transmission selection (IEEE 802.1Qaz)
- Congestion notification (IEEE 802.1Qau)
- DCB exchange (DCBX) (IEEE 802.1Qaz combined with 802.1AB)

Each of these DCB tools is discussed in turn. However, before doing so, it may be helpful to provide some context by considering a related legacy flow-control mechanism: Ethernet Flow Control (IEEE 802.3x).

Ethernet Flow Control: IEEE 802.3x

Recommendation:

- Understand how Ethernet flow control (EFC) can provide a lossless Ethernet service, in addition to understanding its limitations.

IEEE 802.3x EFC—defined in 1997—introduced the a concept of a PAUSE frame, which supports lossless service over Ethernet.

The way EFC works is if a sending station (either a server or a network switch) is transmitting faster than the receiving station can accept, the receiver issues a PAUSE frame back to the sender, which (in turn) forces a halt to the transmission for a specified period of time.

Figure 22-6 shows an EFC operation example, in addition to the limitations this mechanism can impose.

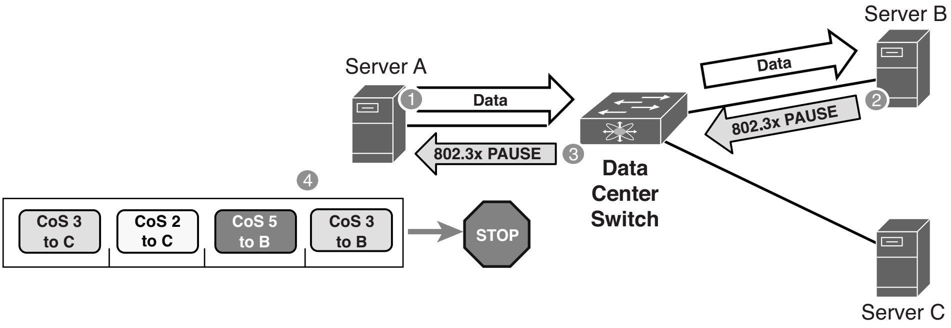


Figure 22-6 *EFC Operation Example*

In Figure 22-6, the following operations occur:

1. Server A is sending traffic to Server B. For the sake of this example, let's say this traffic is marked CoS 3 (which is the default for FCoE).
2. At some point, Server B begins to be overwhelmed and sends a return PAUSE frame to the network switch. Let's assume that in this case the network switch has no buffering capacity (so as to simplify the example and to highlight—in subsequent examples—the critical value of switch buffering in DCB networks).
3. In this case, because of this assumed lack of network switch buffering capacity, the network switch would in turn immediately send a PAUSE frame of its own to Server A.
4. Server A must temporarily stop all transmission. At the point in time when transmission is halted, let's say that Server A has additional traffic to transmit, including the following:

   - A frame marked CoS 3 to transmit to Server B
   - A high-priority frame marked CoS 5 to transmit to Server B
   - A frame marked CoS 2 to transmit to Server C (which is completely available to receive flows)
   - A frame marked CoS 3 to transmit to Server C (which again, is completely available to receive flows)

   However, because of the required observance of the PAUSE frame, none of these additional flows may be transmitted until the pause timer has expired—even though these additional flows may be marked to a higher-priority CoS value or intended for available receiving stations. This resulting delay to other traffic flows is referred to as head-of-line (HoL) blocking and represents a serious functional limitation of EFC.

Priority Flow Control: IEEE 802.1Qbb

Recommendation:

- Appreciate the improvements that priority flow control (PFC) has over EFC and how to leverage it to provide a lossless service for storage protocols.

In 2008, the Data Center Bridging Task Force introduced an enhancement to EFC called priority flow control.

PFC provides a link-level flow control mechanism that can be controlled independently for each (802.1p) CoS priority. The goal of this mechanism is to ensure zero frame loss due to congestion in DCB networks, while at the same time mitigating some of the HoL blocking scenarios inherent to EFC.

With PFC, the PAUSE frame includes (and is associated with) a specific CoS value. Sending and receiving stations provision receive and transmit buffers/queues for each CoS value. These respective transmit and receive queues form "virtual lanes" (or virtual links), with one virtual lane per CoS value. These virtual lanes can be paused and restarted independently by PFC. This approach enables the network to create a no-drop CoS for an individual virtual lane that can coexist with other traffic types on the same interface. One or more CoS values may receive a no-drop service.

If a station is unable to receive markings of a specific CoS value (because of overwhelming receive buffers), it can send a PAUSE frame for that specific CoS value in the opposite direction as the flow of data, as shown in Figure 22-7.

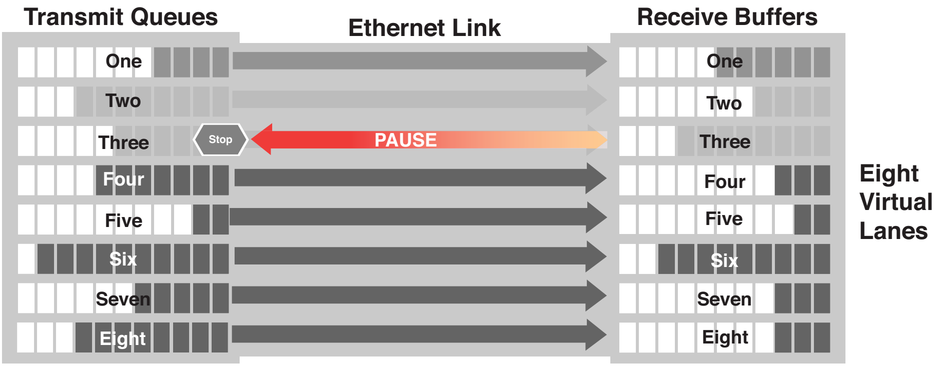


Figure 22-7 *PFC Transmit Queues and Receive Buffers*

The improvement PFC brings to mitigating HoL blocking scenarios can be seen by revisiting the previous example, as illustrated in Figure 22-8.

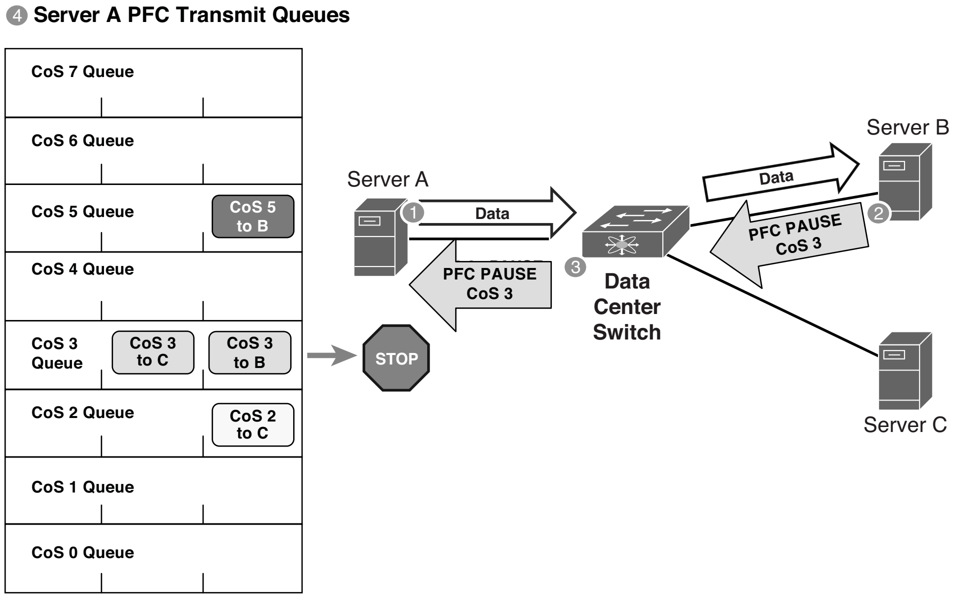


Figure 22-8 *PFC Operation Example*

In Figure 22-8, the following operations occur:

1. Server A is sending CoS 3 traffic to Server B.
2. At some point, Server B's CoS 3 receive buffers begin filling to capacity and it sends a return PFC PAUSE (CoS 3) frame to the network switch. As before, let's assume that the network switch has no buffering capacity.
3. Because this assumed lack of network switch buffering capacity, the network switch immediately sends a PFC PAUSE (CoS 3) frame of its own to Server A.
4. Server A must temporarily stop transmitting CoS 3. At the point in time when transmission is halted, let's say that Server A has the same additional traffic to transmit as in the previous example.

   However, unlike the previous example, instead of having all outgoing traffic enqueued in a single first in, first out (FIFO) transmit queue, Server A has eight separate transmit queues: one for each CoS value. Therefore, Server A can still send the CoS 5 frame to Server B, in addition to the CoS 2 frame to Server C. In fact, the only flow suffering from HoL blocking in this particular case is the CoS 3 frame intended for Server C (which is blocked behind the paused CoS 3 frame intended for Server B).

As you can see, PFC not only enables a per-CoS lossless service, but it also significantly reduces HoL blocking scenarios. However, further improvement can still be made via switch buffering architectures, which are discussed next.

PFC is enabled within NX-OS with the **pause** command.

Note PFC is also supported on Cisco Unified Computing System (UCS) platforms, in addition to VMware ESX/ESXi platforms.

Skid Buffers and Virtual Output Queuing

Recommendations:

- Know why skid buffers are required.
- Understand how virtual output queues are more efficient and effective in managing switch buffering in data center environments.

Although technically not part of the DCB toolset, skid buffers and virtual output queuing (VOQ) play an integral part in supporting a lossless transport DCB network, in addition to further minimizing HoL blocking. Therefore, they merit consideration in the context of this discussion.

Buffer management is critical to the operation of PFC. For example, if transmit or receive buffers are allowed to overflow, the transport will no longer be lossless. In addition to provisioning buffers to each virtual lane in both the transmit and receive directions, the network switch needs to have sufficient buffering capacity to

- Store frames sent during the time it takes to send the PFC pause frame across the network between stations
- Store frames that are already in transit when the sender receives the PFC pause frame

The buffers allocated to accommodate these two related scenarios are called *skid buffers* and are usually engineered on a per-port basis in hardware in the ingress direction.

In addition, in the data center the primary reason end stations (servers) become overwhelmed by transmissions is not because of speed mismatches, but rather because of east-west many-to-one traffic flows, called incast flows. Incast flows are particularly common in big data and MSDC architectures.

To recap, here are a few networking flow descriptions:

- **Unicast** describes one-to-one flows.
- **Broadcast** describes one-to-all flows.
- **Multicast** describes one-to-many flows.
- **Incast** describes many-to-one flows.

Figure 22-9 shows an incast flow, which demonstrates how switch buffering can further reduce HoL blocking.

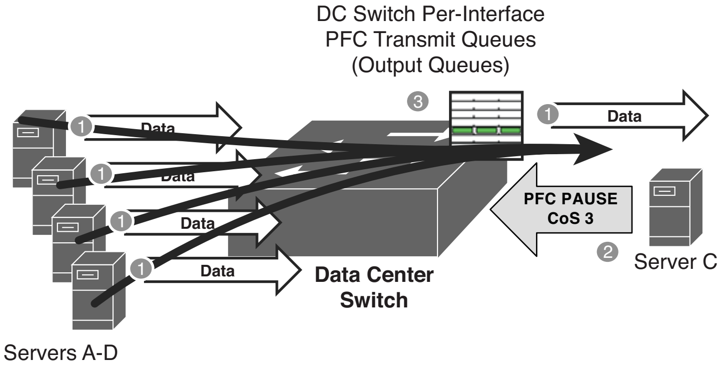


Figure 22-9 *Incast Example with PFC and Switch Egress Port Buffering*

In Figure 22-9, Servers A through D are sending incast traffic flows marked CoS 3 toward Server E, which—when overwhelmed—generates a PFC PAUSE (CoS 3) frame toward the network switch. In this example, the network switch has buffering capabilities and is able to absorb the overflow (at least for a while). When the per-port output buffers of the switch fill to capacity, it must (in turn) send PFC PAUSE (CoS 3) frames toward the sending stations.

As also shown in Figure 22-9, in an incast scenario, the network switch egress port becomes an aggregation node. Therefore, to accommodate such flows, the output queuing buffers at this node need to be relatively deep (to accommodate *n* number of incast flows).

However, a more efficient way to handle data center switch buffering (without requiring excessively deep per-port output buffers) is to force the congestion/queuing to occur *before* the ingress traffic reaches the switch fabric. This approach is called virtual output queuing.

With VOQ, congestion is artificially induced within the switching system (primarily, though not exclusively) before the switching fabric, as shown in Figure 22-10.

Note VOQ should not be confused with ingress queuing. Ingress queuing deals with *actual* congestion scenarios due to backplane/fabric oversubscription (that is, total ingress bandwidth exceeding backplane/fabric capacity) and is governed by *ingress* queuing policies; VOQ, in contrast, generates a *virtual* congestion scenario at a node before the switching fabric, and is governed by *egress* queuing policies.

The advantage of VOQ versus physical egress per-port buffering is that significantly fewer buffers are required to handle incast scenarios with VOQ, as shown in Figure 22-10.

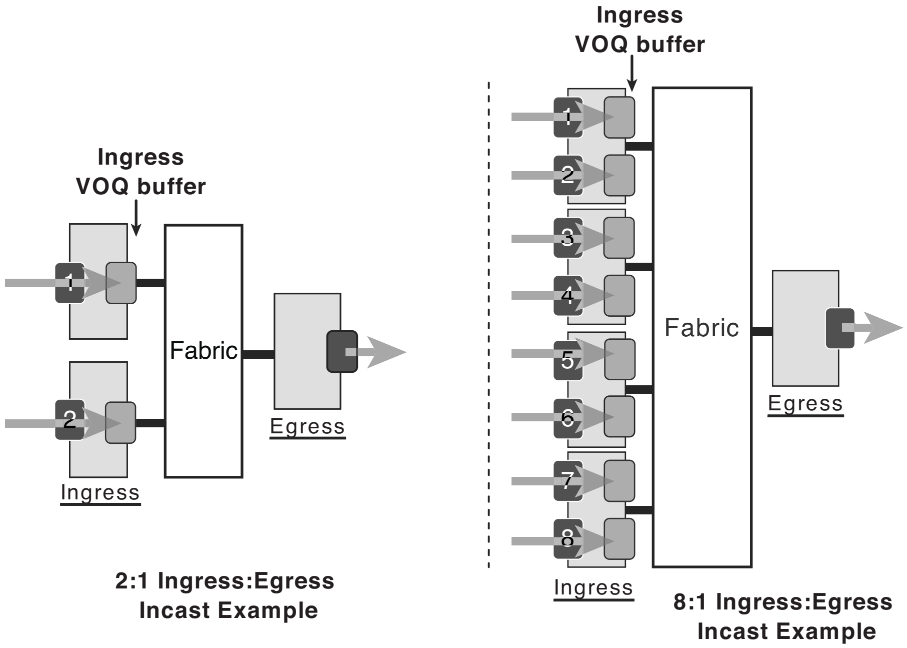


Figure 22-10 *VOQ Buffering Versus Egress Port Buffering*

As shown in Figure 22-10, if the same amount of egress buffering capacity were reallocated as VOQs, the total amount of switch buffering capacity would increase directly with the number of incast flows. (For example, a 2:1 incast flow would have twice the number of buffers with VOQ, an 8:1 flow would have eight times the number of buffers, and so on.)

A VOQ hardware architecture absorbs congestion at every ingress port contributing to the egress congestion, thus optimizing switch buffering capacity to accommodate incast flows. Furthermore, with VOQs excess traffic does not consume fabric bandwidth only to be dropped at an egress port.

Cisco Nexus data center switching platforms employ VOQ architectures, as discussed in more detail on a per-platform basis in the design chapters to follow.

Enhanced Transmission Selection: IEEE 802.1Qaz

Recommendation:

- Understand how enhanced transmission selection (ETS) can improve overall bandwidth efficiency of a DCB network.

Another Ethernet enhancement that the Data Center Bridging Task Force put forward is enhanced transmission selection, which provides a common management framework for the assignment and management of bandwidth to virtual lanes.

Extending the virtual lane concept, a DCB-enabled NIC (also called a converged network adaptor [CNA]) leverages ETS to provision virtual interface queues—one for each virtual lane. Each virtual interface queue is accountable for managing its allotted bandwidth for its traffic group, but has flexibility within the group to dynamically manage traffic. For instance, if a virtual lane is not using its allotted bandwidth, ETS can dynamically reassign unused bandwidth to another virtual lane that may require it.

ETS virtual interface queues can be serviced as follows:

- **Priority:** A virtual lane can be assigned a strict-priority service.
- **Guaranteed bandwidth:** A virtual lane can be guaranteed a percentage of the physical link bandwidth.
- **Best effort:** A default virtual lane service.

Figure 22-11 illustrates ETS operation over a 10GE interface with three virtual lanes: one each for storage traffic (CoS 3), HPC traffic (CoS 2), and LAN traffic (CoS 0). Each virtual lane is assigned a specified bandwidth percentage. However, in time slot t3, LAN traffic can use the spare/unused bandwidth available from the server cluster virtual lane.

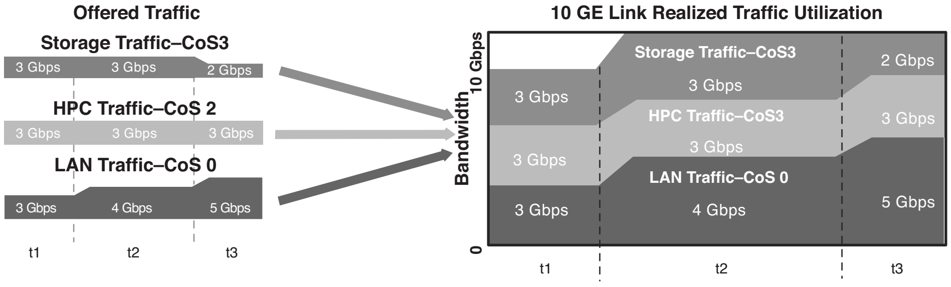


Figure 22-11 *ETS Operation Example*

Congestion Notification: IEEE 802.1Qau

Recommendation:

- Become familiar with 802.1Qau congestion notification capabilities and operation.

IEEE 802.1Qau congestion notification is a Layer 2 traffic management system that pushes congestion to the edge of the network by instructing rate limiters to shape the traffic that is causing congestion.

Congestion is measured at the *congestion point*, and if congestion is encountered, rate limiting, or backpressure, is imposed at the *reaction point* to shape traffic and reduce the effects of the congestion on the rest of the network. In this architecture, an aggregation-level switch (acting as a congestion point) can send control frames to two access-level switches (acting as reaction points) asking them to throttle back their traffic, as shown in Figure 22-12.

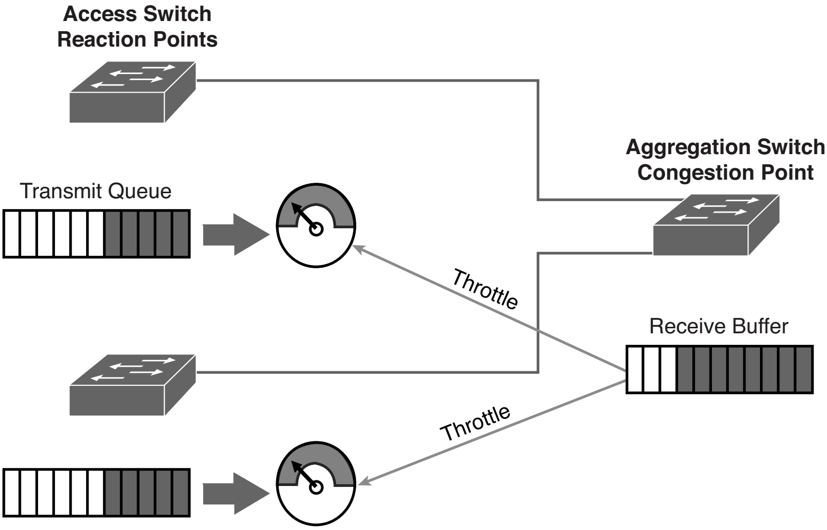


Figure 22-12 *Congestion Notification Operation Example*

Data Center Bridging Exchange: IEEE 802.1Qaz + 802.1AB

Recommendation:

- Recognize the role that the Data Center Bridging Exchange (DCBX) protocol plays in the DCB network.

DCBX falls under the same IEEE umbrella as ETS (802.1Qaz).

DCBX is a discovery and capability exchange protocol used to discover peers and exchange configuration information between DCB-compliant bridges. DCBX leverages functionality provided by IEEE 802.1AB (Link Layer Discovery Protocol [LLDP]).

DCBX capabilities include the following:

- DCB peer discovery
- Mismatched configuration detection
- DCB link configuration of peers

The following DCB parameters can be exchanged with DCBX:

- PFC
- ETS

- Congestion notification
- Applications
- Logical link-down
- Network interface virtualization

DCBX allows the automatic exchange of Ethernet and DCB parameters, in addition to discovery functions between switches and endpoints, as shown in Figure 22-13.

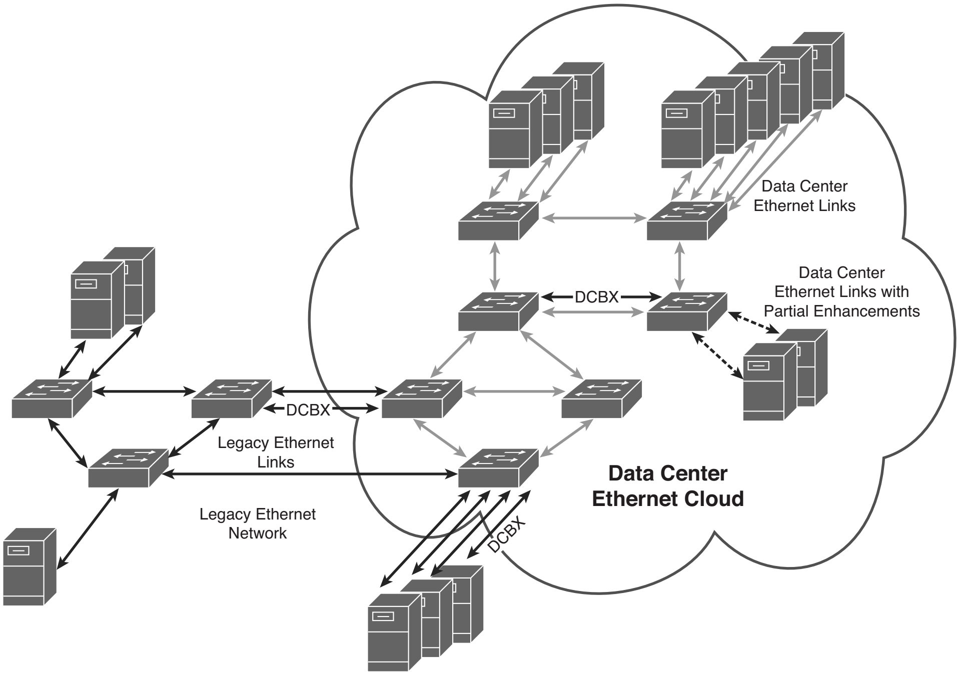


Figure 22-13 *DCBX Operation Example*

Data Center Transmission Control Protocol

Recommendation:

- Become familiar with Data Center Transmission Control Protocol (DCTCP) and how it interacts with IP ECN and improves overall goodput in data centers.

In addition to providing lossless service to storage protocols, another key design goal of data center QoS is to maximize goodput. *Goodput* is defined as application-level throughput, which excludes protocol overhead bits in addition to retransmitted data packets. Therefore, goodput is always lower than throughput (the gross bit rate that is transferred physically), which, in turn, is generally is lower than physical link bandwidth (because links are rarely utilized to 100 percent of capacity on a sustained basis).

One factor that significantly reduces goodput is transport layer flow control and congestion avoidance. Specifically, TCP slow-start significantly lowers goodput.

To address this, researchers at Stanford University and Microsoft developed an enhancement to TCP congestion control, leveraging IP ECN. Whereas the standard TCP congestion control algorithm is only able to detect the *presence* of congestion, DCTCP (using IP ECN) can gauge the *extent* of congestion.

Note IP ECN was previously discussed in Chapter 5, "Congestion Management and Avoidance Tools."

DCTCP is based on two key concepts:

- **React in proportion to the extent of congestion, not its presence:** This reduces variance in sending rates.
- **Mark ECN based on instantaneous queue length:** This enables fast feedback and corresponding window adjustments to better deal with bursts.

To illustrate how DCTCP works, consider the following two scenarios—each of which compares standard TCP operation with DCTCP—as summarized in Table 22-1. In each scenario, ten packets are sent: some with their IP ECN Congestion Experienced (CE) bits set to 1, some not.

Table 22-1 *TCP Operation Versus DTCP Operation*

IP ECN CE Bit Markings by Packet	TCP Response	DCTCP Response
1 - 0 - 1 - 1 - 1 - 1 - 0 - 1 - 1 - 1 (8 of 10 packets experienced congestion)	Cut window by 50 percent	Cut window by 40 percent
0 - 0 - 0 - 0 - 0 - 0 - 0 - 0 - 0 - 1 (1 of 10 packets experienced congestion)	Cut window by 50 percent	Cut window by 5 percent

In the first scenario, 80 percent of packets received have their CE bits set to 1 (indicating that congestion was experienced in transit). TCP will react by slow-starting and thus reducing the TCP window size by 50 percent. DCTCP, in turn, will *proportionally* reduce the window size by 40 percent (half of the amount of congestion that was experienced). In this scenario, the difference between standard TCP and DCTCP is not significant.

However, in the second scenario, only 10 percent of packets received have their CE bits tripped to 1. TCP again reacts by slow-starting and reducing the window size by 50 percent. TCP is thus reacting the same way as the previous scenario, oblivious to the *extent* of congestion and responding only to its *presence*. However, DCTCP again *proportionally* reduces its windows size, this time by only 5 percent (again, half the amount of congestion experienced). In this second scenario, the difference between TCP and DTCP becomes increasingly apparent.

Therefore, DCTCP not only handles bursts well (due to its quick responses to instantaneous ECN markings), but it also keeps queuing delays low (due to smoothing flows out)—both of which combine to achieve high goodput. And all with a relatively simple change to TCP behavior.

Note DCTCP is not featured in the design sections that follow simply because this feature was not yet available at the time of this writing (on the platforms discussed).

NX-OS QoS Framework

Recommendations:

- Recognize that Nexus OS (NX-OS) QoS—although similar to MQC—has several unique elements.
- Understand the different types of QoS objects within NX-OS, including **type qos**, **type queuing**, and **type network-qos**, in addition to the roles these serve.
- Know the role of **system qos** in applying global QoS parameters, in addition to how to override these.

Nexus OS implements QoS using a modified version of Modular QoS command-line interface (MQC). NX-OS still leverages MQC objects (such as class maps, policy maps, and service policy statements), but it adds some unique policy **type** designations:

- **type qos:** Defines MQC objects used for marking and policing.
- **type queuing:** Defines MQC objects used for queuing and scheduling, in addition to a limited set of the marking objects.
- **type network-qos:** Defines the characteristics of DCB network-wide QoS properties and should be applied consistently on all switches participating in the network.
- **control-plane:** Defines MQC objects used for control plane policing (CoPP).

In addition, NX-OS policies can be applied to targets such as interfaces and VLANs (and the control plane). However, there is a special target unique to NX-OS: **system qos**. Any policy applied to **system qos** is effectively applied globally to the whole switch, with the exception of when a different policy is applied to an interface or VLAN, in which case the latter will control.

Platform-specific guidelines and caveats of NX-OS QoS are discussed in the respective design chapters that follow.

Data Center QoS Models

Recommendation:

- Follow the five general steps of deploying QoS in the data center.

Generally speaking, there are five main steps to deploying QoS in the data center:

1. Amend/expand your enterprise strategic QoS marking model to accommodate the data center environment.
2. Apply various data center ingress QoS models (as required), including the following:
   - Trusted Server Model
   - Untrusted Server Model
   - Single-Application Server Model
   - Multi-Application Server Model
   - Server Policing Model
   - Lossless Transport Model
3. Configure ingress queuing.
4. Configure egress queuing.
5. (Optionally) Configure control plane policing.

Each of these steps is discussed further in the sections that follow.

Data Center Marking Models

Recommendation:

- Amend/expand your enterprise strategic QoS marking model to the data center environment.

While an RFC 4594-based 4-, 8- or 12-class strategic model serves well over nearly every other place in the network (PIN), the data center has some unique traffic characteristics, applications, and limitations that may affect the marking model used therein. Therefore, these strategic models may need to be adapted, expanded, or modified to this unique environment.

Considerations affecting the marking model to be used in the data center include the following:

- Data center applications and protocols
- CoS/DSCP marking
- CoS 3 overlapping considerations
- Application-based marking models
- Application- and tenant-based marking models

Each of these considerations is discussed in turn.

Data Center Applications and Protocols

Recommendations:

- Consider which applications/protocols are present in the data center that may not already be reflected in the enterprise QoS model and how these may be integrated.
- Consider which applications/protocols may not be present (or may have a significantly reduced presence) in the DC.

Several applications and protocols are unique (or predominately native) to the data center, including the following:

- **Compute virtualization protocols:**
  - **Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE)** is a protocol that supports the direct memory access of one computer into that of another—over the Converged Ethernet (that is, DCB) network—without involving either one's operating system. This permits high-throughput, low-latency networking, which is especially useful in massively parallel computer clusters. RoCE is a link layer protocol and hence allows communication between any two hosts in the same Ethernet broadcast domain. Recommendation: RoCE requires a lossless service via PFC. When implemented along with FCoE, it should be assigned its own no-drop class / virtual lane, such as CoS 4. In such a case, however, other applications marked to CoS 4 (such as video) would need to be reassigned so as to improve RoCE performance.
  - **Internet Wide Area RDMA Protocol (iWARP)** extends the reach of RDMA over IP networks. Recommendation: iWARP does not require a lossless service because it runs over TCP or Stream Control Transmission Protocol (SCTP), which provide reliable transport. Therefore, it can be marked to an unused DSCP/CoS combination (if one exists, depending on the strategic QoS model in use)—or it can be combined with internetwork control traffic (CS6/CoS 6) or network

control (CS7/CoS 7) or even another noncontrol traffic class—as long as additional bandwidth is provisioned accordingly to the combined class.

- **Virtual machine control and live migration protocols (VM control):** Virtual machines (VMs) require control traffic to be passed between hypervisors; such traffic may be marked by default or require explicit marking to be protected as control plane traffic. Recommendation: VM control traffic is control plane traffic for VMs and may be marked to CoS 6 or CoS 7, depending on the QoS model in use.

- **Live migration:** Refers to protocols that support the process of moving a running VM (or application) between different physical machines without disconnecting the client or application. Memory, storage, and network connectivity of the VM are transferred from the original host machine to the destination.

VMware vMotion is one of the most popular live migration protocols, supporting the transfer of running VMs from one physical server to another with zero downtime, continuous service availability, and transactional integrity.

Recommendation: Live migration protocols, including vMotion, provide a control function to the VM infrastructure (and therefore could be argued to be control plane protocols). However, because of their traffic patterns (that is, large VM file transfers), these are generally not considered good candidates for the internetwork control (CoS 6) class because these could easily overwhelm the class. Therefore, live migration protocols would be better provisioned in a separate and dedicated class (if one exists, depending on the strategic QoS model in use) or can be combined with another traffic class, such as CoS 4, CoS 2, or even CoS 1.

Note vMotion traffic may be marked on compute hardware platforms—such as on Cisco Unified Compute Service (UCS) platforms—via Cisco UCS Manager QoS policies. However, the configuration of such policies is beyond the scope of this book, which is focused on *network* QoS design. Refer to the documentation to leverage this feature: http://www.cisco.com/en/US/docs/unified_computing/ucs/sw/gui/config/guide/2.1/b_UCSM_GUI_Configuration_Guide_2_1_chapter_010100.html.

- **Storage virtualization protocols:**

  - **Fibre Channel over Ethernet (FCoE)** is a computer network technology that encapsulates Fibre Channel (FC) frames over Ethernet networks. FCoE requires a lossless transport service and is a Layer 2 protocol that cannot natively be routed (unless tunneled over IP as in the case of Fibre Channel over IP [FCIP]). Recommendation: FCoE requires a lossless service via PFC and is generally recommended to be marked to CoS 3 (a default with NX-OS). Whenever possible, it is usually best to dedicate this class / virtual lane to FCoE.

  - **Internet Protocol Small Computer System Interface (iSCSI)** encapsulates SCSI commands within IP to enable data transfers. iSCSI can be used to transmit data over LANs, WANs, or the Internet and can enable location-independent data storage and retrieval. Recommendation: iSCSI does not require a lossless service because it usually uses TCP to achieve reliable transport. Therefore, it may be provisioned in a separate and dedicated class (if one exists, depending on the strategic QoS model in use) or can be combined with another traffic class, such as CoS 2 or CoS 1.

These protocols—several of which are Layer 2 protocols—need to be included within the DC marking model.

Furthermore, it should be recognized that not all enterprise applications may be present within the data center—or perhaps some of these may have a significantly reduced presence there. For example, voice and interactive video may to a large extent be absent from the data center, because most of these types of endpoints reside outside the DC (although some of these types of media flows may be present as they are directed to/from conferencing servers/gateways). Similarly, there may be significantly fewer scavenger applications within the data center because physical access and administration of this PIN is more strictly controlled by the enterprise IT department. Such application traffic patterns in the DC should likewise be reflected in the DC marking and provisioning models.

CoS/DSCP Marking

Recommendations:

- Realize that some (Layer 2) protocols within the data center require CoS-based classification.
- Recognize the limitations of a CoS-only marking model and consider a hybrid CoS-and-DSCP marking model (when supported).

While it was previously recommended to use a DSCP-based marking model (because this is end to end), this recommendation usually needs to be relaxed within the data center because some of the critical protocols therein are Layer 2 and therefore do not support DSCP-marking, only CoS.

Although it might be tempting to consider a CoS-only model, this may be insufficient, as you can see by examining the IEEE 802.1Q-2005 CoS-use recommendations shown in Table 22-2.

Table 22-2 *IEEE 802.1Q-2005 CoS-Use General Recommendations*

CoS	Acronym	Description
0	BE	Best Effort
1	BK	Background
2	EE	Excellent Effort

CoS	Acronym	Description
3	CA	Critical Applications
4	VI	Video (<100 ms latency)
5	VO	Voice (<10 ms latency)
6	IC	Internetwork Control
7	NC	Network Control

Taking a closer look at these recommendations reveals some significant limitations:

- CoS 7 is reserved for network control.
- CoS 6 is reserved for internetwork control.
- CoS 5 is recommended for voice (but as has been discussed, there may not be much voice within the DC).
- CoS 4 is recommended for video (but as has also been discussed, there may not be much *interactive video* within the DC). Broadcast video (CS5) and streaming video (AF3) may be present, although it should be pointed out that neither of these map to CoS 4 by default.
- CoS 3 is generally reserved and dedicated to FCoE.

This leaves an administrator with effectively only three classes to choose from: CoS 2 (Excellent Effort), CoS 1 (Background), or the default CoS 0 (Best Effort) class.

Therefore, to preserve classification and marking granularity, a hybrid CoS-and-DSCP-based marking model may be used within the data center (on platforms that support this), as will be shown.

CoS 3 Overlap Considerations and Tactical Options

Recommendations:

- Recognize the potential overlap of signaling (and multimedia streaming) markings with FCoE.
- Select a tactical option to address this potential overlap.

Perhaps the most controversial consideration relating to data center QoS design is how to reconcile and accommodate both signaling and FCoE within the data center.

Cisco has recommended signaling traffic markings that map (by default) to CoS 3 for nearly 20 years, and therefore a large customer base has engineered the QoS designs over their enterprise networks to align to this recommendation. Notwithstanding these designs, with the standardization of FCoE in 2009/10 the industry de facto marking recommendation for it has also landed on CoS 3.

Because of the unique and strict service level requirements of FCoE, some data center administrators prefer to leave CoS 3 dedicated to FCoE. This, however, raises the issue of what to do with signaling traffic within the data center—as it traverses to and from Cisco Unified Communications Managers (CUCMs).

In addition, this problem is not limited to signaling versus FCoE, but also extends to include multimedia streaming—which per RFC 4594 is recommended to be marked to AF3 and, therefore, also maps by default to CoS 3. Furthermore, as previously noted, multimedia streaming is a type of video traffic that is indeed present in most data centers networks.

Several tactical options exist to manage this potential marking overlap concern, including the following:

- Hardware isolation
- Layer 2 versus Layer 3 classification
- Asymmetrical CoS/DSCP markings
- DC/campus DSCP mutation
- Coexistence

Each of these tactical options is discussed in turn.

Hardware Isolation

Some platforms and interface modules do not support FCoE. For example, on the Cisco Nexus 7000, M-Series modules do not support FCoE, although F-series modules do. Therefore, an easy way to avoid CoS 3 overlap on such a platform is to utilize M-Series modules to connect to CUCMs, in addition to multimedia streaming servers, while using F-Series modules to connect to the DCB extended fabric supporting FCoE.

Layer 2 Versus Layer 3 Classification

Some Nexus platforms support either CoS-to-queue mapping or DSCP-to-queue mapping. For example, the Nexus 5500 can map CoS values to (QoS groups and then to) queues or DSCP values to (QoS groups and then to) queues. Therefore, signaling and multimedia streaming can be classified by DSCP values (CS3 and AF3, respectively) and be assigned to their respective queues—which would be separate and distinct from FCoE, which alone would be classified by CoS 3 and mapped to its own dedicated queue.

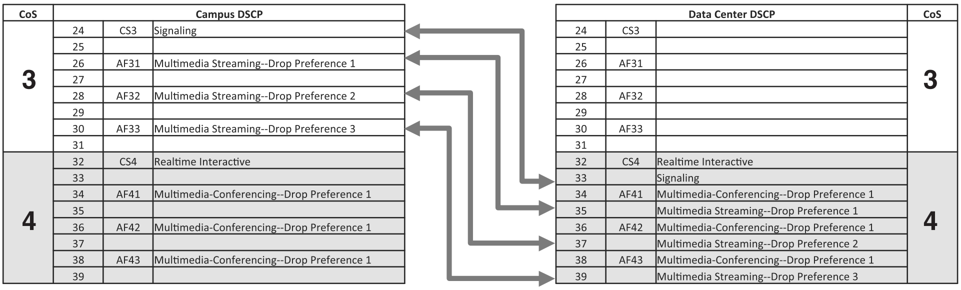
Note On some platforms, FCoE may be recognized by Ethertype or a CoS 3 marking.

Asymmetrical CoS/DSCP Marking

Another marking tactic that can be used to achieve signaling and FCoE separation is asymmetrical CoS/DSCP marking. In this context, *asymmetrical* would refer to the state where the three bits forming the CoS value do not match the first 3 bits of the DSCP value (which they would in a default/symmetrical state). For example, signaling could be marked CoS 4 (binary 100), but DSCP CS3 (binary 011000).

DC/Campus DSCP Mutation

Some network administrators may elect to sidestep the potential overlap entirely by performing ingress and egress DSCP mutation on data center-to-campus links. In this manner, they can reassign signaling and multimedia streaming DSCP markings on a one-to-one basis to unique, nonstandard DSCP values that map by default to CoS 4 (rather than CoS 3). Figure 22-14 illustrates an example of such bidirectional DSCP mutation.



CoS	Campus DSCP		
3	24	CS3	Signaling
	25		
	26	AF31	Multimedia Streaming--Drop Preference 1
	27		
	28	AF32	Multimedia Streaming--Drop Preference 2
	29		
	30	AF33	Multimedia Streaming--Drop Preference 3
	31		
4	32	CS4	Realtime Interactive
	33		
	34	AF41	Multimedia-Conferencing--Drop Preference 1
	35		
	36	AF42	Multimedia-Conferencing--Drop Preference 1
	37		
	38	AF43	Multimedia-Conferencing--Drop Preference 1
	39		

Data Center DSCP			CoS
24	CS3		3
25			
26	AF31		
27			
28	AF32		
29			
30	AF33		
31			
32	CS4	Realtime Interactive	4
33		Signaling	
34	AF41	Multimedia-Conferencing--Drop Preference 1	
35		Multimedia Streaming--Drop Preference 1	
36	AF42	Multimedia-Conferencing--Drop Preference 1	
37		Multimedia Streaming--Drop Preference 2	
38	AF43	Multimedia-Conferencing--Drop Preference 1	
39		Multimedia Streaming--Drop Preference 3	

Figure 22-14 *DC/Campus DSCP Mutation Example*

Coexistence

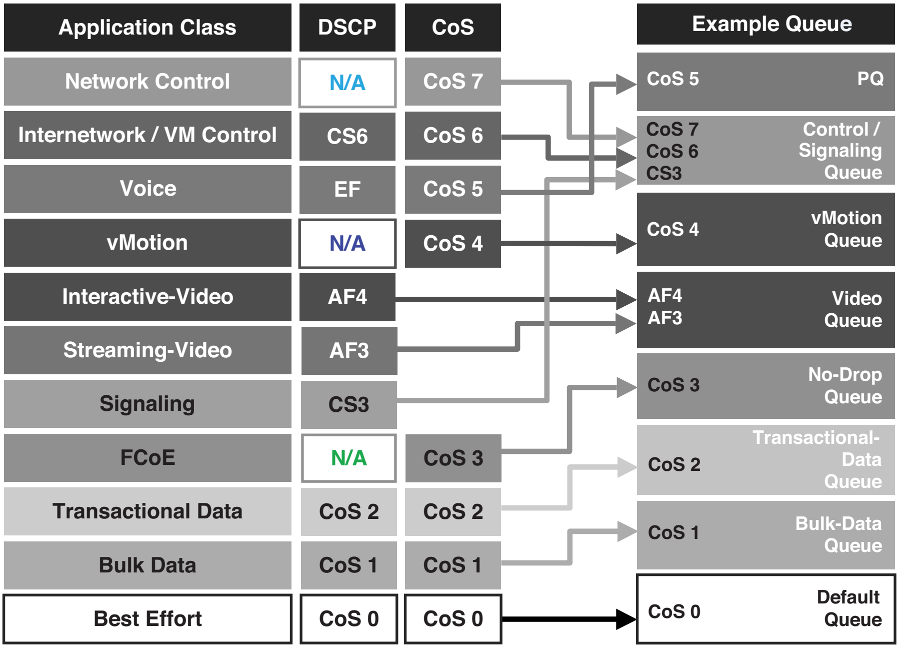
It bears mentioning that some administrators have also elected coexistence of FCoE and signaling—both with CoS 3 markings—over their data center networks. Reasoning that even if CUCM resides on a server with a converged network adaptor (CNA)—that is, an adapter that supports DCB and DCBX—then mixing these traffic types within the same virtual lane/queue has the overall result of extended lossless service to signaling in addition to FCoE.

Data Center Application-Based Marking Models

Recommendation:

- Recognize the limitations of a CoS-only marking model and consider a hybrid CoS-and-DSCP marking model (where supported).

As previously discussed, some Layer 2 data center protocols cannot be identified by DSCP, only CoS. In addition, it has been shown that in some instances (and where supported) it might prove beneficial to classify by both CoS and DSCP to achieve greater granularity. Figure 22-15 illustrates an example of a hybrid CoS-and-DSCP-based classification, marking, and queue-mapping model.



Application Class	DSCP	CoS
Network Control	N/A	CoS 7
Internetwork / VM Control	CS6	CoS 6
Voice	EF	CoS 5
vMotion	N/A	CoS 4
Interactive-Video	AF4	
Streaming-Video	AF3	
Signaling	CS3	
FCoE	N/A	CoS 3
Transactional Data	CoS 2	CoS 2
Bulk Data	CoS 1	CoS 1
Best Effort	CoS 0	CoS 0

Figure 22-15 *Data Center Hybrid CoS-and-DSCP-based Application-Based Data Center Marking and Mapping Example*

Note The queuing structure presented in Figure 22-15 does not match a specific Nexus queuing model, but is representative of logical concepts borrowed from several. These concepts include the option to map CoS-to-queue or DSCP-to-queue (from the Nexus 5500) in addition to the option of choosing which CoS values to map to a specific queue (from the Nexus 7000). Platform and module-specific queuing model recommendations are presented in detail in the respective design chapters to follow.

Data Center Application/Tenant-Based Marking Models

Recommendation:

- Leverage marking options to not only identify applications but also to reflect the customer tier in multitenant data center marking models.

A twist on application-based marking is to use marking values to not only identify applications but also to reflect user/group membership, such as customer tiers of service identification in a multitenant data center model, as shown in Table 22-3.

Table 22-3 *Data Center Application-/Tenant-Based Data Center Marking Model Example*

Traffic Type	Network Class	COS	Class, Property, BW Allocation
Infrastructure	Control	6	Platinum, 10 percent
	vMotion	4	Silver, 20 percent
Tenant	Gold, Transactional	5	Gold, 30 percent
	Silver, Transactional	2	Bronze, 15 percent
	Bronze, Transactional	1	Best Effort, 10 percent
Storage	FCOE	3	No Drop, 15 percent
	NFS datastore	5	Silver
Nonclassified	Data	0	Best Effort

Data Center QoS Models

Recommendation:

- Deploy the best models to match your data center network design QoS requirements.

Several data center ingress QoS models exist to choose from—one or more of which may be deployed within a single data center—including the following:

- **Trusted Server Model:** In most cases in the data center, you will be able to trust L2/L3 markings set on application servers, especially as a certain level of administrative control has already been exercised by allowing the servers to be physically installed within the data center. Nexus platforms trust CoS and DSCP by default. Therefore, no explicit configuration is needed on the interfaces/ports connecting to trusted application servers.
- **Untrusted Server Model:** In the case where you may not trust a server (or the team administering it, as has been known to happen), you may configure policies to re-mark CoS and DSCP to 0 on the port connected to it.
- **Single-Application Server Model:** The Single-Application Server Model is effectively the same as the Untrusted Server Model, with the exception that all traffic originating from it is marked to a nonzero value on the port connecting to it.
- **Multi-Application Server Model:** In the Multi-Application Server Model, access lists are used for classification and traffic is marked to multiple codepoints. This may be

because the application server does not mark traffic at all or perhaps it marks traffic to different codepoints as compared with your enterprises QoS model.

- **Server Policing Model:** In the Server Policing Model, one (or more) application classes are metered via one-rate or two-rate policers, with conforming, exceeding, and (optionally) violating traffic marked to different DSCP values.
- **Lossless Transport Model:** This **network-qos** model is enabled by default on some Nexus platforms/modules to provision a lossless service to FCoE. Note that this model may be combined with other QoS models.

After one (or more) ingress QoS models have been selected and configured, both ingress and egress queuing policies should also be configured. Finally (and optionally), control plane policing may also be configured.

Figure 22-16 illustrates these data center QoS models.

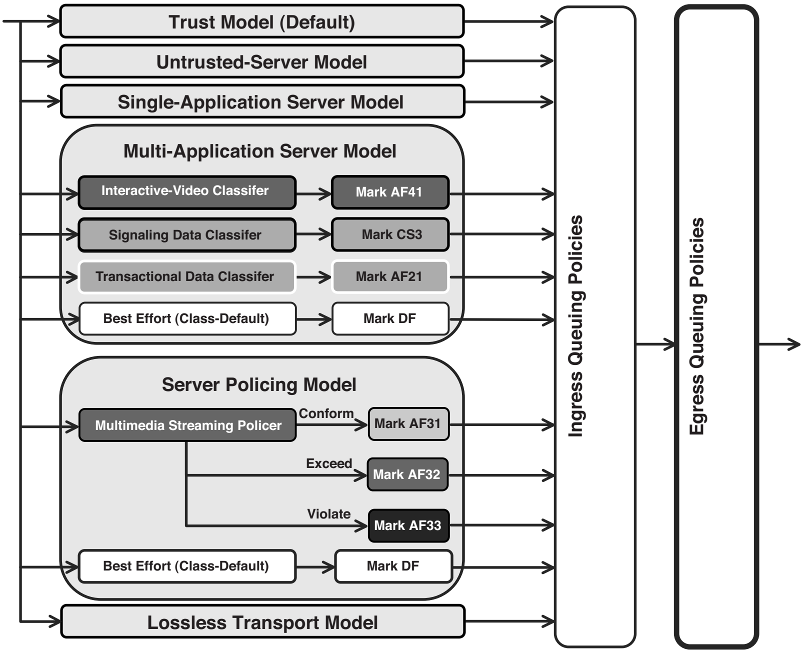


Figure 22-16 *Data Center QoS Models*

Data Center Port QoS Roles

Recommendation:

- Define consistent ingress and egress QoS policies for all data center Qos models, including the following:

- Trusted Server / Network Connection Model
- Untrusted Server Model
- Single-Application Server Model
- Multi-Application Server Model
- Server-Policing Model
- Lossless Transport Model

The QoS policy elements discussed thus far can be grouped into roles that various switch ports (or interfaces) serve within the campus architecture, including the following:

- **Trusted Server / Network Connection Model:**
  - This model is intended for ports connecting to trusted servers and/or ports/interfaces connecting to other trusted network devices.
  - No explicit ingress QoS policy is required—because both CoS and DSCP trust is enabled on ingress by default.
  - Ingress **type queuing** policies should be configured on these ports (or globally).
  - Egress **type queuing** policies should be configured on these ports (or globally).
- **Untrusted Server Model:**
  - This model is intended for ports connecting to untrusted servers.
  - Explicit ingress **type qos** re-marking policies are required to re-mark CoS and/or DSCP to zero.
  - Ingress **type queuing** policies should be configured on these ports (or globally).
  - Egress **type queuing** policies should be configured on these ports (or globally).
- **Single-Application Server Model:**
  - This model is intended for ports connecting to untrusted single-application servers.
  - Explicit ingress **type qos** re-marking policies are required to re-mark CoS and/or DSCP to a nonzero value.
  - Ingress **type queuing** policies should be configured on these ports (or globally).
  - Egress **type queuing** policies should be configured on these ports (or globally).

- **Multi-Application Server Model:**
  - This model is intended for ports connecting to untrusted multi-application servers.
  - Explicit ingress **type qos** access control list (ACL)-based classification and re-marking policies are required to re-mark CoS and/or DSCP to a nonzero value.
  - Ingress **type queuing** policies should be configured on these ports (or globally).
  - Egress **type queuing** policies should be configured on these ports (or globally).
- **Server Policing Model:**
  - This model is intended for ports connecting to servers that require metering to drop or to re-mark.
  - Explicit ingress **type qos** policing policies are required to drop or re-mark CoS and/or DSCP values.
  - Ingress **type queuing** policies should be configured on these ports (or globally).
  - Egress **type queuing** policies should be configured on these ports (or globally).
- **Lossless Transport Model:**
  - This model is intended for DCB ports requiring a lossless service.
  - Implicit **type network-qos** policies must be applied to **system qos** (which may be enabled by default on some platforms/modules) to provision a lossless service—typically for FCoE as identified by Ethertype and/or CoS 3; this model may be combined with other models.
  - Ingress **type queuing** policies should be configured on these ports (or globally).
  - Egress **type queuing** policies should be configured on these ports (or globally).

Figure 22-17 illustrates these data center port roles.

Note VM-facing interfaces on the Nexus 1000V do not support ingress/egress queuing. In addition, network-facing interfaces on the Nexus 1000V only support egress queuing.

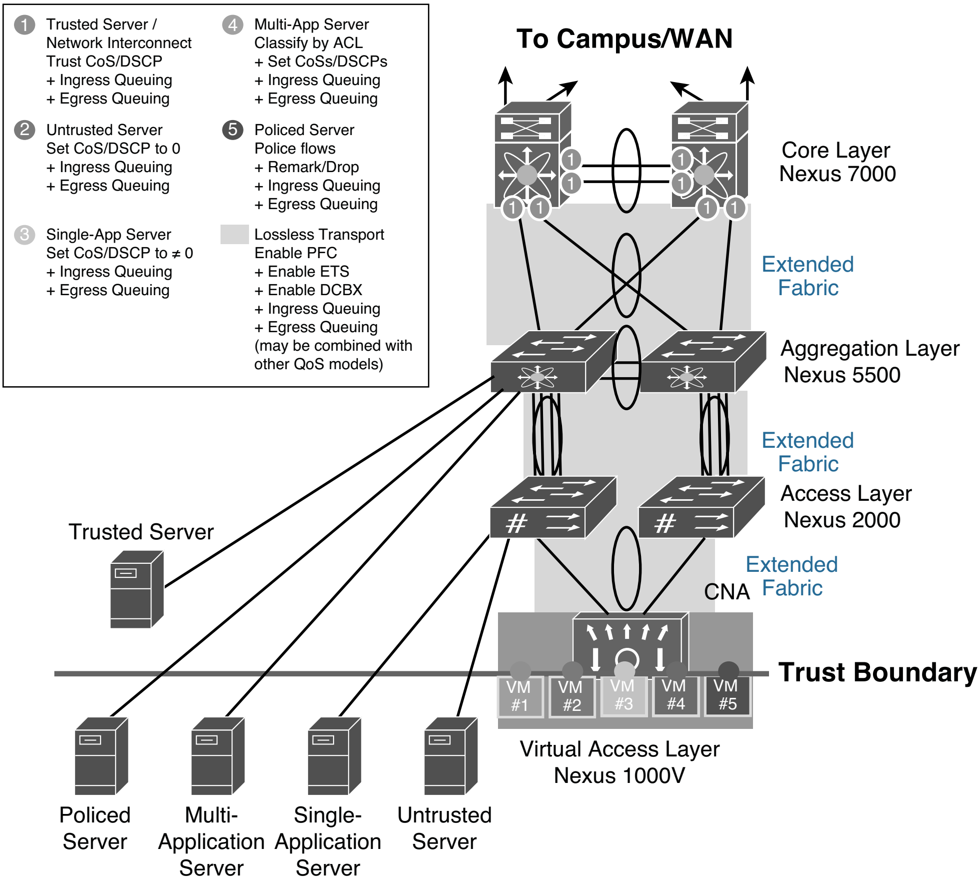

1 Trusted Server / Network Interconnect Trust CoS/DSCP + Ingress Queuing + Egress Queuing
2 Untrusted Server Set CoS/DSCP to 0 + Ingress Queuing + Egress Queuing
3 Single-App Server Set CoS/DSCP to ≠ 0 + Ingress Queuing + Egress Queuing
4 Multi-App Server Classify by ACL + Set CoSs/DSCPs + Ingress Queuing + Egress Queuing
5 Policed Server Police flows + Remark/Drop + Ingress Queuing + Egress Queuing
Lossless Transport Enable PFC + Enable ETS + Enable DCBX + Ingress Queuing + Egress Queuing (may be combined with other QoS models)
To Campus/WAN
Core Layer Nexus 7000
Extended Fabric
Aggregation Layer Nexus 5500
Extended Fabric
Access Layer Nexus 2000
Extended Fabric
CNA
Trust Boundary
VM #1
VM #2
VM #3
VM #4
VM #5
Virtual Access Layer Nexus 1000V
Trusted Server
Policed Server
Multi-Application Server
Single-Application Server
Untrusted Server

Figure 22-17 *Data Center QoS Roles*

Summary

This chapter discussed the considerations and recommendations for QoS designs for data center networks. The primary role of QoS in data center networks was identified as managing packet loss—even to the point of enabling DCB technologies to deliver a lossless transport (that is, a no-drop service). A secondary role of QoS in the data center that was discussed the classifying, marking, and (optionally) policing of application traffic.

Several strategic QoS design principles that apply in the data center were also reviewed, including the following:

- Always perform QoS in hardware rather than software when a choice exists.
- Classify and mark applications as close to their sources as technically and administratively feasible.

- Police unwanted traffic flows as close to their sources as possible.
- Enable queuing policies at every node where the potential for congestion exists.
- (Optional) Protect the control plane by enabling control plane policing.

Following this, the chapter covered several data center architectures, in addition to their respective QoS requirements, including the following:

- High-performance trading data centers, which have little or no QoS requirements.
- Big data data centers, which have QoS requirements similar to campus networks, but also benefit from DCTCP.
- Virtualized multiservice data centers, which have extended QoS requirements beyond regular Ethernet capabilities, including providing lossless transport. Therefore, these virtualized data centers require not only conventional QoS tools but also DCB tools to be applied throughout their designs.
- Secure multitenant data centers, which are a variation of VMDCs, but have different marking models to reflect not only application-based classification but also customer tiers of service for the tenants sharing the common infrastructure.
- Massively scalable data centers, which are (as the name implies) massively scaled virtualized data centers (thus sharing the same QoS/DCB requirements), and which particularly benefit from DCTCP due to their sheer amounts of traffic flow.

Following this, the data center bridging toolset was overviewed, including the following:

- Priority flow control (PFC), which provides a lossless service via a per-CoS PAUSE frame
- Enhanced transmission selection (ETS), which optimizes DCB bandwidth
- Data Center Bridging Exchange (DCBX), which discovers and shares DCB capabilities between peers

In addition, DCTCP was presented in brief, showing how leveraging IP ECN markings could amend standard TCP operation to significantly improve goodput.

Next, NX-OS QoS was summarized to highlight similarities and differences between it and MQC.

After this, various data center marking considerations were discussed, including tactical options for separating signaling from FCoE—both of which (by default) share the same CoS value of 3. Next, application-based and application/tenant-based marking models were presented for the data center. Finally, data center QoS models were summarized, along with port-specific data center roles.

Additional Reading

Cisco Design Zone for Data Centers: http://www.cisco.com/en/US/netsol/ns743/networking_solutions_program_home.html

Cisco Data Center Bridging and Virtualization: IEEE 802.1 Data Center Bridging Overview: https://www.cisco.com/en/US/netsol/ns783/index.html

Cisco Data Center Solutions: Priority Flow Control: Build Reliable Layer 2 Infrastructure White Paper: http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/white_paper_c11-542809_ns783_Networking_Solutions_White_Paper.html

IEEE DCB task group: http://www.ieee802.org/1/pages/dcbridges.html

Priority Flow Control (PFC—IEEE 802.1Qbb): http://www.ieee802.org/1/pages/802.1bb.html

Enhanced Transmission Selection (ETS—IEEE 802.1Qaz): http://www.ieee802.org/1/pages/802.1az.html

Congestion Notification (IEEE 802.1Qau): http://www.ieee802.org/1/pages/802.1au.html

Data Center Transmission Control Protocol (DCTCP): http://simula.stanford.edu/~alizade/Site/DCTCP_files/dctcp-final.pdf

DCTCP Overview: http://www.ietf.org/proceedings/80/slides/iccrg-3.pdf

Chapter 23

Data Center Virtual Access (Nexus 1000V) QoS Design

A new paradigm has emerged in the data center, changing the definition and role of the server access layer. In years past, servers were directly connected to physical access switches, thus giving a clear demarcation point between the server and the network. With the prevalence of virtual machines (VMs), such as VMware ESX/ESXi and Microsoft's Hyper-V, it is now possible to support numerous virtual servers on a single physical server.

This approach has helped consolidate the plethora of underutilized application-specific servers into far fewer physical servers that run multiple virtual operating systems on the same hardware. Today, the popularity this model has not only resulted in far less physical hardware being required in the data center, but it has led to reduced power consumption, cooling, rack space, and overall cost involved in supporting multiple servers. To make matters even better, thanks to centralized management tools like VMware's Virtual Center, virtualized servers have become much simpler to manage than separate standalone servers. However, the same is not necessarily true for the management of the networks that connect to the virtualized servers.

In the past, the job of the network engineer was simpler—the natural demarcation between server and network meant that he was able enforce all Layer 2 access policies (including QoS, security, and so on) on a device that was clearly in the domain of the network team. Nowadays, with a multiplicity of VMs running on a hypervisor (the host software that allows multiple VMs and operating systems to run on a single physical server), the access layer has moved away from the physical L2 switch toward a virtual switch, which resides as a software layer inside of the hypervisor.

Although the VM approach has many advantages and is likely to be a mainstay in the data center for many years to come, it has also introduced a new set of challenges, particularly for the network team. For example, if an organization has a well-defined set of quality of service (QoS) policies that have been implemented on the physical switching infrastructure in the data center, these policies are now less controlled because the vast majority of server access ports exist in the virtual switch managed by the hypervisor. Obviously, unless you can implement a consistent set of QoS policies from server to host, the usefulness of any QoS design will greatly suffer. For example, if a hypervisor's default virtual switch lacks the ability to implement a consistent set of QoS policies that are in-line with the overall QoS design, then it becomes difficult to support QoS across the entire data center.

In an effort to address this challenge, Cisco has introduced the Nexus 1000V virtual access layer switch, based on NX-OS. When installed in a VM environment, this switch replaces the default virtual switch normally used by the hypervisor and helps to promote a consistent set of QoS features across the data center.

The virtual access switch acts exactly like a physical L2 switch, except that it is running in software on the hypervisor. It does not run in a VM itself; instead, it exists at the hypervisor layer and interacts closely with the virtual ports connect to the VMs and the physical network interface card (pNIC) used by the server. Like any other L2 access switch in the data center, the main role of the virtual access switch is to establish a trust boundary between the network and the servers and to manage traffic according to class on the uplink.

Figure 23-1 illustrates the roles of QoS in the virtual access layer.

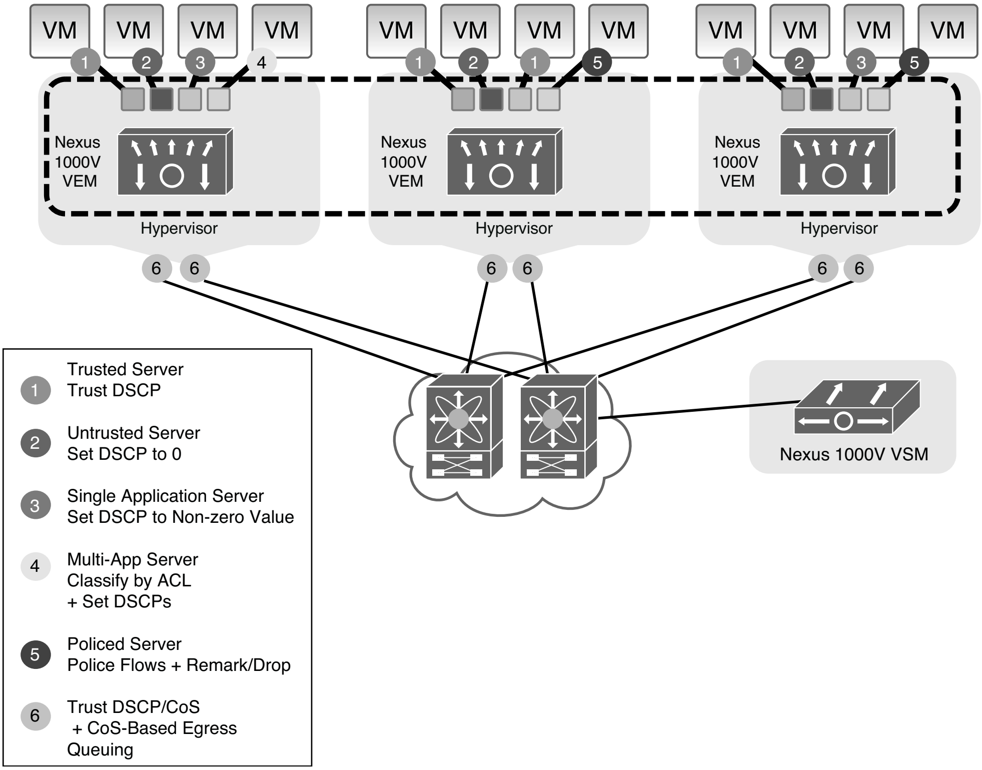

VM
VM
VM
VM
1
2
3
4
Nexus 1000V VEM
Hypervisor
6
6
VM
VM
VM
VM
1
2
1
5
Nexus 1000V VEM
Hypervisor
6
6
VM
VM
VM
VM
1
2
3
5
Nexus 1000V VEM
Hypervisor
6
6
Nexus 1000V VSM
1 Trusted Server Trust DSCP
2 Untrusted Server Set DSCP to 0
3 Single Application Server Set DSCP to Non-zero Value
4 Multi-App Server Classify by ACL + Set DSCPs
5 Policed Server Police Flows + Remark/Drop
6 Trust DSCP/CoS + CoS-Based Egress Queuing

Figure 23-1 *Data Center Virtual Access Layer Switch Port QoS Roles*

This chapter focuses on the QoS design of the virtual access layer, and in particular, the Nexus 1000V. The chapter begins with a review of the Nexus 1000V system architecture and then follows with a detailed examination of recommended QoS designs for this platform.

Cisco Nexus 1000 System Architecture

The Nexus 1000V consists of two main components: the Virtual Ethernet Module (VEM) and the Virtual Supervisor Module (VSM).

The VSM is the control plane of the Nexus 1000V. It provides all the configuration, management, and control functions for the VEMs. The VEMs are responsible for the actual data forwarding and never pass noncontrol traffic to the VSM, meaning that the VSM never has to deal with data plane traffic. In this way, the relationship between the VSM and the VEM is analogous to how the Nexus 5000 switch and Nexus 2000 Fabric Extender (FEX) interact; just like the FEX is a remote linecard controlled by the Nexus 5000, similarly the VEM can be considered as a remote linecard controlled by the VSM.

For example, when configuring multiple VEMs running on different physical servers, all configurations are done via the VSM, not the VEM. Whereas the VEMs reside at the hypervisor layer, the VSM is an application running on an actual server (either installed on a standalone server or as a VM). The VSM/VEM relationship goes beyond a simple remote linecard analogy. For example, even if the VSM fails or becomes unavailable for some reason, the VEMs are able to continue functioning normally.

Each VSM is capable of managing up to 64 remote VEMs, and therefore one single VSM can really be thought of as a switch with 64 remote linecards. If the VEMs reside on the same L2 domain, they function as a distributed switch capable of supporting features such as vMotion. vMotion is a technology where VMware Virtual Center (vCenter) is able to move an instantiation of a VM from one physical server to another on the same L2 segment. One of the elegant capabilities of the Nexus 1000V system architecture is that as a VM is moved between physical servers through vMotion, the QoS configuration of that particular VM seamlessly moves from one VEM to the next.

Note At the time of this writing, the Nexus 1000V supports both VMware ESX/ESXi and Microsoft's Hyper-V. However, the examples given in this chapter focus on using the Nexus 1000V in a VMware-only environment.

Within a single hypervisor, each VM connects to a virtual Ethernet (vEthernet) access port on the Nexus 1000V. From the perspective of the network administrator, these vEthernet interfaces are just like physical Ethernet interfaces on any other L2 access switch. On these interfaces, you can configure QoS features and establish a trust boundary to the VM.

Figure 23-2 illustrates the system architecture of the Nexus 1000V.

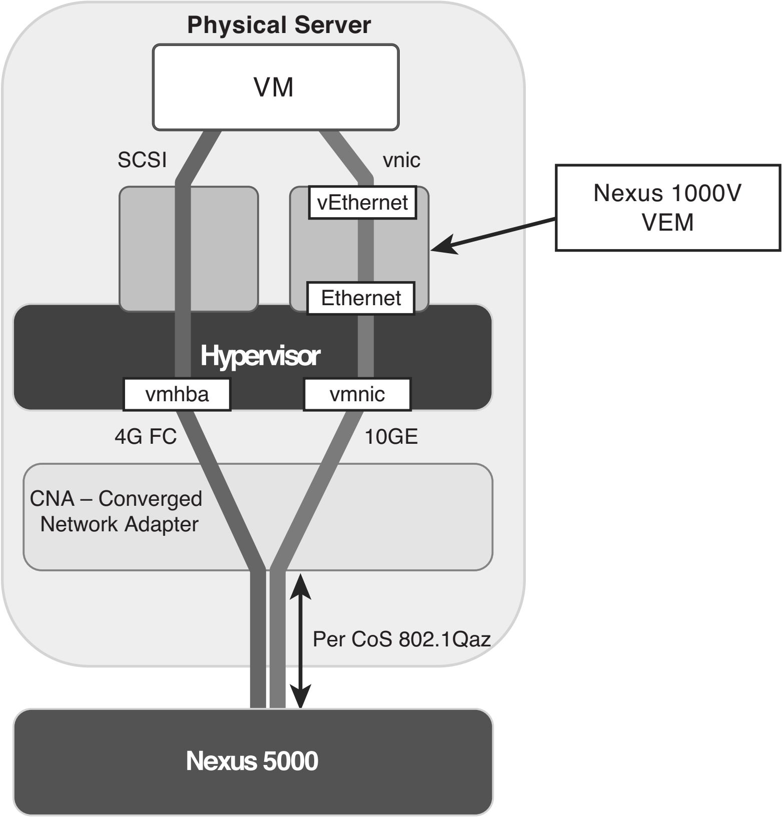


Figure 23-2 *Nexus 1000V System Architecture Overview*

As shown in Figure 23-2, the VEM sits between the hypervisor and the VMs. Although only one VM is shown attached to the VEM (connected via the vEthernet interface), it is possible to have up to 216 vEthernet interfaces, meaning that 216 VMs can be connected to a single VEM. The Ethernet interface shown at the bottom of the VEM is an Ethernet interface that the VMware administrator maps to the physical NIC on the server. The uplink in Figure 23-2 is shown as a single Ethernet interface connected to a converged network adapter (CNA). However, the uplink could also be a port channel of bundled 10GE links.

It is also important to note from Figure 23-2 that no Fiber Channel over Ethernet (FCoE) traffic is processed through the VEM. All SCSI data connections are connected through a separate virtual host bus adapter (HBA), which then passes the FCoE traffic to the CNA, which in turn allows FCoE transmission over the same 10GE links used for IP-based traffic. From a QoS perspective, it is important to understand this point because while the other NX-OS platforms, such as the Nexus 5000 and 7000 series, have the capability to classify and manage FCoE traffic, the Nexus 1000V has no need to process this traffic as it is managed by the CNA (which is configured through the server management utility).

As you can see from Figure 23-2, the CNA plays an important role in QoS. It is through enhanced transmission selection (ETS) that per-group traffic class allocation is accomplished. For example, if the CNA is physically connected to a switch capable of ETS, such as the Nexus 5500, the two devices communicate bandwidth requirements for each class. Furthermore, if the Nexus 5500 begins to encounter congestion on a no-drop class, it is capable of sending a PFC pause signal to the CNA instructing it to temporarily pause traffic just for that class, while all other Ethernet traffic continues to be switched normally. This is particularly effective for FCoE traffic because the SCSI payload has no facility to retransmit lost data. TCP/IP traffic, however, is more resilient to dropped packets because of the connection-oriented nature of TCP and therefore would benefit little from being configured as a no-drop class.

Note The upstream switch also needs to be capable of Data Center Bridging Exchange (DCBX) protocol. Examples include the Nexus 5500 or the Nexus 7000 F2/F2e modules.

Although the CNA's role in the end-to-end QoS design is important, especially if lossless traffic such as FCoE is in play, the main focus of this chapter is the Nexus 1000V, and in particular the QoS features applied to the VM and uplink interfaces shown in Figure 23-2.

Nexus 1000V Configuration Notes

Unlike other NX-OS switching platforms, on the Nexus 1000V you do not need to configure each vEthernet, Ethernet, or port channel interface directly. Instead, you need to configure port profiles. Port profiles are network configuration settings that describe the overall characteristics of an interface. These might be access control lists (ACLs) applied to the port, QoS service policies, VLAN assignments, NetFlow collection, and so on. After these port profiles have been configured and enabled, the VSM dynamically pushes them to vCenter, where they show up as port groups. The VMware administrator then assigns these port groups to either VMs or pNICs. At this point, the interfaces in the VEM inherit the capabilities defined in the port profiles, and these are pushed to the VEM by the VSM.

For example, suppose that you want to create a port profile to support Cisco's Unified Communication Manager (CUCM). Without going into specific details, the port profile you create has the following properties:

- **VLAN number:** Voice protocol control VLAN assignment
- **Security:** ACL to allow only SIP
- **QoS Profile:** QoS profile supporting CUCM
- **Management:** NetFlow enabled

Once the port profile is configured in the VSM, it automatically becomes visible in vCenter. All the VMware administrator needs to do now is to create the VM for CUCM and connect this port group to the CUCM VM for the vEthernet interface to inherit all these capabilities.

Therefore, you will notice that throughout this chapter the configuration examples always apply QoS policies to port profiles, not to interfaces.

One important distinction to be aware of between the Nexus 1000V and other NX-OS platforms is the types of class and policy maps that can be configured. In the other NX-OS platforms, there are three types of class and policy maps. These are the **qos**, **network-qos**, and **queuing** type maps. However, in the Nexus 1000V, only the **qos** and **queuing** type class maps and policy maps exist.

As with other NX-OS platforms, type **qos** and **queuing** QoS policies are used for the following purposes:

- **qos:** Defines Modular QoS command-line interface (MQC) objects used for marking and policing.
- **queuing:** Defines MQC objects used for queuing and scheduling. Unlike some other NX-OS platforms, the queuing type policy cannot be used for any marking.

At the time of this writing, unlike the other NX-OS platforms, the Nexus 1000V does not support control plane policing (CoPP) QoS configuration.

Monitoring QoS Statistics

When monitoring QoS performance on the Nexus 1000V, it is important to confirm that the QoS statistics feature has been enabled. This will allow you to see the packet and byte counters for each traffic class associated with the various QoS policies. To enable QoS statistics, use the global NX-OS command shown in Example 23-1.

Example 23-1 *Enabling QoS Statistics*

```
N1KV(config)# qos statistics
 ! globally enables qos statistics
```

Ingress QoS Model

Ingress QoS on the Nexus 1000V primarily focuses on the vEthernet interfaces—those connected to the VMs. Although re-marking and policing can in theory be applied on ingress to the uplink ports, this has little value because the uplink is part of the trusted network. The following section sets a model for the chapters to come in the data center part of this book. First, various server trust models are considered, and then classification and marking are examined for both the single-application and multi-application server models. The ingress policed server model is then examined.

Trust Models

In the data center, it is common to trust the QoS markings from the servers. In some cases, such as when third-party managed applications are being hosted, or in a cloud-based model where multiple customers share the data center infrastructure, it might be necessary to consider the servers as untrusted. The following sections examine both of these scenarios: the trusted and untrusted server models.

Trusted Server Model

In most cases, the class of service (CoS) and differentiated services code point (DSCP) QoS markings of the servers running in the VM environment are trusted. By default, both the CoS and DSCP markings of all packets are trusted as they enter both the uplink and vEthernet interfaces on the Nexus 1000V. Typically, the vEthernet interfaces are not trunked ports, so only DSCP markings are present.

Untrusted Server Model

In some cases, an application server may not trust the DSCP value that is marked by the server's application, and therefore may want to re-mark it to another value. Most commonly, the untrusted server's DSCP value is marked to zero.

Configuring the untrusted server policy is done in four steps:

1. Create an access control list (ACL) to match all traffic from the server.
2. Create a **qos** type class map that uses the ACL to match all traffic originating from the server.
3. Configure a **qos** type policy map to rewrite the DSCP to zero.
4. Attach the **qos** type policy map to the port profile for the given server, allowing the VMware administrator to map it to the appropriate VMs.

Example 23-2 shows how to configure a port profile that can be applied to untrusted server ports. Notice in Step 2 in this example the various options that are available for matching traffic. Although many options are available, only the **access-group** option can correctly identify traffic coming from a particular server.

Example 23-2 *Configuring a Port Profile for an Untrusted Server*

```
! Step 1 - Create the matching ACL
N1KV(config)# ip access-list UNTRUSTED
N1KV(config-acl)# permit ip any any
```

```
 ! Step 2 – Create the qos type class map
N1KV(config)# class-map UNTRUSTED-SERVER
N1KV(config-cmap-qos)# match ?
  access-group   Access group
  class-map      Class map
  cos            IEEE 802.1Q class of service
  discard-class  Discard class
  dscp           DSCP in IP(v4) and IPv6 packets
  ip             IP
  not            Negate this match result
  packet         Packet
  precedence     Precedence in IP(v4) and IPv6 packets
  qos-group      Qos-group
N1KV(config-cmap-qos)# match access-group name UNTRUSTED
 ! Create a class map that matches on the ACL

 ! Step 3 – Create the qos type policy map
N1KV(config)# policy-map UNTRUSTED-SERVER-POLICY
N1KV(config-pmap-qos)# class UNTRUSTED-SERVER
N1KV(config-pmap-c-qos)# set ?
  cos            IEEE 802.1Q class of service
  discard-class  Discard class
  dscp           DSCP in IP(v4) and IPv6 packets
  precedence     Precedence in IP(v4) and IPv6 packets
  qos-group      Qos-group

N1KV(config-pmap-c-qos)# set dscp 0
 ! Set the DSCP to 0

 ! Step 4 – Attach the service-policy to the correct port profile
N1KV(config)# port-profile type vethernet APPLICATION-X
N1KV(config-port-prof)# service-policy type qos input UNTRUSTED-SERVER-POLICY
 ! Attach the policy map to a vethernet port profile – in this case, "APPLICATION-X"
```

Notice from Example 23-2 that no specific **qos** type policy was specified in the class and policy maps. This is because the **qos** type is default. In this example, a port profile called **APPLICATION-X** was used. This is just a generic example of a port profile used to illustrate where the service policy is attached. The port profile that you use will be the same one that defines the other characteristic used for a set of VMs.

Note As you can see from Example 23-2, the Nexus 1000V has many options for classifying traffic in the **qos** type class map, such as discard class, packet length, IP precedence, and QoS group. Although these options for classification of traffic are available to you, they are not used in this chapter.

You can verify the configuration in Example 23-2 with the following commands:

- **show class-map type qos**
- **show policy-map type qos**
- **show port-profile name**
- **show policy-map interface vethernet [interface number]**

Example 23-3 shows how to verify the correct configuration of the class and policy map.

Example 23-3 *Verifying the qos Type QoS Policy*

```
N1KV# show class-map UNTRUSTED-SERVER

  Type qos class-maps
  ====================

    class-map type qos match-all UNTRUSTED-SERVER
      match access-group name UNTRUSTED

N1KV# show policy-map UNTRUSTED-SERVER-POLICY

  Type qos policy-maps
  ====================

  policy-map type qos UNTRUSTED-SERVER-POLICY
    class  UNTRUSTED-SERVER
      set dscp 0
```

Note that whereas the policy map is applied to the port profile, the policy map is actually verified on the interface that the port profile is applied to through vCenter. For example, if the VMware administrator applies the port profile created in Example 23-2 to interface vethernet 10, the verification is shown in Example 23-4.

Example 23-4 *Verifying the qos Type QoS Policy*

```
N1KV# show policy-map interface vethernet 10

Global statistics status :   enabled

Vethernet1

  Service-policy (qos) input:   UNTRUSTED-SERVER-POLICY
    policy statistics status:   enabled
    Class-map (qos):   UNTRUSTED-SERVER (match-all)
      114 packets
      Match: access-group UNTRUSTED
      set dscp 0
```

Classification and Marking

Although the Nexus 1000V is capable of classification and marking on egress, this action primarily happens only on ingress and only on the port profiles that are applied to vEthernet interfaces. Because all traffic coming from the upstream network is trusted, no re-marking needs to happen on the uplink ports.

The following sections examine how to classify and re-mark VM traffic for the following server models:

- Single-Server Application Model
- Multiserver Application

One important point to understand is that when re-marking ingress traffic on vEthernet interfaces, you need to set *both* the DSCP and the CoS values. Although DSCP is the primary classification tool throughout the IP network, the Nexus 1000V only supports CoS-to-queue mapping. This means that unless the CoS is correctly set on ingress from the VM, the egress queuing policy will not be able to correctly classify traffic and it will assign it to the default queue (CoS 0).

Single-Application Server Model

The Single-Server Application Model is very similar to the Untrusted Server Model shown in Examples 23-2 to 23-4. The only difference is that the DSCP is set to some nonzero value and the CoS is set to a corresponding value. In Example 23-2, it was not necessary to set the CoS value because all incoming traffic from the VM toward the wired network is untrusted and by default gets assigned a CoS value of 0. Example 23-5 demonstrates how to correctly re-mark the DSCP and CoS for a media server.

Example 23-5 *Configure the Single-Server Server QoS Policy*

```
N1KV(config)# class-map type qos SINGLE-SERVER
N1KV(config-cmap-qos)# match access-group name MEDIA-SERVER
 ! Match on the ACL defining the MEDIA-SERVER

N1KV(config)# policy-map type qos REMARK
N1KV(config-pmap-qos)# class SINGLE-SERVER
N1KV(config-pmap-c-qos)# set dscp af41
 ! This line marks the DSCP
```

```
N1KV(config-pmap-c-qos)# set cos 4
 ! This line marks the CoS - necessary for egress queuing used later
! Attach the service-policy to the correct port profile
N1KV(config)# port-profile type vethernet APPLICATION-Y
N1KV(config-port-prof)# service-policy type qos input REMARK
```

After the policy map has been configured, it needs to be attached to the correct port profile.

Once this is done, it is important to verify that both the DSCP and the CoS have been set correctly, which you can do with the following commands:

- **show class-map** [**type qos**]
- **show policy-map** [**type qos**] (see Example 23-6)
- **show port-profile name**
- **show policy-map interface**

Example 23-6 *Verifying the Policy Map*

```
N1KV# show policy-map type qos

  policy-map type qos REMARK
    class  SINGLE-SERVER
      set dscp af41
      set cos 4

! Verifies that *both* DSCP and CoS are set correctly
```

Multi-Application Server Model

Now take an example where a single VM has multiple applications running on it, or at least an application is being used that has multiple communication protocols that require different DSCP values. As before, ACLs are used for classification, and policy maps are used to mark the correct DSCP and corresponding CoS values.

Consider an example where the network team chooses to explicitly mark traffic from a multimedia application server such that media traffic is identified and marked as DSCP AF41, signaling traffic is identified and marked CS3, conferencing data flows (transactional data) are marked to AF21, and all remaining flows are marked to the default class (DSCP and CoS 0). Example 23-7 demonstrates this configuration. Note that in this example various ACLs are referenced, but they are not shown for simplicity.

Example 23-7 *Configuring the Multi-App Server Model*

```
! Step 1 - configure the class maps
N1KV(config)# class-map type qos MEDIA-FLOWS
N1KV(config-cmap-qos)# match access-group name MEDIA-FLOWS
 ! Media flows are identified via MEDIA-FLOWS ACL
N1KV(config-cmap-qos)# class-map type qos SIGNALING
N1KV(config-cmap-qos)# match access-group name SIGNALING
 ! Signaling traffic is identified via the SIGNALING ACL
N1KV(config-cmap-qos)# class-map type qos CONFERENCING-DATA
N1KV(config-cmap-qos)# match access-group name CONFERENCING-DATA
 ! Conferencing data traffic is identified via a CONFERENCING-DATA ACL

 ! Step 2 - configures the policy maps
N1KV(config)# policy-map type qos MULTI-APP-SERVER
N1KV(config-pmap-qos)# class type qos MEDIA-FLOWS
N1KV(config-pmap-c-qos)# set dscp af41
N1KV(config-pmap-c-qos)# set cos 4
 ! Media flows data flows are marked DSCP AF41 and CoS 4
N1KV(config-pmap-c-qos)# class type qos SIGNALING
N1KV(config-pmap-c-qos)# set dscp cs3
N1KV(config-pmap-c-qos)# set cos 3
 ! Signaling data flows are marked DSCP CS3 and CoS 3
N1KV(config-pmap-c-qos)# class type qos CONFERENCING-DATA
N1KV(config-pmap-c-qos)# set dscp af21
N1KV(config-pmap-c-qos)# set cos 2
 ! Conferencing data flows are marked DSCP AF21 and CoS 2
N1KV(config-pmap-c-qos)# class class-default
N1KV(config-pmap-c-qos)# set dscp 0
N1KV(config-pmap-c-qos)# set cos 0
 ! Configures the default class to set DSCP and CoS to 0

! Step 3 - Attach the policy map to the correct port profile
N1KV(config)# port-profile type vethernet MULTI-APP-SERVER
N1KV(config-port-prof)# service-policy type qos input MULTI-APP-SERVER
```

One interesting facet of this configuration is that signaling traffic is marked as CoS 3 on the uplink 802.1Q trunk. Of course, this is the normal CoS marking for call signaling traffic. However, it is also the same CoS value used to identify FCoE traffic. As shown previously in Figure 23-2, FCoE traffic is not directly processed by the Nexus 1000V; instead, it is set by the DCBX communication between the CNA and the upstream L2 switch, such as the Nexus 5000. Therefore, if FCoE traffic is being used, the effect is that call signaling and FCoE will both be classified and handled in the same way by the upstream switch.

For example, one key feature of DCBX is that FCoE is typically treated as a no-drop class, where a congested switch is able to signal back to the connected device to pause transmission until the congestion is freed up. Because signaling traffic is also being marked as CoS 3 in this case, it also ends up being treated as no drop. There isn't a significant downside to this because signaling traffic is usually very small and is TCP based, which means it is resilient to paused traffic flows.

You can verify the configuration in Example 23-7 with the following commands:

- **show class-map** [**type qos**]
- **show policy-map** [**type qos**]
- **show port-profile name**
- **show policy-map interface**

Server Policing Model

The Nexus 1000V supports a full array of policing features, including single-rate and dual-rate policers. Single-rate policers monitor the specified committed information rate (CIR) of traffic, and dual-rate policers monitor both the CIR and peak information rate (PIR) of traffic, as defined in RFC 2697, RFC 2698, and RFC 4115.

Three traffic policing conditions (colors) are supported by the Nexus 1000V: conform (green), exceed (yellow), or violate (red). When the data rate exceeds the CIR or PIR values, packets are either marked down or dropped. Table 23-1 illustrates the various policer exceed or violate actions.

Table 23-1 *Nexus 1000V Policer Actions for Exceed or Violate*

Condition	Color	Description	Policer Action (only one action is allowed per condition)
Conform	Green	The data rate is within the defined boundaries.	The policer either transmits these packets as is, or changes the value in the header (DSCP or CoS), and then transmits these packets.
Exceed	Yellow	The data rate exceeds the defined boundary.	The policer can drop or mark down these packets.
Violate	Red	The data rate violates the defined boundaries.	The policer can drop or mark down these packets.

Although it is possible to apply the policer to both the vEthernet interfaces and the uplink interfaces, it is most practical to apply the policer on ingress on the vEthernet interfaces. This allows the Nexus 1000V to monitor and control the bandwidth consumed by each application as it enters the virtual switch.

In Example 23-8, a multimedia-conferencing server's media traffic is metered by a two-rate three-color marking scheme where AF41 DSCP values are marked down. Whereas the conform marking can be set directly, exceed and violate re-marking values are set via the **cir-markdown-map** and **pir-markdown-map** table-maps (which by default are set to reflect RFC 2597 markdown rules).

Example 23-8 *Verifying the Policy Map*

```
! This section configures the policing policy map
N1KV(config)# policy-map type qos SERVER-POLICING
N1KV(config-pmap-qos)# class class-default
N1KV(config-pmap-c-qos)# police cir percent 50 pir percent 75 conform set-dscp-
transmit af41 exceed set dscp dscp table cir-markdown-map violate set dscp dscp
table pir-markdown-map
 ! Traffic is policed to a CIR of 50% and a PIR of 75%
 ! Conforming traffic is marked AF41
 ! Exceeding traffic is re-marked per the cir-markdown-map
 ! Violating traffic is re-marked per the pir-markdown-map

N1KV(config)# port-profile type vethernet MEDIA-SERVER
N1KV(config-port-prof)# service-policy input SERVER-POLICING
 ! Attach the policing service policy to the correct port profile
```

You can verify the configuration in Example 23-8 with the following commands:

- **show table-map** [**cir-markdown-map**] (as shown in Example 23-9)
- **show table-map** [**pir-markdown-map**] (as shown in Example 23-10)
- **show class-map** [**type qos**]
- **show policy-map** [**type qos**]
- **show port-profile name**
- **show policy-map interface**

As highlighted in Example 23-9, AF41 (DSCP 34) and AF42 (DSCP 36) are re-marked to AF42 (DSCP 36) if found to be *exceeding* the CIR.

Example 23-9 *Verifying CIR Markdown Values: show table-map cir-markdown-map*

```
N1KV# show table-map cir-markdown-map


  Table-map cir-markdown-map
    default copy
    from 10,12 to 12
    from 18,20 to 20
    from 26,28 to 28
    from 34,36 to 36
```

As highlighted in Example 23-10, AF41 (DSCP 34) and AF42 (DSCP 36) are re-marked to AF43 (DSCP 38) if found to be *violating* the PIR.

Example 23-10 *Verifying PIR Markdown Values: show table-map pir-markdown-map*

```
N1KV# show table-map pir-markdown-map

  Table-map pir-markdown-map
    default copy
    from 10,12 to 14
    from 18,20 to 22
    from 26,28 to 30
    from 34,36 to 38
```

Egress QoS Model

The Nexus 1000V supports class-based weighted fair queuing (CBWFQ) on uplink interfaces in the egress (outbound) direction only. On the vEthernet interfaces, packets are treated in a first in, first out (FIFO) manner, and CBWFQ is not supported. Unlike the other NX-OS switching platforms, the Nexus 1000V is a software-based switching platform, which means that there are no hardware limitations imposed by interface application-specific integrated circuits (ASICs) and physical memory buffer sizes. The Nexus 1000V supports up to 56 software-defined custom queues, plus eight system-defined queues.

Note As with other platforms considered in this book, the approach in the data center focuses on using a four-class and eight-class generic egress queuing model. Although the 12-class model is also discussed considered throughout this book, because of the queuing architecture of the other Nexus switching platforms only the 4-class and 8-class models are considered in this section.

A queuing policy consists of the following components:

- A **queuing** type class map used to classify traffic
- A **queuing** type policy map used define queuing behavior
- The service policy, which is used to apply the policy to a port profile

As described in Table 23-2, the Nexus 1000V supports eight predefined protocols used by the **queuing** class map that help to identify various types of VMware/vMotion and VSM/VEM control traffic.

Table 23-2 *The Eight Predefined Classes on the Nexus 1000V*

Protocol Classes	Description
Nexus 1000V Control	All control traffic that carries sync and programming information for the VEM
Nexus 1000V Packet	Packets that carry any communication traffic that require processing by the VSM
Nexus 1000V Management	Management plane traffic for Nexus 1000V sourced from and destined to the mgmt0 interface (including VMware vCenter communications)
VMware vMotion	All vMotion traffic needed for live migration of VMs between hosts
VMware Fault-Tolerance Logging	Nondeterminism and synchronization data needed to achieve fault tolerance for VMs
VMware Management	Management traffic associated with the service console and the management-enabled vmk interface
VMware NFS Storage	NFS input/output packets
VMware iSCSI Storage	iSCSI input/output packets

Example 23-11 illustrates how to select any one of these predefined protocol classes in a queuing policy.

Example 23-11 *Using a Predefined Protocol in the Queuing Type Class Map*

```
N1KV(config)# class-map type queuing match-any EGRESS-QUEUE
N1KV(config-cmap-que)# match protocol ?
  n1k_control  N1K control traffic
  n1k_mgmt     N1K management traffic
  n1k_packet   N1K inband traffic
  vmw_ft       VMware fault tolerance traffic
  vmw_iscsi    VMware iSCSI traffic
  vmw_mgmt     VMware management traffic
  vmw_nfs      VMware NFS traffic
  vmw_vmotion  VMware vmotion traffic
```

Besides these eight predefined classes, you can also classify traffic by CoS value. However, unlike other NX-OS switching platforms, it is only possible to classify by either protocol (as shown in Example 23-11) or by CoS. This is why it is critical that the **qos** policy applied to the vEthernet interface correctly set *both* the DSCP and CoS value.

Note that the list of protocols shown in Example 23-11 is only available to the **queuing** type class map, not the **qos** type class map. This means that it is not possible to adjust the CoS markings of these protocols. That said, the system automatically marks all control traffic as CoS 6.

The following sections examine how to configure the Nexus 1000V egress queuing policy for both the four-class and eight-class generic models.

Four-Class Egress Queuing Model

The generic four-class reference model that is used in this book consists of the following classes:

- Real-Time
- Control
- Transactional Data
- Best Effort

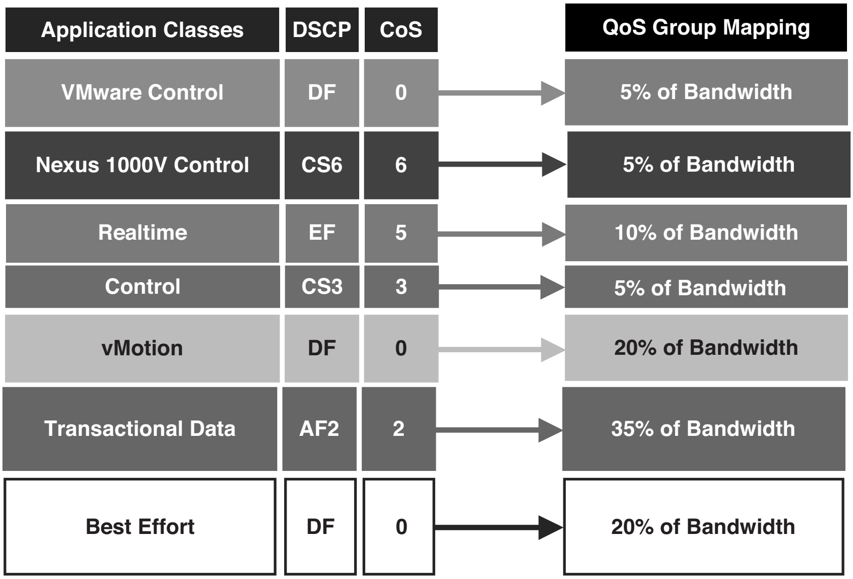
In the data center, and in particular when using the Nexus 1000V and VMware, there are some other traffic types that need special handling. As a minimum, there are the classes that are used for Nexus 1000V control, VMware control, and vMotion. In total therefore, to support the generic four-class model, a total of seven egress queues need to be configured on the Nexus 1000V. Figure 23-3 illustrates these seven classes along with their associated CoS and DSCP markings.

Although you might expect to just see four classes in the QoS design description shown in Figure 23-3, these three extra control classes that are required in the virtual data center push the actual QoS model to a seven-class model.

The Nexus 1000V Control class shown in Figure 23-3 is the sum of three protocols mapped into the one class, as follows:

- Nexus 1000V Control
- Nexus 1000V Management
- Nexus 1000V Packet

Note that the Nexus 1000V automatically marks these control packets as CS6/CoS 6.



Application Classes	DSCP	CoS	QoS Group Mapping
VMware Control	DF	0	5% of Bandwidth
Nexus 1000V Control	CS6	6	5% of Bandwidth
Realtime	EF	5	10% of Bandwidth
Control	CS3	3	5% of Bandwidth
vMotion	DF	0	20% of Bandwidth
Transactional Data	AF2	2	35% of Bandwidth
Best Effort	DF	0	20% of Bandwidth

Figure 23-3 *Adapting the Four-Class Egress Queuing Model the Nexus 1000V*

The two VMware-specific classes—vMotion and VMware control—are both unmarked by the system. Although the Nexus 1000V can classify and queue these protocols on egress, it is not able to classify and mark these protocols. The result is that they are sent to the upstream switch marked as CoS 0. Because the default behavior is to assign these protocols to the default class, particular care must be taken, especially with vMotion. Because vMotion is considered a mission-critical application, and because it can also be a massive bandwidth hog within the data center, treating it simply as best effort is hardly acceptable. The best way to handle this situation is to re-mark these protocols to a designated QoS class on the next upstream switch, such as the Nexus 5000.

The bandwidth percentages shown in Figure 23-3 are not meant to be an absolute recommendation, but are rather illustrated for example purposes only. The actual traffic flows in your data center will guide the best allocation of bandwidth to use. For example, you may not have any real-time traffic whatsoever in your data center, so this class can be eliminated and the bandwidth reassigned to other classes.

As with other control traffic, VMware control and the Nexus 1000V control classes do not need a significant amount of bandwidth. However, vMotion can consume a huge amount of bandwidth and uses large packet sizes, which can aggressively fill up egress queues. Therefore, depending on the amount of vMotion traffic expected in your data center, the bandwidth assignment should be adjusted accordingly.

It is important to remember that the Nexus 1000V sits at the hypervisor layer—situated between the VMs and a pNIC. The bandwidth percentages used in the configuration in effect determine how the packets are scheduled from the Nexus 1000V to the output interface queue on the pNIC. The packets are actually transmitted out into the physical network from the pNIC output interface queue.

Example 23-12 demonstrates how to configure the Nexus 1000V to support the adapted four-class QoS model.

Example 23-12 *Adapting the Four-Class QoS Model to the Nexus 1000V*

```
! The first step is to create the class maps
N1KV(config)# class-map type queuing match-any N1KV-CONTROL
N1KV(config-cmap-que)# match protocol n1k_control
N1KV(config-cmap-que)# match protocol n1k_packet
N1KV(config-cmap-que)# match protocol n1k_mgmt
 ! This combines the three N1K control protocols into a single class
N1KV(config-cmap-que)# class-map type queuing match-all VMOTION
N1KV(config-cmap-que)# match protocol vmw_vmotion
 ! Configures the VMware vMotion class
N1KV(config-cmap-que)# class-map type queuing match-all VMW-CONTROL
N1KV(config-cmap-que)# match protocol vmw_mgmt
 ! Configures the VMware management class
N1KV(config-cmap-que)# class-map type queuing match-all REALTIME
N1KV(config-cmap-que)# match cos 5
 ! Creates the Real-Time class
N1KV(config-cmap-que)# class-map type queuing match-all CONTROL
N1KV(config-cmap-que)# match cos 3
 ! Creates the Control class
N1KV(config-cmap-que)# class-map type queuing match-all TRANSACTIONAL-DATA
N1KV(config-cmap-que)# match cos 2
 ! Creates the Transactional Data class

 ! The second step is to define the queuing policy maps
N1KV(config)# policy-map type queuing 4-CLASS-QUEUING-POLICY
N1KV(config-pmap-que)# class type queuing N1KV-CONTROL
N1KV(config-pmap-c-que)# bandwidth percent 5
 ! Assigns 5% to the Nexus 1K control class
N1KV(config-pmap-c-que)# class type queuing VMOTION
N1KV(config-pmap-c-que)# bandwidth percent 20
 ! Assigns 20% to the vMotion class
N1KV(config-pmap-c-que)# class type queuing VMW-CONTROL
N1KV(config-pmap-c-que)# bandwidth percent 5
! Assigns 5% to the VMware Management class
N1KV(config-pmap-c-que)# class type queuing REALTIME
N1KV(config-pmap-c-que)# bandwidth percent 10
 ! Assigns 10% to the Real-Time class
N1KV(config-pmap-c-que)# class type queuing CONTROL
```

```
N1KV(config-pmap-c-que)# bandwidth percent 5
 ! Assigns 5% to Control
N1KV(config-pmap-c-que)# class type queuing TRANSACTIONAL-DATA
N1KV(config-pmap-c-que)# bandwidth percent 35
 ! Assigns BW to the Transactional Data class, leaving 20% for default

 ! The final step is to attach the policy to the uplink port profile
N1KV(config)# port-profile type ethernet UPLINK
N1KV(config-port-prof)# service-policy output 4-CLASS-QUEUING-POLICY
```

As usual, you can verify the configuration in Example 23-12 with the following commands:

- **show class-map** [**type queuing**]
- **show policy-map** [**type queuing**]
- **show port-profile name**
- **show policy-map interface**

With the Nexus 1000V, it is also possible to verify how the queues are behaving on each individual VEM. The first step is to verify that the configured queues have been correctly deployed to the VEM, which you can do with the following command:

module vem [**module-number**] **execute vemcmd show qos node**

The output of this command lists all the defined traffic classes, each with a unique identifier (called NodeID). Example 23-13 illustrates the output of this command.

Example 23-13 *Displaying the Configured Queues on the VEM*

```
N1KV# module vem 4 execute vemcmd show qos node
nodeid   type     details
-------- -------- --------
       0    class   op_AND
                       ACL
       1    class    op_OR
                  protocol
                 n1k_cntrl
                  protocol
                   n1k pkt
                  protocol
                  n1k mgmt
       2    class   op_AND
                  protocol
```

```
               vmw vmotion
       3    class   op_AND
                  protocol
                  vmw mgmt
       4    class   op_AND
                   queuing
       5    class   op_AND
                   queuing
       6    class   op_AND
                   queuing
```

It is also possible to check that traffic classes are being matched outbound on the module uplink with the following command:

```
module vem [module-number] execute vemcmd show qos pinst [uplink LTL]
```

Note The LTL is a local identifier that pertains to each interface. Example 23-14 examines uplink interface Ethernet 1 on VEM 4 (interface Ethernet 4/1). The LTL can be determined for each interface by running the following command beforehand:

```
module vem [module-number] execute vemcmd show port
```

The output shown in Example 23-14 lists all the defined traffic classes that are queued outbound under the Egress_q class column. This class number belongs to the NodeID for that class, which can be correlated to the output shown in Example 23-13. For instance, notice that Egress_q class 2 pertains to the vMotion protocol.

Example 23-14 *Showing the Packet and Byte Counters for the Queues on the VEM*

```
N1KV# module vem 4 execute vemcmd show qos pinst 17

id       type
-------- --------
      17 Egress_q
            class        bytes matched         pkts matched
         -------- -------------------- --------------------
                1               342342                  684


                2            144903523                85803


                3              2223112                 3028


                4             43223340                13543
```

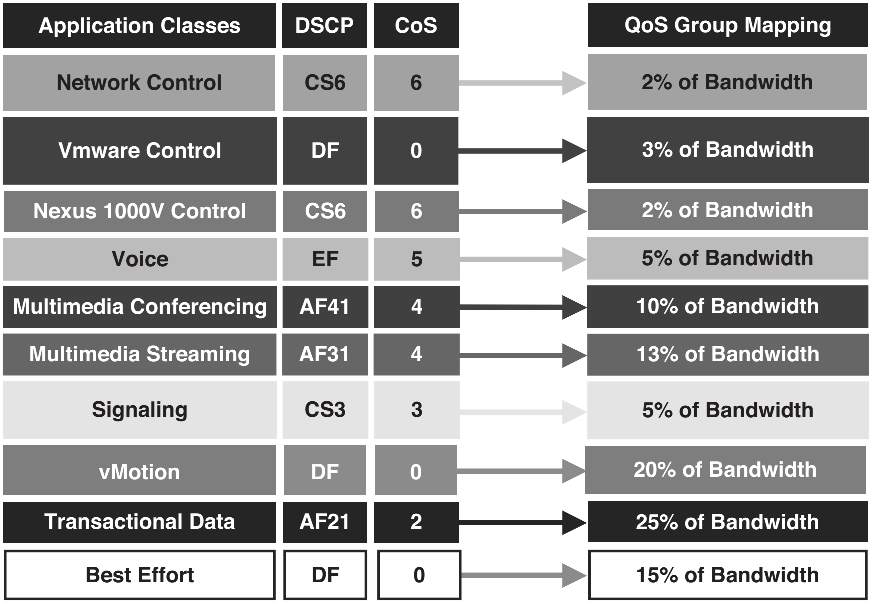
```
        5           223234            463

        6         23423223           8545
```

Eight-Class Egress Queuing Model

The eight-class QoS Model expands on the four-class model by adding Multimedia Conferencing and Multimedia Streaming, and splits Control into two new classes: Signaling and Network Control. Because the data center is a controlled environment, the Scavenger class is dropped from the typical eight-class model used in other places in the network. Therefore, in the data center, the eight-class model really only needs to support seven classes.

As with the four-class model, three system classes are added to support Nexus 1000V control, VMware control, and vMotion. Therefore, adding all the classes up and including the default class, there are now ten classes in use for this model (seven relevant classes from the eight-class model and three control classes).

Figure 23-4 illustrates the breakdown of these ten classes along with their associated DSCP and Cos markings.



Application Classes	DSCP	CoS	QoS Group Mapping
Network Control	CS6	6	2% of Bandwidth
Vmware Control	DF	0	3% of Bandwidth
Nexus 1000V Control	CS6	6	2% of Bandwidth
Voice	EF	5	5% of Bandwidth
Multimedia Conferencing	AF41	4	10% of Bandwidth
Multimedia Streaming	AF31	4	13% of Bandwidth
Signaling	CS3	3	5% of Bandwidth
vMotion	DF	0	20% of Bandwidth
Transactional Data	AF21	2	25% of Bandwidth
Best Effort	DF	0	15% of Bandwidth

Figure 23-4 *Adapting the Eight-Class Egress Queuing Model the Nexus 1000V*

One anomaly here is that even though multimedia streaming traffic is marked with DSCP AF31, it is recommended to be marked with CoS 4. (AF31 maps by default to CoS 3 based on the three MSB.) If the CoS value of multimedia streaming is set to 3 it would end up being treated in the same class as FCoE traffic. However because multimedia streaming traffic is likely to consume a significant amount of bandwidth, it is best to ensure that it gets treated separately from FCoE on the upstream switch.

You may not have any real-time voice or video traffic present in your data center, so these classes can be reassigned as needed. Example 23-15 shows the configuration steps required to enable this QoS model.

Example 23-15 *Adapting the Generic Eight-Class Model to the Nexus 1000V*

```
! The first step is to create the class maps
N1KV(config)# class-map type queuing match-any N1KV-CONTROL
N1KV(config-cmap-que)# match protocol n1k_control
N1KV(config-cmap-que)# match protocol n1k_packet
N1KV(config-cmap-que)# match protocol n1k_mgmt
 ! Combine the three N1K control protocols into a single class
N1KV(config-cmap-que)# class-map type queuing match-all VMOTION
N1KV(config-cmap-que)# match protocol vmw_vmotion
 ! Configures the VMware vMotion class
N1KV(config-cmap-que)# class-map type queuing match-all VMW-CONTROL
N1KV(config-cmap-que)# match protocol vmw_mgmt
 ! Configures the VMware management class
N1KV(config-cmap-que)# class-map type queuing match-all NETWORK-CONTROL
N1KV(config-cmap-que)# match cos 6
 ! Creates the Control class
N1KV(config-cmap-que)# class-map type queuing match-all VOICE
N1KV(config-cmap-que)# match cos 5
 ! Creates the Real-Time class
N1KV(config-cmap-que)# class-map type queuing match-all MULTIMEDIA-CONFERENCING
N1KV(config-cmap-que)# match cos 4
 ! Creates the Multimedia Conferencing class
N1KV(config-cmap-que)# class-map type queuing match-all MULTIMEDIA-STREAMING
N1KV(config-cmap-que)# match cos 4
 ! Creates the Multimedia Streaming class
N1KV(config-cmap-que)# class-map type queuing match-all SIGNALING
N1KV(config-cmap-que)# match cos 3
 ! Creates the Signaling class
N1KV(config-cmap-que)# class-map type queuing match-all TRANSACTIONAL-DATA
N1KV(config-cmap-que)# match cos 2
 ! Creates the Transactional Data class

 ! The second step is to define the queuing policy maps
```

```
N1KV(config)# policy-map type queuing 8-CLASS-QUEUING-POLICY
N1KV(config-pmap-que)# class type queuing N1KV-CONTROL
N1KV(config-pmap-c-que)# bandwidth percent 2
 ! Assigns 2% to the Nexus 1K control class
N1KV(config-pmap-c-que)# class type queuing VMOTION
N1KV(config-pmap-c-que)# bandwidth percent 20
 ! Assigns 20% to the vMotion class
N1KV(config-pmap-c-que)# class type queuing VMW-CONTROL
N1KV(config-pmap-c-que)# bandwidth percent 3
! Assigns 3% to the VMware Management class
N1KV(config-pmap-c-que)# class type queuing NETWORK-CONTROL
N1KV(config-pmap-c-que)# bandwidth percent 2
 ! Assigns 2% to the Network Control class
N1KV(config-pmap-c-que)# class type queuing VOICE
N1KV(config-pmap-c-que)# bandwidth percent 5
 ! Assigns 5% to the Voice class
N1KV(config-pmap-c-que)# class type queuing MULTIMEDIA-CONFERENCING
N1KV(config-pmap-c-que)# bandwidth percent 10
 ! Assigns 10% to Multimedia Conferencing
N1KV(config-pmap-c-que)# class type queuing MULTIMEDIA-STREAMING
N1KV(config-pmap-c-que)# bandwidth percent 13
 ! Assigns 13% to Multimedia Streaming
N1KV(config-pmap-c-que)# class type queuing SIGNALING
N1KV(config-pmap-c-que)# bandwidth percent 5
 ! Assigns 5% to Signaling
N1KV(config-pmap-c-que)# class type queuing TRANSACTIONAL-DATA
N1KV(config-pmap-c-que)# bandwidth percent 25
 ! Assigns 25% of BW to the Transactional Data class

 ! The final step is to attach the policy to the uplink port profile
N1KV(config)# port-profile type ethernet UPLINK
N1KV(config-port-prof)# service-policy output 8-CLASS-QUEUING-POLICY
```

You can verify the configuration in Example 23-15 with the following commands:

- show class-map [type queuing]
- show policy-map [type queuing]
- show port-profile name
- show policy-map interface
- module vem [module-number] execute vemcmd show qos node
- module vem [module-number] execute vemcmd show qos pinst [uplink LTL]

Summary

This chapter focused on the QoS design of the virtual access layer using the Nexus 1000V switching platform. The chapter began by examining the need for a feature-rich switching platform that sits at the hypervisor layer. In particular, the Nexus 1000V provides a coherent set of network features, and in particular delivers the tools necessary to support a consistent QoS design for the data center. Although several VM hypervisors are available on the market, this chapter focused attention on VMware environments.

The Nexus 1000V system architecture was examined. The interaction of the VSM control module and the VEM switching module was discussed in detail. This also included a discussion of how the Nexus 1000V interacts with the VMs, the hypervisor, and the physical uplink.

Next, some of the Nexus 1000V configuration highlights were discussed, focusing particular attention on some subtle differences between this platform and the other NX-OS switches. Also, because of the role the Nexus 1000V plays in the virtual access layer, several feature differences were examined, particularly around the fact that this platform is software based instead of being reliant on hardware ASICs for switching and queuing.

The ingress QoS models were examined, including the various trust models. Because servers located in the data center are generally trusted, it was shown how to trust incoming DSCP markings and map these to the appropriate CoS values which are used for egress queuing. Also, classification and re-marking designs were examined in detail. Included in this section was a discussion on how to use the two-rate three-color policer.

Lastly, the egress queuing design was examined. It was shown that the Nexus 1000V has eight built-in queuing classes used for various control and management functions, in addition to for vMotion. The four- and eight-class egress QoS models were examined in detail, including how to incorporate Nexus 1000V and VMware control classes, in addition to vMotion traffic, into these queuing models.

Additional Reading

Nexus 1000V QoS home page: http://www.cisco.com/go/nexus1000

Comparing the Nexus 1000V with the VMware Virtual Switch: http://www.vmware.com/files/pdf/technology/cisco_vmware_virtualizing_the_datacenter.pdf

Nexus 1000V Deployment Guide: http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9902/guide_c07-556626.html

Nexus 1000V QoS Configuration Guide: http://www.cisco.com/en/US/products/ps9902/products_installation_and_configuration_guides_list.html

This page intentionally left blank

Chapter 24

Data Center Access/ Aggregation (Nexus 5500/2000) QoS Design

In today's world of virtualized data centers, the edge of the network is becoming increasingly blurred. For example, virtual server systems like VMware and Hyper-V allow the network edge—and therefore the quality of service (QoS) trust boundary—to be enforced in the virtual machine (VM) hypervisor itself. With the trust boundary being extended toward the hypervisor, traditional data center access/aggregation switches are required to play multiple roles.

For example, when used in the role of an access layer switch, the switch must establish a trust boundary for the data center, in addition to potentially re-marking untrusted traffic. Adding the function of Layer 2 aggregation to the picture, the switch must also carefully manage packet loss through queuing mechanisms, while trusting differentiated services code point (DSCP) and class of service (CoS) markings.

The Nexus 5500 family of switches is featured in this chapter in the dual role of a data center access and aggregation switch, and the Nexus 2000 Fabric Extender (FEX) is featured in the role of a server access switch only.

Figure 24-1 illustrates the port-specific QoS roles of the Nexus 5500 and Nexus 2000 switches. As you can see here, within the data center the Layer 2 access switch usually trusts traffic from outside of the network, in addition to server-originated traffic. A trust boundary can be established at this layer and DSCP markings may take place, but in most cases it is expected that servers correctly mark their own application traffic so that the switches simply trust and classify traffic accordingly.

As depicted in Figure 24-1, each of the Nexus data center access and aggregation platforms classify and queue traffic according to their respective priorities.

Note Although the Nexus 2000 FEXs are also supported on the Nexus 7000, they are considered in detail in this chapter rather than in Chapter 25, "Data Center Core (Nexus 7000) QoS Design."

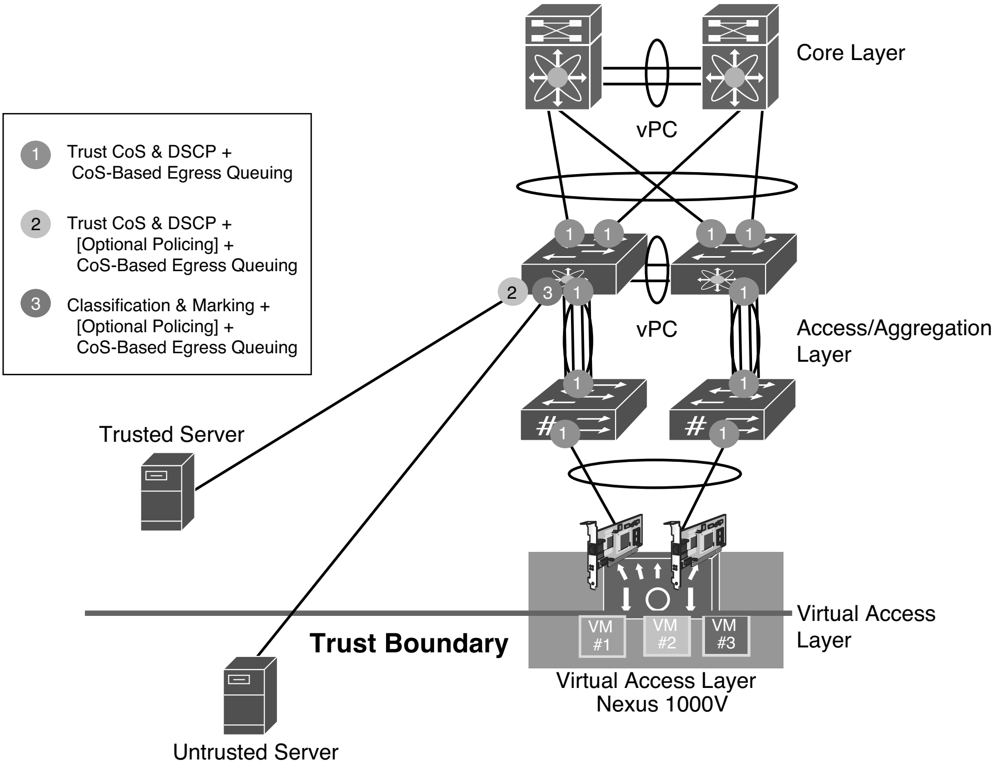

Core Layer
vPC
1 Trust CoS & DSCP + CoS-Based Egress Queuing
2 Trust CoS & DSCP + [Optional Policing] + CoS-Based Egress Queuing
3 Classification & Marking + [Optional Policing] + CoS-Based Egress Queuing
vPC
Access/Aggregation Layer
Trusted Server
Virtual Access Layer
Trust Boundary
VM #1
VM #2
VM #3
Virtual Access Layer Nexus 1000V
Untrusted Server

Figure 24-1 *Data Center Access/Aggregation Port QoS Roles*

Because the Nexus 5500 has a flexible role in the data center, its QoS capabilities must also be adaptable to the various roles it plays within the data center. To begin, this chapter examines the architecture of the Nexus 5500, followed by a look at how to configure QoS policies within NX-OS on this platform.

Cisco Nexus 5500 System Architecture

The Nexus 5500 supports a new set of QoS capabilities that are designed to provide per-system class-based traffic control. These new features include the following:

- **Lossless Ethernet:** Priority flow control (IEEE 802.1Qbb)
- **Traffic protection:** Bandwidth management (IEEE 802.1Qaz)
- **Configuration signaling to endpoints:** DCBX (part of IEEE 802.1Qaz)

To better understand how these features work, let's first examine the hardware architecture of the Nexus 5500.

Architectural Overview

The Nexus 5500 series is built upon a three-stage switching fabric:

- The ingress unified port controller (UPC)
- The crossbar fabric
- The egress UPC

Each UPC is responsible for traffic management of groups of eight 1/10GE ports. This includes buffering of packets, arbitration onto the crossbar fabric, and a host of other interface-related features. Figure 24-2 illustrates a high-level view of the Nexus 5500 architecture and how the ingress and egress UPCs connect to the crossbar fabric. Like other Cisco switching platforms, the Nexus 5000 and 5500 families do all packet processing in hardware application-specific integrated circuits (ASICs)—meaning that a packet is never managed directly by the control plane CPU.

Note Although there are some key QoS differences between the currently shipping Nexus 5500 and the older Nexus 5010 and 5020 switches, this chapter focuses primarily on the Nexus 5500 family specifically, so many of the features discussed in this chapter are not supported on the older 5010 and 5020 switches.

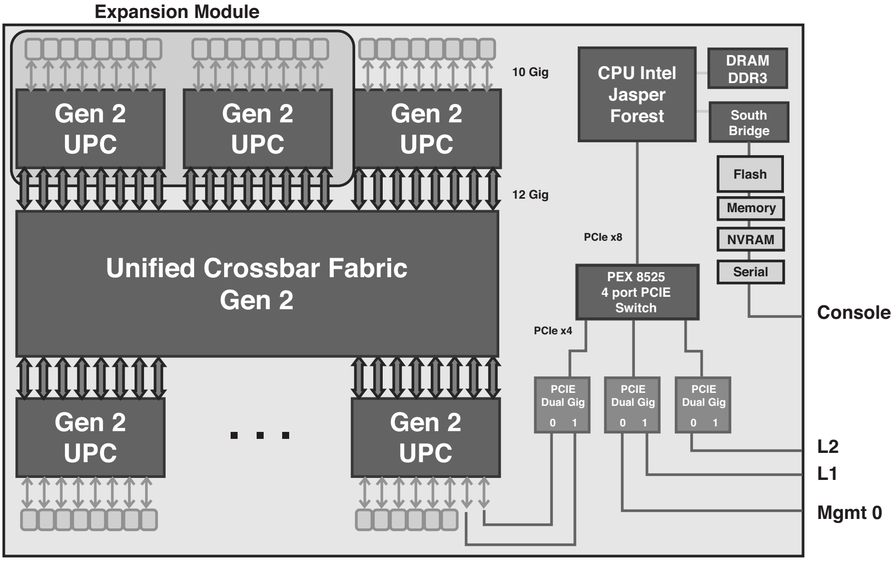


Figure 24-2 *Nexus 5500 System Architecture*

From a QoS perspective, all classification, marking, queuing, policing, and other traffic management features are done in hardware—either on the ingress UPC, crossbar fabric, or on the egress UPC. This becomes interesting when you consider how QoS is actually configured on the Nexus 5500.

The Nexus 5500 supports the following types of QoS policies:

- **qos:** Defines Modular QoS command-line interface (MQC) objects used for marking and policing
- **network-qos:** Defines the characteristics of network-wide QoS properties (such as used in data center bridging [DCB] networks) and should be applied consistently on all switches participating in the network
- **queuing:** Defines MQC objects used for queuing and scheduling, in addition to a limited set of marking objects
- **control-plane:** Defines MQC objects used for control plane policing (CoPP)

Type **qos**, **network-qos**, and **queuing** policies are attached to the three stages in the switching fabric, as shown in Figure 24-2. For example, the **qos** type policies are applied to either the ingress interface (actually processed on the ingress UPC) or on the system (the crossbar fabric). The **network-qos** policies are only applied to the system—meaning the crossbar fabric. The **queuing** policies, on the other hand, can be applied to either the ingress interface (ingress UPC), the egress interface (UPC), or the system (crossbar fabric).

Virtual Output Queuing

One of the most important QoS-related aspects of the Nexus 5500 architecture is that it is built on a virtual output queuing (VOQ) architecture. As you will have noted throughout this book, each Cisco switching platform is built in a different way. The switch architecture usually reflects the way the switch is to be used in the network and the types of congestion that are likely to be experienced at that place in the network. For example, the Catalyst 4500 (designed primarily as a *campus* access and aggregation switch) uses an egress-based queuing design where packets enter the switch and then are queued up on egress before they are transmitted. To support this architecture, the Catalyst 4500 linecards support large egress buffers.

Contrast this with how the Nexus 5500 queues packet in a buffer. Instead of placing queued packets in a large egress buffer, the egress port signals back to the ingress port to tell it that congestion has occurred. Therefore, instead of storing and buffering the packets at the egress port, they are buffered at all the ingress ports that are sending traffic to the congested port. When you consider that most data center designs include multiple ingress server access ports transmitting to a few—or one—egress ports (that is, incast traffic patterns), this model works well to accommodate such expected traffic patterns.

Even though the Nexus 5500 port buffers are relatively small (only 640 KB), VOQ can be extremely effective in absorbing traffic loads at congestion points within the data center. For example, egress-based queuing architectures (such as the Catalyst 6500) require large buffers at each egress port and may still be forced to drop a packet *after* it has transited the backplane. VOQ architectures, however, implement a large distributed buffer pool to manage congestion and can drop excessive traffic *before* it is transmitted across the backplane.

Putting this mathematically, the total queue size available to egress ports is equal to the total number of ingress ports times the queue depth per port. Consider an example where ten server ports are aggregating traffic toward a single uplink. Instead of relying on a single uplink port to buffer this significant load of traffic, the ten ingress ports effectively increase the buffer space for the sole uplink by ten, thus giving 10 × 640 KB = 6.4 MB of buffer space to the congested port.

Statistically, VOQ provides the same advantages as shared buffer memory architectures.

Figure 24-3 illustrates a hypothetical situation where three ingress ports are sending packets to a single egress port that has encountered congestion. Packets are held at the ingress port buffer until the egress queue frees up.

Note The Nexus 7000 M-Series and F-Series modules also use VOQ architectures, as is discussed in more detail in Chapter 25.

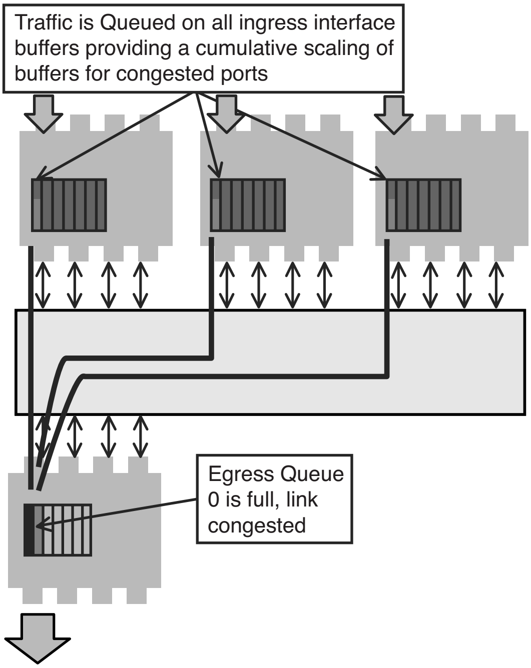

Traffic is Queued on all ingress interface buffers providing a cumulative scaling of buffers for congested ports
Egress Queue 0 is full, link congested

Figure 24-3 *Nexus 5500 Virtual Output Queuing*

The Nexus 5500 uses an eight-queue QoS model for managing unicast traffic. Therefore, each ingress port on the Nexus 5500 has a set of eight virtual output queues for each egress port—one for each class of service (CoS) value per egress port. For example, if a server is connected to interface Ethernet 1/8 and the destination is on ports 1/20 and 1/23, two VOQs are used on Ethernet 1/8, one for each of the two egress ports. In total, NX-OS supports 1024 ingress VOQs on each and every ingress port (8 classes × 128 destinations = 1024).

Note The 128 destinations is derived from the fact that a Nexus 5596 has 96 Ethernet ports, plus 2 optional Layer 3 daughter cards, each having 16 connections to the crossbar fabric, thus 96 + 16 + 16 = 128.

Management of the VOQ system is handled by centralized arbitration. The central arbiter is responsible for managing queues on each port and giving packets on ingress ports access to the crossbar fabric to create a switching path to the egress port. For example, Figure 24-3 illustrates a situation where Queue 0 on an egress port fills up while the other seven queues are still uncongested. The central arbiter is notified and signals to the three sending ports to buffer on ingress until the congestion has been cleared.

In this manner, centrally arbited per-CoS VOQs prevent head-of-line (HoL) blocking scenarios. For example, if a certain CoS value is experiencing congestion at its destination egress port and is signaled by the central arbiter to queue within its VOQ, it will not block other CoS values from transiting the fabric. These other CoS values will continue to access the fabric via their own respective VOQs.

One of the unique concepts on platforms that support VOQ, such as the Nexus 5500 and the Nexus 7000 linecards, is that when congestion reaches the point that packets start to get dropped, they are dropped on the ingress port, not the egress port. This is somewhat of a paradigm shift, especially if you are used to looking for packet drops on the egress port on products like the Catalyst switching family.

The **show queuing interface ethernet x/y** command is used to investigate packet drops on a Nexus 5500. Example 24-1 shows how to look for ingress packet drops on a Nexus 5500.

Example 24-1 *Investigating Packet Drops on a Nexus 5500*

```
N5K# show queuing interface ethernet 1/22
Interface Ethernet1/39 TX Queuing
qos-group  sched-type  oper-bandwidth
    0       WRR            50
    1       WRR            50
Interface Ethernet1/22 RX Queuing
qos-group  0
    q-size: 243200, HW MTU: 1600 (1500 configured)
```

```
drop-type: drop, xon: 0, xoff: 1520
Statistics:
    Pkts received over the port             : 85257
    Ucast pkts sent to the cross-bar        : 930
    Mcast pkts sent to the cross-bar        : 84327
    Ucast pkts received from the cross-bar  : 249
    Pkts sent to the port                   : 133878
    Pkts discarded on ingress               : 543
    Per-priority-pause status               : Rx (Inactive), Tx
```

Note in the output shown in Example 24-1 that packets are being discarded on ingress rather than on egress. The packets are being discarded because of the congestion experienced on the egress port. This congestion has caused a significant enough pileup of packets in the ingress VOQ that the UPC is forced to discard them.

QoS Groups and System Classes

In most IOS platforms, the policy map is attached to an interface. However, in NX-OS, it is also possible to attach policy maps to the system itself. In this way, **system qos** is a type of MQC target. From an architectural perspective, the **system qos** target is the crossbar fabric itself. You use a **service-policy** statement to associate a **policy map** with the **system qos** target. A **system qos** policy applies to all interfaces on the switch unless a specific interface has an overriding service policy configuration. The **system qos** policies are used to define system classes, the classes of traffic across the entire switch, and their attributes.

To ensure QoS consistency (and for ease of configuration), the switch distributes the **system qos** parameter values to all its attached converged network adapters (CNAs) using the Data Center Bridging Exchange (DCBX) protocol.

If service policies are configured at the interface level, the interface-level policy always takes precedence over system class configuration or defaults.

On the Cisco Nexus 5500 series switch, a system class is uniquely identified by a **qos-group** value. A total of six system classes are supported. Class 0 is reserved as the default class, but five additional system classes (Classes 1 to 5) can be created by the administrator and assigned to various traffic types. If Fibre Channel over Ethernet (FCoE) is in use, this is assigned as Class 1, leaving four configurable classes.

To summarize, the default system classes are as follows:

- **Drop system class:** By default, the software classifies all unicast and multicast Ethernet traffic into the default drop system class. This class is identified by QoS group 0. This class is created automatically when the system starts up (the class is named class-default in the CLI). You can neither delete this class nor change the match criteria associated with the default class. This class is also the default for all traffic (class-default), meaning any matched traffic ends up in QoS group 0.

- **FCoE system class:** All Fibre Channel and FCoE control and data traffic is automatically classified into the FCoE system class, which provides no-drop service. This class is created automatically when the FCoE feature is enabled (the class is named class-fcoe in the CLI). Like the Drop system class, you can neither delete this class nor modify the CoS value associated with this class. This class is always identified by QoS group 1.
- **Internetwork Control:** This class is inherent in the system and cannot be deleted. It is responsible for things like Simple Network Management Protocol (SNMP), routing protocol traffic, and so on. Internetwork Control traffic uses CoS 6.
- **Network Control:** This class is also inherent in the system and cannot be deleted. It is used for things like communicating to the Nexus 2000 Fabric Extender. Network Control class traffic is marked as CoS 6.

> **Note** In the older Nexus 5010 and 5020 switches, FCoE traffic is assigned by default to QoS group 1 and is given 50 percent of the available bandwidth. However, in the Nexus 5500 series, this class exists only if the FCoE feature is enabled.

Example 24-2 demonstrates the available QoS groups that may be configured by the network engineer. Notice that QoS group 0 is not available for configuration because this is reserved by the system drop class.

Example 24-2 *Setting the* ***qos-group*** *Value*

```
N5K(config-pmap-c-qos)# set qos-group ?
  <1-5>  QoS-group value
```

So, what exactly is the role of QoS groups in the Nexus 5500? The QoS groups are an internal label used to identify traffic so that it can be correctly managed by the various QoS mechanisms. It is these QoS groups that inherit configurable various QoS characteristics and are thus applied to queues.

Essentially, a QoS group is a set of characteristics for handling traffic within the system. Some attributes that define a QoS group are the maximum transmission unit (MTU) for the class, the bandwidth of traffic in that group, and the CoS value. QoS class maps are used to first match traffic patterns, and then any matched traffic is given all the properties of these QoS groups by a policy map. The characteristics of each QoS group are configured using the **network-qos** policy maps, which are discussed in the configuration sections later in this chapter.

In the Nexus 5500, there is no default CoS or DSCP to QoS group mappings (with the exception of CoS 3 to QoS group 1 for FCoE traffic, when **feature fcoe** is enabled). In NX-OS, CoS or DSCP values are mapped to QoS groups, not directly to queues.

Note that without a specific policy the default behavior of the switch is to assign all traffic to QoS group 0 (the default class). In addition, because NX-OS by default classifies traffic according to the CoS value, any packet that is not tagged with an 802.1p CoS value will end up in the default drop system class. Naturally, this could cause a problem if the attached server is not using an 802.1p trunk, meaning no CoS value is present. Of course, there are ways to deal with this, such as setting up a specific classification policy that matches on incoming differentiated services code point (DSCP) or that uses an access control list (ACL).

As with other Cisco switching platforms, it is preferred to classify traffic based on DSCP if at all possible.

These configuration elements are discussed in the following sections.

QoS Design Steps

In NX-OS, QoS is enabled by default; in fact, it is not even possible to turn it off. All the Nexus switching platforms make extensive use of the MQC framework for nearly all QoS configurations. Even system and interface-level QoS features are enabled via MQC. One advantage of this approach is that there is less repetition of configuration commands on each interface because they are all applied through policy maps, attached to either an interface or to the system as a whole.

Configuration of QoS on the Nexus 5500 in the role of a data center access or aggregation switch is accomplished by following these steps:

1. Configure ingress QoS models, including the following:
   a. Trust models
   b. Classification and marking models
   c. Ingress policing models
2. Configure egress/VOQ queuing model.
3. Configure QoS to support optional designs.

Each of these configuration steps is discussed in the following sections.

Ingress QoS Models

The ingress QoS models include the following:

- Trust
- Classification and marking
- Policing (optional)

Each of these ingress QoS models is discussed in turn.

Trust Models

Unlike most access networks, data center access ports (connected to servers) are generally trusted. When new servers are built and dynamically generated as virtual machines (VMs), the expectation is that the server application correctly marks the DSCP values of its traffic. Therefore, trust is commonly accepted. Sometimes, however, applications either do not mark DSCP or CoS values for traffic classes or the markings used do not line up with the strategic QoS model.

This section examines the configuration of both the trusted and untrusted QoS server models.

Trusted Server Model

By default, all Ethernet interfaces are trusted. In other words, the DSCP and CoS values are preserved unless marking policies are configured to specifically overwrite the QoS values. Therefore, if the servers are trusted, no explicit configuration is required.

Untrusted Server Model

When the server is untrusted, the objective is to reset the QoS markings to zero and to ensure that all traffic from such a server is sent to the default class. In NX-OS lingo, on the Nexus 5500, this means that all traffic from an untrusted server is mapped to QoS group 0 and the DSCP is set to 0 regardless of the original packet QoS markings.

In this case, the intention is to match all traffic, meaning if a specific class map is not used, all traffic will fall into the class-default classifier. The default behavior for class-default is to assign all traffic matched by it to QoS group 0, which is the desired result here. Because class-default always exists, there is no need to create a class map. Example 24-3 shows how to create a **qos** policy map that uses class-default to reset all incoming traffic to DSCP 0 and assigns it to QoS group 0.

Example 24-3 *Configuring a Policy Map for an Untrusted Server*

```
N5K(config)# policy-map type qos NO-TRUST
N5K(config-pmap-qos)# class type qos class-default
N5K(config-pmap-c-qos)# set dscp 0
 ! For matched traffic (all traffic), the DSCP is to 0
```

With the **qos** type policy map, it is not possible to re-mark the CoS value, which is used for internal queuing. Setting/resetting CoS values is done with a **network-qos** type policy map.

The next configuration step is to apply the QoS policy to a target. The Nexus 5500 offers the option of attaching the QoS policy to either the system class, meaning that it will be attached to all interfaces, or simply to a group of interfaces. Example 24-4 illustrates how to apply the **qos** type policy accordingly.

Example 24-4 *Attaching the QoS Policy Map to Different Targets*

```
 ! This section applies the NO-TRUST policy to the system (globally)
N5K(config)# system qos
 ! This command enables system qos (global) configuration mode
N5K(config-sys-qos)# service-policy type qos input NO-TRUST
 ! Applies the NO-TRUST policy globally
 ! This section applies the NO-TRUST policy to an interface
N5K(config)# interface ethernet 1/1-15
N5K(config-if-range)# service-policy type qos input NO-TRUST
ERROR: policy already attached at system level
 ! This is a redundant configuration and is prevented in NX-OS
```

Example 24-4 shows how to first attach the **qos** type policy to the entire system class, and then to a range of interfaces. As this example shows, if the **qos** policy is already attached to the interface, NX-OS does not let you attach it to a group of interfaces (because this would be redundant). Although not shown here, it also possible to apply the **qos** type policy to a VLAN if desired.

However, an important point to remember is if multiple different **qos** policies are configured on the Nexus 5500, the interface level policy map will always take precedence over the less-specific **system qos** policy map.

You can verify the policy map with the following commands:

- **show policy-map** [*name*] (as shown in Example 25-5)
- **show policy-map interface Ethernet x/y** (as shown in Example 25-6)

Example 24-5 *Verifying the Policy Map*

```
N5K# show policy-map NO-TRUST
  Type qos policy-maps
  ====================
  policy-map type qos NO-TRUST
    class type qos class-default
      set qos-group 0
      set dscp 0
 ! Note how the QoS group is set to zero by the default class map
```

Example 24-6 *Verifying the Interface Policy Map*

```
N5K# show policy-map interface ethernet1/13
Global statistics status :   disabled
Ethernet1/13
```

```
  Service-policy (qos) input:   NO-TRUST
    policy statistics status:   disabled
    Class-map (qos):   class-default (match-any)
      Match: any
      set qos-group 0
      set dscp 0
```

Classification and Marking Models

Classification on the Nexus 5500 is done at the ingress port. The platform can classify on a variety of different metrics, but ultimately CoS is used to map to the eight VOQs per port. The Nexus 5500 reserves CoS 6 for internetwork control and CoS 7 is reserved for control traffic transiting the fabric extenders. If FCoE is configured on the system, a CoS value of 3 is used.

Table 24-1 summarizes how the eight CoS values are used in the system.

Table 24-1 *CoS Value Assignments in the Nexus 5500*

CoS Value	Availability	Use
7	Reserved by system	Control
6	Reserved by system	Internetwork control
5	Available for use	Voice
4	Available for use	Video
3	Reserved for FCoE when the feature is enable, otherwise available for use	FCoE
2	Available for use	Transactional data
1	Available for use	Background data
0	Reserved	Best effort

Unlike many other traffic types, FCoE only exists at the data link layer, meaning that there is no IP DSCP value associated with FCoE traffic. In the Nexus product line, FCoE traffic is identified and classified based on CoS 3.

Note The Nexus 5000 can identify FCoE traffic based on Ethertype. However, for purposes of QoS classification, only CoS values are used. By way of contrast, the Nexus 7000 can use either Ethertype or CoS to classify FCoE for use in a QoS policy.

You must exercise caution when using CoS-based classification in the Nexus 5548 because signaling traffic also uses CoS 3 (by default). If signaling traffic is classified as FCoE, because of CoS 3 overlap, it will end up being treated as if it were FCoE and given a lossless service. To keep these applications separate, so that the lossless queue can be dedicated to FCoE, it is better to classify traffic based on DSCP, which is the model followed in the examples shown in this chapter. However, if FCoE is not used on the Nexus 5500, CoS 3 can be reclaimed for other purposes, such as classification of signaling traffic.

The following sections examine how to classify server application traffic into different QoS groups, thus giving them differentiated levels of service.

Single-Application Server Model

The Single-Server Application Model is similar to the Untrusted Server Model, with two exceptions:

- All traffic is marked to a nonzero DSCP value.
- The class-default cannot be used because this only maps to the drop class in the Nexus 5500 (which will automatically set its QoS group to 0).

Consider an example of a server that is running a web-based customer relationship management (CRM) application (classified as transactional data). This equates to a DSCP value of AF21 and a CoS value of 2. Because all incoming data is of one type, a simple subnet matching ACL can be used to correctly classify traffic and then map it to a customized QoS group.

Although it might seem natural to allow class-default to manage single-server application traffic, this introduces several limitations, primarily that it is subject to more drops than any other class. Furthermore, it is not possible to change the QoS group assignment for class-default—meaning that class-default is always assigned to QoS group 0 and a CoS value of 0. If any special QoS handling is required, or if there are different types of application servers connected to this switch, each traffic type should be mapped its own QoS group.

In Example 24-7, an IP ACL matches traffic coming from the server. In this case the ACL is quite specific. However, you could also use an ACL that matches all traffic (**permit ip any any**), although this reduces flexibility if the policy map is applied to the system class.

The first step is to configure the class map that correctly matches the incoming traffic.

Example 24-7 *Configuring the QoS Class Map*

```
N5K(config)# ip access-list CRM-SERVER
N5K(config-acl)# permit ip 50.1.1.0/24 any
! Define an ACL to match traffic
N5K(config)# class-map type qos CRM-CLASS
```

```
N5K(config-cmap-qos)# match access-group name CRM-SERVER
 ! Use the ACL as the match criteria
```

The next step is to assign matched traffic to the desired QoS group. There is no hard-and-fast rule as to which QoS group traffic types get assigned to, as long as you remember that you are limited to five customizable groups (groups 1–5). The section on egress queuing models later in this chapter explores some ideas of how to map the four-class and eight-class generic QoS models to these five QoS groups.

Note The number of configurable QoS groups differs depending on whether FCoE is in use. If it is in use, only groups 2–5 are available. However, if FCoE is not in use, groups 1–5 are available for use.

Example 24-8 illustrates how to configure the policy map to set the QoS group to 2 and DSCP to AF21 for all traffic sourced from these servers.

Example 24-8 *Configuring the qos Type Policy Map*

```
N5K(config)# policy-map type qos CRM-POLICY
N5K(config-pmap-qos)# class type qos CRM-CLASS
N5K(config-pmap-c-qos)# set qos-group 2
 ! Map all matched traffic to QoS group 2
N5K(config-pmap-c-qos)# set dscp af21
 ! Set the DSCP value to AF21
```

At this point, it is recommended to verify the correct configuration of your **qos** class and policy maps, which you can do with the following commands

- **show class-map type qos**
- **show policy-map type qos**

Example 24-9 demonstrates the output from these commands.

Example 24-9 *Verify the qos Type Class and Policy Map Configuration*

```
N5K# show class-map type qos
  Type qos class-maps
  ===================
    class-map type qos match-all CRM-CLASS
      match access-group name CRM-SERVER
    class-map type qos match-any class-fcoe
      match cos 3
    class-map type qos match-any class-default
      match any
N5K# show policy-map type qos
  Type qos policy-maps
  ====================
  policy-map type qos CRM-POLICY
    class type qos CRM-CLASS
      set qos-group 2
      set dscp af21
    class type qos class-default
      set qos-group 0
  policy-map type qos fcoe-default-in-policy
    class type qos class-fcoe
      set qos-group 1
    class type qos class-default
      set qos-group 0
```

Notice from these outputs that **class-fcoe** is listed even though it has not been explicitly configured. In the Nexus 5010 and 5020 switches, the **fcoe** class is always on by default and is mapped to Qos group 1. (This cannot be modified when the **fcoe** feature is enabled). In the Nexus 5500 series, **class-fcoe** is optional so that QoS group 1 may be used for other purposes if the FCoE feature is not enabled. In addition, if **feature fcoe** is enabled, the **qos**, **network-qos**, and **queuing** type class maps for FCoE are automatically enabled and show up in the configuration.

Once the **qos** type policies have been configured, the next step is to attach them to an interface. Example 24-10 shows how to add this policy map to a range of interfaces.

Example 24-10 *Attaching the* ***qos*** *Type Policy Map to a Range of Interfaces*

```
N5K(config)# interface ethernet 1/15-18
N5K(config-if-range)# service-policy type qos input CRM-POLICY
 ! Attaches the policy map to the range of interfaces
```

Now that the classification and marking portions of the configuration are finished, the next steps involve enabling the QoS group to which this traffic is added. *If this step is missed, the queue for this class is not enabled and QoS does not work.* It is necessary to first activate and configure the new QoS group before it can be used. Example 24-11 illustrates this procedure.

Example 24-11 *Configuring the* ***network-qos*** *Type Class and Policy Maps*

```
 ! Step 1 – configure the network-qos class map
N5K(config)# class-map type network-qos CLASS-2
```

```
N5K(config-cmap-nq)# match qos-group 2
 ! Match on all traffic mapped to QoS group 2
 ! Step 2 - configure the network-qos policy map
N5K(config)# policy-map type network-qos NQ-POLICY
N5K(config-pmap-nq)# class type network-qos CLASS-2
 ! Enable the class - essentially turning on the queue
N5K(config-pmap-nq)# set cos 2
 ! Manually set the cos value for this class
 ! Step 3 - attach the policy map to the system class
N5K(config-pmap-nq-c)# system qos
N5K(config-sys-qos)# service-policy type network-qos NQ-POLICY
```

Once the policy map for the QoS group is attached to the system class, it is activated. In this example, the policy map does nothing more than activate the class and set the CoS value, but **network-qos** policy maps are also used to set a host of other features including setting the MTU, ingress buffer size, and no-drop behavior.

Note that the only place where CoS values can be set for egress 802.1Q frames is in the **network-qos** type policy map. The Nexus 5500 does not support a default DSCP-to-CoS mapping scheme, so unless a CoS value is manually set for the custom queues, all frames will get marked with a CoS value of zero.

You can verify the configuration in Example 24-11 with the following commands:

- **show class-map type network-qos**
- **show policy-map type network-qos**
- **show policy-map** [**interface** | **system**]

Multi-Application Server Model

Now consider an example where there are several applications running on a server in the data center. In this example, a multimedia server is using two types of traffic that need to be handled with a different level of service. Table 24-2 outlines the QoS markings and the corresponding QoS group memberships that are configured on the Nexus 5500.

Table 24-2 *An Example of Multi-Application Server QoS Handling*

Application Description	Set DSCP Value	QoS group Assignment
Media flows	AF41	QoS group 4
Signaling	CS3	QoS group 3
Transactional	AF21	QoS group 2
FCoE	N/A	QoS group 1 (system default group)
Best effort	DF	QoS group 0 (system default group)

Example 24-12 shows the **qos** class and policy map configuration steps.

Example 24-12 *Configuring the* ***qos*** *Type Class and Policy Maps*

```
 ! This section configures the class maps
N5K(config-cmap-qos)# class-map type qos MEDIA-FLOWS
N5K(config-cmap-qos)#  match access-group name MEDIA-FLOWS
 ! Media flows are identified via MEDIA-FLOWS ACL
 ! (not shown here for simplicity)
N5K(config-cmap-qos)# class-map type qos SIGNALING
N5K(config-cmap-qos)#  match access-group name SIGNALING
 ! Signaling traffic is identified via SIGNALING ACL
N5K(config-cmap-qos)# class-map type qos TRANSACTIONAL-DATA
N5K(config-cmap-qos)#  match access-group name TRANSACTIONAL-DATA
 ! Transactional data is identified via TRANSACTIONAL-DATA ACL
 ! This section configures the classification and marking policy map
N5K(config-cmap-qos)# policy-map type qos MULTI-APP-SERVER
N5K(config-pmap-qos)#  class type qos MEDIA-FLOWS
N5K(config-pmap-c-qos)# set qos-group 4
 ! Media flows are mapped to QoS group 4
N5K(config-pmap-c-qos)# set dscp af41
 ! Media flows are marked DSCP AF41
N5K(config-pmap-c-qos)# class type qos SIGNALING
N5K(config-pmap-c-qos)# set qos-group 3
 ! Signaling traffic is mapped to QoS group 3
N5K(config-pmap-c-qos)# set dscp cs3
 ! Signaling flows are marked DSCP CS3
N5K(config-pmap-c-qos)# class TRANSACTIONAL-DATA
N5K(config-pmap-c-qos)# set qos-group 2
 ! Transactional data is mapped to QoS group 2
N5K(config-pmap-c-qos)# set dscp af21
 ! Transactional data is mapped DSCP AF21
N5K(config-pmap-c-qos)# class type qos class-default
N5K(config-pmap-c-qos)# set dscp 0
 ! For matched traffic (all traffic), set the DSCP to 0
 ! This section attaches the policy map to the correct interface
N5K(config)# interface ethernet 1/5
N5K(config-if-range)# service-policy type qos input MULTI-APP-SERVER
 ! Attaches the marking and QoS group policy to the correct interface
```

You can verify the configuration in Example 24-12 with the following commands:

- **show class-map type qos**
- **show policy-map type qos**

- **show policy-map interface**

As before, the next step is to configure the **network-qos** class and policy maps, as demonstrated in Example 24-13.

Example 24-13 *Configuring the network-qos Type Class and Policy Maps*

```
 ! Step 1 - configure the class maps
N5K(config)# class-map type network-qos CLASS-2
N5K(config-cmap-nq)# match qos-group 2
 ! Matches all traffic in QoS group 2
N5K(config-cmap-nq)# class-map type network-qos CLASS-3
N5K(config-cmap-nq)# match qos-group 3
 ! Matches all traffic in QoS group 3
N5K(config-cmap-nq)# class-map type network-qos CLASS-4
N5K(config-cmap-nq)# match qos-group 4
 ! Matches all traffic in QoS group 4
 ! Step 2 - configure the network-qos policy map
N5K(config)# policy-map type network-qos NQ-POLICY
N5K(config-pmap-nq)# class type network-qos CLASS-2
 ! Enables CLASS-2
N5K(config-pmap-nq)# class type network-qos CLASS-3
 ! Enables CLASS-2
N5K(config-pmap-nq)# class type network-qos CLASS-4
 ! Enables CLASS-4
 ! Step 3 - attach the policy map to the system class
N5K(config)# system qos
N5K(config-sys-qos)# service-policy type network-qos NQ-POLICY
 ! Attaches the network-qos policy to the system class
```

You can verify the configuration in Example 24-13 with the following commands:

- **show class-map type network-qos**
- **show policy-map network-qos**
- **show policy-map system**

Application Policing Server Model

In some cases, it is necessary to configure an ingress policer on an interface. Note that in IOS-based systems and in the Nexus 7000 series there is an option to re-mark the DSCP if the policer detects a violation. However, this is not possible on the Nexus 5500 platform. At the time of this writing, the only option available is to drop if the specified police rate is exceeded.

The steps required to configure ingress policing are as follows:

1. Configure the **qos** type class map.
2. Configure the **qos** type policy map with the police option set.
3. Attach the **qos** type policy map to an ingress interface.

Similar to IOS, traffic is policed in a policy map after being matched by a class map. One significant difference is apparent, however: With the Nexus 5500, it is only possible to police on ingress, not egress. Considering that the Nexus 5500 uses an ingress-based buffering system, this is useful for protecting the upstream buffers.

Example 24-14 highlights how to configure a policy map that supports ingress policing.

Example 24-14 *An Ingress Policing Policy Map*

```
N5K(config)# policy-map type qos POLICE
N5K(config-pmap-qos)# class class-default
 ! Match traffic in the default-class
N5K(config-pmap-c-qos)# police cir percent 50 bc 20 mbytes conform transmit violate
drop
 ! Police traffic in this class to 50% of bandwidth
N5K(config)# interface ethernet 1/10
N5K(config-if)# service-policy type qos input POLICE-MAP
 ! Attach the policy map to an interface
```

In this example, traffic is policed down to 50 percent of the available bandwidth with a committed burst rate of 20 MB. Note that the **conform** action in this example is to transmit and the **violate** action is to drop.

You can verify the ingress policer with the following commands:

- **show policy-map** [*policy-map-name*]
- **show policy-map interface Ethernet x/y**

Example 24-15 shows a how to verify the ingress policer.

Example 24-15 *Verifying the Ingress Policy Map*

```
N5K# show policy-map POLICE
  Type qos policy-maps
  ====================
  policy-map type qos POLICE
    class type qos CLASS-2
      set qos-group 4
      police cir percent 15 bc 20 mbytes conform transmit violate drop
    class type qos class-default
      set qos-group 0

N5K# show policy-map interface ethernet 1/10
Global statistics status :   disabled
Ethernet1/10
  Service-policy (qos) input:   POLICE
    policy statistics status:   disabled
    Class-map (qos):   POLICE (match-all)
      0 packets
      Match: cos 4-6
      police cir percent 100 bc 200 ms
        conformed 0 bytes, 0 bps action: transmit
        violated 0 bytes, 0 bps action: drop
```

Modifying the Ingress Buffer Size

With the default QoS configuration, the entire available ingress buffer (470 KB) is allocated to class-default. When you create a new QoS group, the buffer required for the new QoS group is taken away from class-default. The amount of buffer that is left for class-default is governed by the following equation:

- $470KB - \sum [28.6KB + B] * N$

Where:

- B = Ingress buffer size of each traffic class configured (see Table 24-3)
- N = The number of QoS groups in the system

Table 24-3 summarizes the default ingress buffer allocations (queue limits) used in the system.

Table 24-3 *Default Ingress Buffer Allocations on the Nexus 5500*

Traffic Class	Ingress Buffer (KB)
Class-fcoe (not used unless the fcoe feature is enabled)	79.360
User-defined no-drop with an MTU less than 2240	79.360
User-defined no-drop class with an MTU greater than 2240	90.204
Tail-drop traffic class	22.720
Class-default	All the remaining buffer (470 with default QoS configuration)

Typically, the queue limits (buffer allocations) are assigned proportionally to the bandwidth assignments in the egress queuing model. In other words, if a certain class is assigned 25 percent of the bandwidth, it follows that it is also assigned 25 percent of the ingress buffer space. The buffer assignment for the priority queue (PQ), however, can be inversely proportional because it is serviced in real time and therefore needs fewer buffers, thus freeing buffers that may be reallocated to lower-priority queues that are serviced less often and which may therefore fill deeper.

Example 24-16 shows how to adjust the ingress buffer size for two classes. In this example, the Transactional Data class is given a system-wide ingress buffer size of 85000 bytes and the Real-Time class is given 30000 bytes.

Example 24-16 *Adjusting the Ingress Buffer Size*

```
 ! Step 1 - Define a QoS class map
N5K(config)# class-map type qos TRANSACTIONAL-DATA
N5K(config-cmap-qos)# match af21
 ! Transactional Data is matched with AF21
N5K(config-cmap-qos)# class-map type qos REALTIME
N5K(config-cmap-qos)# match dscp ef
 ! Realtime traffic is matched with EF
 ! Step 2 - Define a QoS Policy Map
N5K(config)# policy-map type qos POLICY-QOS
N5K(config-pmap-qos)# class type qos TRANSACTIONAL-DATA
N5K(config-pmap-c-qos)# set qos-group 2
 ! Set transactional data to QoS group 2
N5K(config-pmap-c-qos)# class type qos REALTIME
N5K(config-pmap-c-qos)# set qos-group 5
 ! Map Realtime traffic to QoS group 5
 ! Step 3 - Apply the QoS policy map to the system target
N5K(config)# system qos
N5K(config-sys-qos)# service-policy type qos input POLICY-QOS
 ! Attaches the policy map to the system
 ! Step 4 - Define a network-qos Class Map
N5K(config)# class-map type network-qos TRANSACTIONAL-DATA
N5K(config-cmap-nq)# match qos-group 2
 ! Match traffic belonging to QoS group 2 (transactional data)
N5K(config-cmap-nq)# class-map type network-qos REALTIME
N5K(config-cmap-nq)# match qos-group 5
 ! Match traffic belonging to QoS group 5 (real-time traffic)
 ! Step 5 - Define a network-qos policy map
N5K(config)# policy-map type network-qos POLICY-NQ
N5K(config-pmap-nq)# class type network-qos TRANSACTIONAL-DATA
N5K(config-pmap-nq-c) queue-limit 85000 bytes
 ! This is an arbitrary value, for demonstration purposes only
```

```
N5K(config-pmap-nq-c)# class type network-qos REALTIME
N5K(config-pmap-nq-c) queue-limit 30000 bytes
 ! Lower the queue-limit for real-time traffic
 ! Step 6 – Apply the network-qos policy map to the system target
N5K(config)# system qos
N5K(config-sys-qos)# service-policy type network-qos POLICY-NQ
 ! Applies the policy map system-wide (to all interfaces)
```

Egress Queuing Models

Three QoS reference class models are presented throughout this book: these are the 4-class, 8-class, and 12-class models. However, because the Nexus 5000 family only supports four nonsystem queues available to you (plus the FCoE and default classes), only the four-class and the (modified) eight-class models are presented here.

The adaptation of these two reference models for egress queuing is discussed in the following two sections.

Four-Class Model

Because the Nexus 5000 family supports four user-defined queues, plus one for FCoE and one for default, it is easy to adapt the four-class QoS model to this platform. Figure 24-4 illustrates a suggested mapping of traffic classes to QoS groups in the Nexus 5500. Each QoS group corresponds to one egress queue.

It is important to remember that the QoS group numbering is arbitrary, except or QoS groups 0 and 1, which are hard-wired to the default-class and the FCoE class, respectively; however, an attempt has been made to bring some logical order to the this mapping scheme.

Note that Figure 24-4 actually shows seven classes. This includes the normal four classes plus FCoE and the two reserved classes. FCoE is a special case, and it exists only within the confines of the data center (because Fibre Channel traffic is only ever linked between a host and storage-area network [SAN] target). If the FCoE feature is not enabled, this would result in only four classes.

Figure 24-4 illustrates the mapping of three user-defined queues to the appropriate QoS group, along with four system-defined queues. The top two, Internetwork Control and Network Control, identified by CoS values 6 and 7, respectively, are assigned to reserved system queues and are used for control plane traffic types. For example, Class 6 is used for control traffic such as Link Aggregation Control Protocol (LACP), Internet Group Management Protocol (IGMP), and so on. Class 7 is used for control traffic such as Fabric Shortest Path First (FSPF), Spanning Tree Protocol (STP), and communication to the fabric extenders.

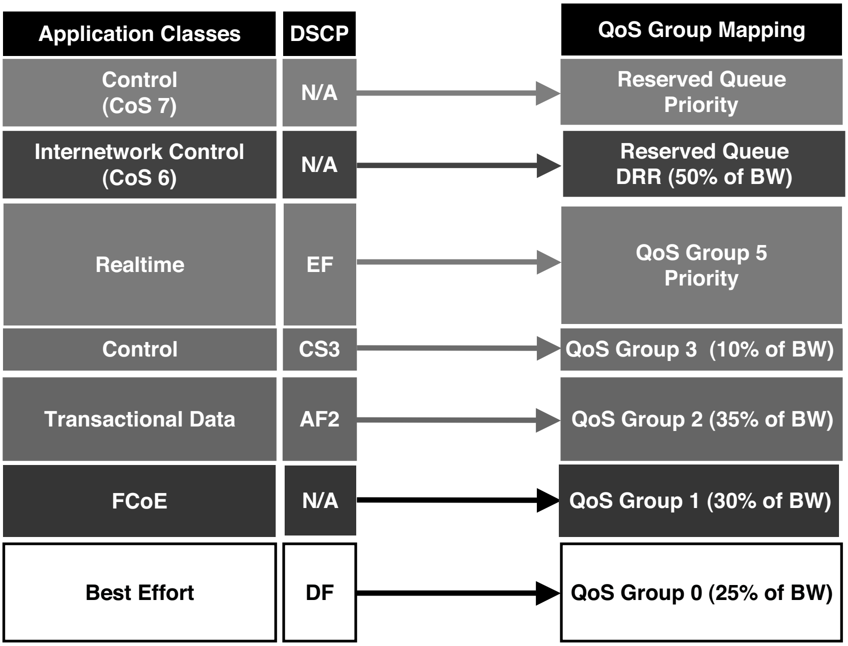


Figure 24-4 *Adapting the Four-Class Model to the Nexus 5500*

Class 7 traffic is treated as a strict-priority queue, while Class 6 is serviced as a deficit round -robin (DRR) queue with 50 percent of the bandwidth. The remaining user-defined queues are serviced with deficit weighted round-robin (DWRR). Although seven queues are shown in total here, eight egress queues are actually available on the Nexus 5500. With the four-class model, only seven queues are necessary, thus leaving one (QoS group 4) unused.

The configuration of the Nexus 5500 to support this design model is highlighted in the examples that follow. The QoS configuration can be accomplished by following these four steps:

1. Configure the **qos** type class and policy maps.
2. Configure the **network-qos** type class and policy maps.
3. Configure the **queuing** type class and policy maps.
4. Apply all the service policies to the system QoS class.

Example 24-17 illustrates the configuration of the **qos** type class and policy maps.

Example 24-17 *Configuring the* ***qos*** *Type Class and Policy Maps to Support the Four-Class QoS Model*

```
 ! Configure the qos class maps
N5K(config)# class-map type qos match-any REALTIME
N5K(config-cmap-qos)# match dscp ef
 ! Real-time traffic is matched by DSCP EF
N5K(config-cmap-qos)# class-map type qos match-any CONTROL
N5K(config-cmap-qos)# match dscp cs3
 ! Control traffic is matched by DSCP CS3
N5K(config-cmap-qos)# class-map type qos match-any TRANSACTIONAL-DATA
N5K(config-cmap-qos)# match dscp af21 af22 af23
 ! Transactional data is matched by DSCP AF2
 ! Configure the qos policy map
N5K(config)# policy-map type qos 4-CLASS-QOS-POLICY
N5K(config-pmap-qos)# class REALTIME
N5K(config-pmap-c-qos)# set qos-group 5
 ! Maps real-time traffic to QoS group 5
N5K(config-pmap-c-qos)# class CONTROL
N5K(config-pmap-c-qos)# set qos-group 3
 ! Maps control traffic to QoS group 3
N5K(config-pmap-c-qos)# class TRANSACTIONAL-DATA
N5K(config-pmap-c-qos)# set qos-group 2
 ! Maps transactional data traffic to QoS group 2
N5K(config-pmap-c-qos)# class class-fcoe
N5K(config-pmap-c-qos)# set qos-group 1
 ! Maps FCoE traffic to QoS group 1
 ! Note - you don't need to configure a class map for FCoE
```

Note Beginning with Cisco NX-OS Release 5.0(2)N1(1), for the Cisco Nexus 5500 family of switches there are four predefined policy maps for FCoE that must be enabled in the system QoS class to allow FCoE QoS to work, as follows::

```
service-policy type qos input fcoe-default-in-policy
service-policy type queuing input fcoe-default-in-policy
service-policy type queuing output fcoe-default-out-policy
service-policy type network-qos fcoe-default-nq-policy
```

Before you enable FCoE on the Nexus 5548 switch, you must enable **class-fcoe** in all three types of QoS policies.

Note that even though FCoE is being used in this example, no class map is defined for it. When the FCoE feature is enabled, the Nexus 5500 automatically classifies FCoE traffic without the need for an explicit class map, so there is no need to configure it. This also applies for the **network-qos** and **queuing** classification steps. When it comes to the policy map, the predefined class map for FCoE traffic can be easily referenced.

Once the **qos** class and policy maps are configured, it is now time to configure the **network-qos** class and policy maps.

Example 24-18 illustrates how to enable the QoS groups corresponding to each traffic class.

Example 24-18 *Configuring the* ***network-qos*** *Type Class and Policy Maps to Support the Four-Class QoS Model*

```
 ! Configure the network-qos class maps
N5K(config)# class-map type network-qos REALTIME
N5K(config-cmap-nq)# match qos-group 5
 ! Matches traffic in QoS group 5
N5K(config-cmap-nq)# class-map type network-qos CONTROL
N5K(config-cmap-nq)# match qos-group 3
 ! Matches traffic in QoS group 3
N5K(config-cmap-nq)# class-map type network-qos TRANSACTIONAL-DATA
N5K(config-cmap-nq)# match qos-group 2
 ! Matches traffic in QoS group 2
 ! Configure the network-qos policy map
N5K(config)# policy-map type network-qos 4-CLASS-NQ-POLICY
N5K(config-pmap-nq)# class type network-qos REALTIME
 ! Enable the Real-Time class
N5K(config-pmap-nq-c)# class type network-qos CONTROL
 ! Enable the Control class
N5K(config-pmap-nq-c)# class type network-qos TRANSACTIONAL-DATA
 ! Enable the Transactional Data class
N5K(config-pmap-nq-c)# class type network-qos class-fcoe
N5K(config-pmap-nq-c)# pause no-drop
N5K(config-pmap-nq-c)# mtu 2158
 ! This last policy creates a PFC class with the correct MTU for FCoE
```

Note In this policy, the FCoE MTU is set to 2158 bytes. It is important to understand that although the FC and FCoE payload MTU is 2112 bytes, when the FCoE header, CRC, and EOF fields are added, the total MTU adds up to 2158 bytes.

After the **qos** and **network-qos** policies have been configured, the next step is to configure the queuing policies for each traffic class, as demonstrated in Example 24-19.

Example 24-19 *Configuring the **queuing** Type Class and Policy Maps to Support the Four-Class QoS Model*

```
 ! Configure the queuing class maps
N5K(config-cmap-que)# class-map type queuing TRANSACTIONAL-DATA
N5K(config-cmap-que)# match qos-group 2
 ! Matches traffic in QoS group 2
N5K(config-cmap-que)# class-map type queuing CONTROL
N5K(config-cmap-que)# match qos-group 3
 ! Matches traffic in QoS group 3
N5K(config-cmap-que)# class-map type queuing REALTIME
N5K(config-cmap-que)# match qos-group 5
 ! Matches traffic in QoS group 5
 ! Configure the queuing policy map
N5K(config)# policy-map type queuing 4-CLASS-GLOBAL-QUEUING-POLICY
N5K(config-pmap-que)# class type queuing REALTIME
N5K(config-pmap-c-que)# priority
 ! Assign EF traffic to the strict-priority queue
N5K(config-pmap-c-que)# class type queuing CONTROL
N5K(config-pmap-c-que)# bandwidth percent 10
 ! Assign 10% of bandwidth to control
N5K(config-pmap-c-que)# class type queuing TRANSACTIONAL-DATA
N5K(config-pmap-c-que)# bandwidth percent 35
! Assign 35% of bandwidth to transactional data
N5K(config-pmap-c-que)# class type queuing class-fcoe
N5K(config-pmap-c-que)# bandwidth percent 30
 ! Here FCoE is given 30% of the available bandwidth
N5K(config-pmap-c-que)# class type queuing class-default
N5K(config-pmap-c-que)# bandwidth percent 25
 ! Assign 25% of bandwidth to the default queue
```

In the preceding example, real-time traffic is assigned to the strict-priority queue. In this platform, it is possible to assign only one class to the PQ, and all others are serviced with DWRR. You will note that the PQ is an unbounded queue on the Nexus 5500 series, meaning that there is no upper limit to how much of the bandwidth can be consumed by it. This raises a serious concern in situations where excessive amounts of high-priority traffic has the potential to completely starve out the other queues. Therefore, take care as to how much traffic is mapped into this class.

Note in the preceding example that the FCoE class is given a specific bandwidth of 30 percent. By default, the Nexus 5500 gives FCoE 50 percent of available bandwidth, so it is necessary to adjust the value here.

The final configuration step is to attach these three policy maps to the system. This activates the policies for all interfaces on the Nexus 5500. Example 24-20 demonstrates this step.

Example 24-20 *Applying the QoS Policies to the System Class*

```
N5K(config)# system qos
N5K(config -sys-qos)# service-policy type qos input 4-CLASS-QOS-POLICY
  ! Attaches the qos policy to the system class
N5K(config -sys-qos)# service-policy type queuing output 4-CLASS-GLOBAL-QUEUING-
POLICY
  ! Attaches the queuing policy on the output direction
N5K(config -sys-qos)# service-policy type network-qos 4-CLASS-NQ-POLICY
  ! Attaches the network-qos policy map to the system
```

Notice that the service policy is applied in both the ingress and egress directions. The queuing policy applied in the ingress (**input**) direction defines the ETS queuing policies for DCBX capable devices, such as FEXs or CNAs. When the queuing policy is applied in the egress (**output**) direction, scheduling in the VOQs is activated.

You can verify the configuration in Example 24-20 with the following commands:

- **show policy-map type** [**qos** | **network-qos** | **queuing**]
- **show policy-map system**

Eight-Class Model

Although the Nexus 5500 only offers up to six custom queues (including the FCoE and default classes), it is still possible to map the eight-class reference model to fit your QoS design reasonably well. For this to work, however, you need to make some modifications.

Figure 24-5 illustrates a suggested design model to adapt the eight-class QoS reference design to the Nexus 5500.

Considering an example where FCoE is in use, only four custom queues are available. (QoS groups 0 and 1 are reserved, leaving QoS groups 2–5 available for use.) In the data center where the servers are generally trusted, it is safe to assume that scavenger traffic is not present, so this class is eliminated from this reference model. Network control traffic is naturally mapped to the reserved system PQ used for this purpose. Therefore, to accommodate four custom queues, one of the standard seven remaining classes must share a common QoS group. A reasonable design here would be to group interactive video and streaming video into a common QoS group (shown as QoS group 4 in Figure 24-5).

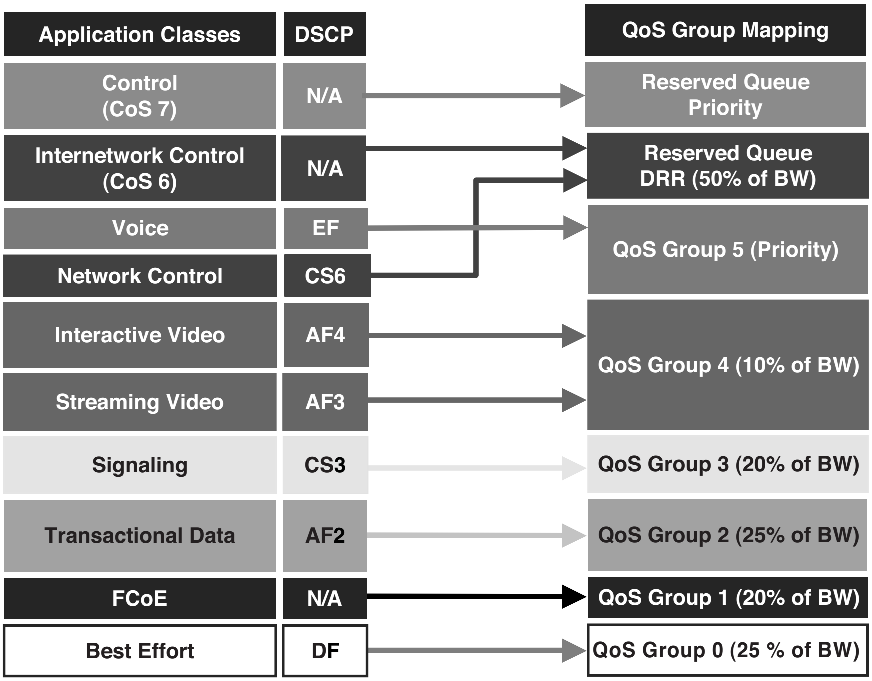


Figure 24-5 *Adapting the Eight-Class Model to the Nexus 5500*

As before, the eight-class model design is configured in four steps, as follows:

1. Configure the **qos** type class and policy maps.
2. Configure the **network-qos** type class and policy maps.
3. Configure the **queuing** type class and policy maps.
4. Apply all the service policies to the system QoS class.

Example 24-21 illustrates the configuration of the **qos** type class and policy maps. Note that in this example five class maps are defined because the FCoE and default classes are hard-coded in the Nexus operating system.

Example 24-21 *Configuring the* ***qos*** *Type Class and Policy Maps to Support the Eight-Class QoS Model*

```
 ! Configure the qos class maps
N5K(config-cmap-qos)# class-map type qos match-any VOICE
N5K(config-cmap-qos)# match dscp ef
 ! Match voice traffic with DSCP EF
N5K(config-cmap-qos)# class-map type qos match-any SIGNALING
N5K(config-cmap-qos)# match dscp cs3
 ! Match signaling traffic with DSCP CS3
```

```
N5K(config-cmap-qos)# class-map type qos match-any INTERACTIVE-VIDEO
N5K(config-cmap-qos)# match dscp af41 af42 af43
 ! Match interactive video traffic with DSCP AF4
N5K(config-cmap-qos)# class-map type qos match-any STREAMING-VIDEO
N5K(config-cmap-qos)# match dscp af31 af32 af33  ! Match streaming video traffic
with DSCP AF3
N5K(config-cmap-qos)# class-map type qos match-any TRANSACTINOAL-DATA
N5K(config-cmap-qos)# match dscp af21 af22 af23
 ! Match transactional data with DSCP AF2
 ! Configure the qos policy map
N5K(config)# policy-map type qos 8-CLASS-QOS-POLICY
N5K(config-pmap-qos)# class VOICE
N5K(config-pmap-c-qos)# set qos-group 5
 ! Map voice traffic to QoS group 5
N5K(config-pmap-c-qos)# class INTERACTIVE-VIDEO
N5K(config-pmap-c-qos)# set qos-group 4
 ! Map interactive video traffic to QoS group 4
N5K(config-pmap-c-qos)# class STREAMING-VIDEO
N5K(config-pmap-c-qos)# set qos-group 4
 ! Map streaming video to QoS group 4 (same as INTERACTIVE-VIDEO)
N5K(config-pmap-c-qos)# class SIGNALING
N5K(config-pmap-c-qos)# set qos-group 3
 ! Map signaling traffic to QoS group 3
N5K(config-pmap-c-qos)# class TRANSACTIONAL-DATA
N5K(config-pmap-c-qos)# set qos-group 2
 ! Map transactional data to QoS group 2
N5K(config-pmap-c-qos)# class class-fcoe
N5K(config-pmap-c-qos)# set qos-group 1
 ! Map FCoE to QoS group 1
```

Once the **qos** class and policy maps are configured, the next step is to configure the **network-qos** class and policy maps. Because there are only four customizable QoS groups to work with, the **network-qos** class map must consolidate two of these classes. This is most easily done by consolidating the Interactive Video and Streaming Video classes into one class called simply Video. Example 24-22 demonstrates this step.

Example 24-22 *Configuring the **network-qos** Type Class and Policy Maps to Support the Eight-Class QoS Model*

```
 ! Configure the network-qos class maps
N5K(config)# class-map type network-qos VOICE
N5K(config-cmap-nq)# match qos-group 5
 ! Matches traffic in QoS group 5
N5K(config)# class-map type network-qos VIDEO
N5K(config-cmap-nq)# match qos-group 4
```

```
 ! Matches traffic in QoS group 4
N5K(config)# class-map type network-qos SIGNALING
N5K(config-cmap-nq)# match qos-group 3
 ! Matches traffic in QoS group 3
! This is the consolidate class for the two video classes
N5K(config)# class-map type network-qos TRANSACTIONAL-DATA
N5K(config-cmap-nq)# match qos-group 2
 ! Matches traffic in QoS group 2
 ! Configure the network-qos policy map
N5K(config)# policy-map type network-qos 8-CLASS-NQ-POLICY
N5K(config-pmap-nq)# class type network-qos VOICE
 ! Activates the Voice network-qos class
N5K(config-pmap-nq-c)# class type network-qos SIGNALING
 ! Activates the Signaling network-qos class
N5K(config-pmap-nq-c)# class type network-qos VIDEO
 ! Activates the Video network-qos class
N5K(config-pmap-nq-c)# class type network-qos TRANSACTIONAL-DATA
 ! Activates the Transactional Data class
N5K(config-pmap-nq-c)# class type network-qos class-fcoe
N5K(config-pmap-nq-c)# pause no-drop
N5K(config-pmap-nq-c)# mtu 2158
 ! Activates the FCoE class and establishes the class as no-drop
```

After the **qos** and **network-qos** policies have been configured, the next step is to configure the queuing policies for each traffic class, as demonstrated in Example 24-23.

Example 24-23 *Configuring the* ***queuing*** *Type Class and Policy Maps to Support the Eight-Class QoS Model*

```
 ! Configure the queuing class maps
N5K(config-cmap-que)# class-map type queuing VOICE
N5K(config-cmap-que)# match qos-group 5
 ! Matches traffic in QoS group 5
N5K(config-cmap-que)# class-map type queuing VIDEO
N5K(config-cmap-que)# match qos-group 4
 ! Matches traffic in QoS group 4
N5K(config-cmap-que)# class-map type queuing SIGNALING
N5K(config-cmap-que)# match qos-group 3
 ! Matches traffic in QoS group 3
N5K(config-cmap-que)# class-map type queuing TRANSACTIONAL-DATA
N5K(config-cmap-que)# match qos-group 2
 ! Matches traffic in QoS group 2
 ! Configure the queuing policy map
N5K(config)# policy-map type queuing 8-CLASS-GLOBAL-QUEUING-POLICY
```

```
N5K(config-pmap-que)# class type queuing VOICE
N5K(config-pmap-c-que)# priority
 ! Gives voice the priority queue
N5K(config-pmap-c-que)# class type queuing SIGNALING
N5K(config-pmap-c-que)# bandwidth percent 10
 ! Signaling is given 10% of available BW
N5K(config-pmap-c-que)# class type queuing VIDEO
N5K(config-pmap-c-que)# bandwidth percent 25
 ! Video is given 25% of available BW
N5K(config-pmap-c-que)# class type queuing TRANSACTIONAL-DATA
N5K(config-pmap-c-que)# bandwidth percent 20
 ! Transactional Data is given 20% of available BW
N5K(config-pmap-c-que)# class type queuing class-fcoe
N5K(config-pmap-c-que)# bandwidth percent 20
 ! FCoE is given 20% of available BW
N5K(config-pmap-c-que)# class type queuing class-default
N5K(config-pmap-c-que)# bandwidth percent 25
 ! The default class is given 25% of available BW
```

The bandwidth settings here are somewhat arbitrary and are only shown as an example of what could be used. The actual bandwidth settings used in your network should reflect the expected traffic patterns in your data center.

As before, the final configuration step is to attach these three policy maps to the system, as demonstrated in Example 24-24. This activates the policies for all interfaces on the Nexus 5500.

Example 24-24 *Applying the QoS policies to the System Class*

```
N5K(config)# system qos
N5K(config -sys-qos)# service-policy type qos input 8-CLASS-QOS-POLICY
 ! Attach the qos policy to the system class
N5K(config -sys-qos)# service-policy type queuing output 8-CLASS-GLOBAL-QUEUING-
POLICY
 ! Attach the queuing policy in the output direction
N5K(config -sys-qos)# service-policy type network-qos 8-CLASS-NQ-POLICY
 ! Attaches the network-qos policy to the system
```

As before, you can verify this configuration with the following commands:

- **show policy-map type** [**qos** | **network-qos** | **queuing**]
- **show policy-map system**
- **show queuing interface ethernet** x/y

Additional QoS Designs Options

In addition to the ingress and egress queuing models already discussed, the Nexus 5500 supports several important other QoS design elements, including the following:

- Configuring QoS with the L3 card present
- Supporting QoS on the Nexus 2000 Fabric Extender
- Setting the MTU of different traffic types

Nexus 5500 L3 QoS Configuration

In its base configuration, the Nexus 5500 switch supports Layer 2 forwarding only. However, with the addition of the Layer 3 daughter card, the switch becomes capable of IP routing.

When the Layer 3 daughter card is used, there are some QoS design characteristics to be aware of. After an L2 frame is classified by the Nexus 5500, the QoS information is known and used by the switch for further QoS processing. However, if the frame is passed to the L3 daughter card, this information is not passed along. Similar to how the Nexus L2 UPC supports QoS, the L3 daughter card makes QoS decisions based on the QoS group label. Therefore, is it necessary that all traffic passed to the L3 daughter card have the QoS group value correctly marked by the **qos** type policy.

Figure 24-6 illustrates the traffic flow from the UPC to the L3 daughter card and back to another UPC on egress.

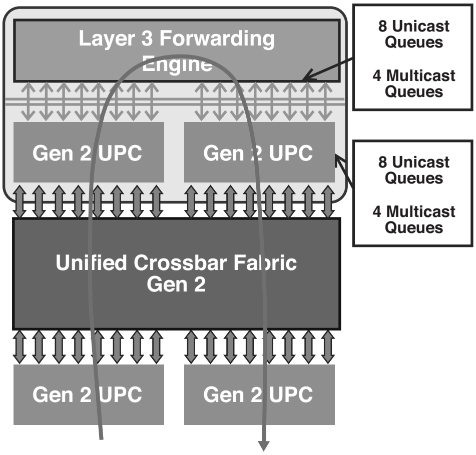


Figure 24-6 *Flow of Traffic Through the L3 Daughter Card*

As Figure 24-6 illustrates, the L3 forwarding engine connects to the Nexus 5500 as if it were actually connected by 16x 10GE ports. Congestion can be encountered both on egress from the UPC toward the L3 forwarding engine, and as traffic leaves the L3 engine back to the UPC. For example, if traffic is congested on egress toward the L3 engine, it is queued on the UPC according to the QoS group internal label of the frames.

To correctly design QoS with the L3 daughter card, you should adhere to the following recommendations:

- Apply the **qos** and **network-qos** type policies for classification to both the L3 and L2 interfaces in the traffic path (or just apply the service policies to the entire system).
- Apply the queuing type policy to the system in the egress direction (output).

Nexus 2000 Fabric Extender QoS

The Nexus 5500 supports Nexus 2000 series fabric extenders (known as FEX), providing greater scalability of server facing ports. The Nexus 2000 FEXs behave as remote linecards for a parent Cisco Nexus switch, including the Nexus 5500 and the Nexus 7000. Together, the FEX and the parent switch form a distributed modular switching system. The FEX is entirely controlled by the parent switch where all configuration and software control is done.

The FEX has two types of interface:

- **FEX host interface (HIF):** Server-facing ports
- **FEX network interface (NIF):** Ports on the FEX that connect to the upstream parent switch

Figure 24-7 illustrates the QoS mechanism of the FEX when connected to the Nexus 5500.

As illustrated in Figure 24-7, the FEX queuing and buffering system works in the following way:

1. Incoming traffic is classified based on CoS.
2. Traffic is queued and scheduled at the egress NIF.
3. The Nexus 5500 ingress port (ingress UPC) classifies traffic and does buffer allocation, an MTU check, and CoS marking.
4. Traffic is queued and scheduled on the Nexus 5500 egress interface (egress UPC).
5. The FEX does CoS-based classification on the ingress NIF port.
6. The FEX queues schedules traffic on the egress HIF ports; the HIF does egress tail drop for each port.

An important point to understand about the Nexus 5500 to FEX communication is that all traffic is part of an 802.1Q trunk. This means that queuing between the 5500 and the FEX (in both directions) relies on CoS. This highlights the importance of correct CoS marking that must be done in the **network-qos** policy map.

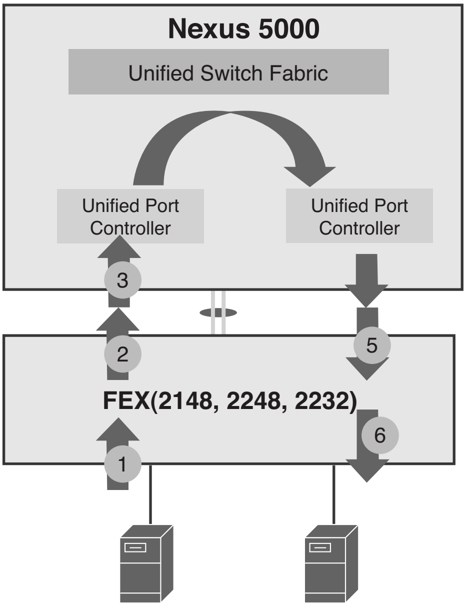


Figure 24-7 *Nexus 5500/2000 FEX QoS Architecture*

Unlike the shared ingress buffer model used for normal Nexus 5500 ports, the FEX operates as a linecard, meaning that it does egress queuing on the linecard and is not part of the shared ingress buffering system on the Nexus 5500. Because FEXs operate as egress-based queuing devices, they behave differently than the Nexus 5500.

Each FEX has local port buffers that can be adjusted to prevent one blocked receiver from affecting traffic that is sent to other noncongested receivers. As well, adjusting to a higher queue limit provides better burst absorption, but gives less HoL blocking protection.

Example 24-25 shows how to tune the FEX queue limits.

Example 24-25 *Tuning the FEX Buffers*

```
N5K(config)# system qos
N5K(config-sys-qos)# no fex queue-limit
 ! Disabling the per-port tail-drop threshold
N5K(config)# fex 100
N5K(config-fex)# hardware N2248T queue-limit 356000
N5K(config-fex)# hardware N2248T queue-limit ?
  <CR>
  <2560-652800>  Queue limit in bytes
! Changing the FEX queue limit on an HIF port
```

You can verify the configuration in Example 24-25 with the following commands:

- **show fex detail**
- **show queuing interface ethernet fex/x/y**

One interesting aspect of the FEX queuing architecture is that some FEX models support a shared memory buffer across the entire FEX, whereas other FEX models support buffering on groups of four or eight ports.

Table 24-4 shows a comparison of the different FEX buffer capabilities.

Table 24-4 *Cisco Nexus 2000 FEX Buffer Comparison*

FEX Model	Interfaces	Buffer Capacity
N2K-C2248TP-E	48 × 100/1GE RJ-45	32 MB shared across all ports
N2K-C2232TM-E	32 × 1G/10G RJ-45	1280 KB per 8 ports
N2K-C2232PP	32 × 1G/10G SFP+	1280 KB per 8 ports
N2K-C2232TM	32 × 1G/10G RJ-45	1280 KB per 8 ports

Consider an example where the Nexus 2248TP-E is attached to a Nexus 5548 switch. In this example, there is a 10GE network-attached server (NAS) in the data center that is sending large bursts of data toward a 1GE server. (Hadoop is another technology that generates bursty traffic where this example might apply.) Figure 24-8 illustrates this situation.

Because the 2248TP-E FEX model uses a 32-MB shared memory buffer architecture, it is well suited to handling large traffic bursts. The speed mismatch between the NAS and the server (10GE versus 1GE) has the obvious potential of causing congestion on the egress server-facing ports on the FEX. Because the 2248TP-E shares the 32 MB of buffering across the entire FEX, this speed mismatch has the potential to consume the entire switch buffer by just serving one egress port! This congested port can thus negatively impact all other noncongested ports because the entire shared memory is consumed. If the queue limit value on the FEX is reduced, however, this HoL blocking scenario can be easily managed.

Example 24-26 illustrates how to reduce the queue limit on a Nexus 2248TP-E FEX.

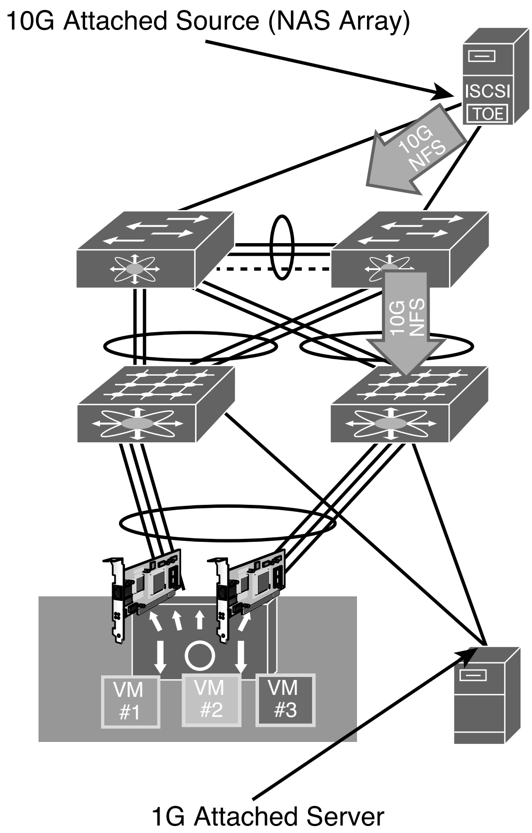


Figure 24-8 *Tuning the Nexus 2248TP-E Buffer for Bursty NAS Traffic*

Example 24-26 *Tuning the Nexus 2248TP-E Buffers*

```
N5K(config)# fex 110
N5K(config-fex)# hardware N2248TPE queue-limit 4000000 rx
N5K(config-fex)# hardware N2248TPE queue-limit 4000000 tx
N5K(config)# interface e110/1/1
N5K(config-if)# hardware N2348TP queue-limit 4096000 tx
! Reduce the queue limit for the server speaking to the NAS
```

You can verify the configuration in Example 24-26 with the following commands:

- **show fex detail**
- **show queuing interface ethernet fex/x/y**

Using the network-qos Policy to Set MTU

One of the key differences between IOS-based MQC and NX-OS is that in NX-OS interface-level features are configured in policy maps. One example of this is setting the MTU. Instead of setting the MTU on an interface, NX-OS sets it by using the **network-qos** type policy and attaching it to the system target. For example, when the **feature fcoe** command is executed (thus enabling FCoE), a network policy giving FCoE an MTU of 2158 bytes needs to be configured to allow the correct transmission of FCoE traffic.

In Example 24-27, the MTU for the default class is set to support jumbo frames, where the MTU needs to be set to 9216 bytes.

In this example, note that no class map is defined. This is because the policy is referencing the default class, which matches all unclassified traffic.

Example 24-27 *Using a Policy Map to Set the MTU*

```
N5K(config)# policy-map type network-qos SET-MTU
N5K(config-pmap-nq)# class type network-qos class-default
N5K(config-pmap-nq-c)# mtu 9216
 ! Sets the MTU for the default class
N5K(config)# system qos
N5K(config-sys-qos)# service-policy type network-qos SET-MTU
 ! Attaches the policy to the system QoS class
```

You can verify the configuration in Example 24-27 with the **show policy-map type network-qos** command.

Summary

This chapter detailed best-practice recommendations for QoS design on the Nexus 5500 data center switch in the role of a Layer 2 server access, aggregation, and even a Layer 3 distribution layer switch.

This chapter began by reviewing the Nexus 5500 system architecture, especially the VOQ architecture. This mechanism is used to buffer packets on ingress with a shared memory model and virtual output queuing. Because the Nexus 5500 uses internal CoS values for classification of traffic, it was shown how the eight CoS values are used for VOQ in the system.

Next, the ingress QoS models were discussed, including various trust and marking schemes. To apply these QoS policies to the system, an analysis of the three types of class and policy maps was presented.

Following this, the egress queuing models were discussed, in addition to how to configure the egress queues on this platform. The four-class and eight-class QoS models were presented for this platform. Because the Nexus 5500 reserves two CoS values (6 and 7), it was demonstrated how to adapt the eight-class QoS Model to the four customizable QoS classes while still supporting the FCoE class. Because the Nexus 5500 supports only 4 (or 5 if not using FCoE) customizable QoS classes, the 12-class QoS model is not supported on this platform.

Finally, several other additional QoS options were presented, including how to design and configure the L3 daughter card, Nexus 2000 FEXs, and how to adjust the MTU of a specific class.

Additional Reading

Nexus 5500 QoS Configuration Guide, NX-OS Version 6.0(2)N1(2): http://www.cisco.com/en/US/docs/switches/datacenter/nexus5500/sw/qos/602_N1_2/b_5500_QoS_Config_602N12.html

Nexus 5500 L3 daughter card configuration: http://www.cisco.com/en/US/docs/switches/datacenter/nexus5500/sw/unicast/602_N1_2/cisco_n5500_layer3_ucast_cfg_rel_602_N1_2.html

NX-OS and Cisco Nexus Switching: Next Generation Data Center Architectures, Second Edition, by Fuller, Jansen, McPherson (Cisco Press, 2013)

Data Center Configuration Files—Smart Business Architecture: http://www.cisco.com/en/US/docs/solutions/SBA/February2013/Cisco_SBA_DC_DataCenterConfigurationFilesGuide-Feb2013.pdf

Chapter 25

Data Center Core (Nexus 7000) QoS Design

The primary quality of service (QoS) role of the data center core switch is to manage packet loss. Class of service (CoS) or differentiated services code point (DSCP) markings have already been set at the access layer or the virtual access layer of the data center, and therefore explicit classification, marking, and policing policies are usually not required at the core layer. So, the data center core switch should trust CoS and DSCP on ingress (which are NX-OS defaults) and perform ingress and egress queuing, as shown in Figure 25-1.

The Cisco Nexus 7000 has been engineered to fit the role of a data center core switch and is featured in this design chapter.

Many data center design options exist, but this chapter assumes the use of Nexus M2 modules (which are discussed in the following sections) in the Layer 2/Layer 3 core for routed interconnects, campus/WAN interconnects, and Overlay Transport Virtualization (OTV) edges—and some additional design options for direct server connections.

F2/F2e modules (discussed later in this chapter) are assumed to extend the data center bridging (DCB) fabric to the core layer by providing the interconnections to the Nexus 5500s at the access layer.

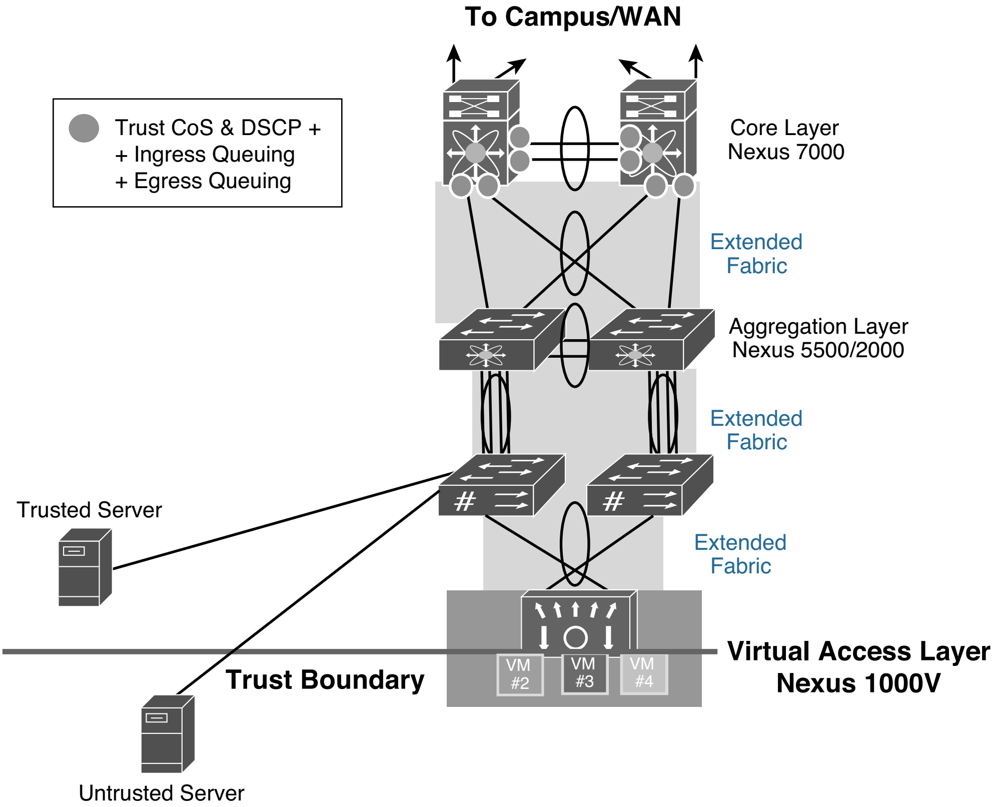


Figure 25-1 *Data Center Core Switch Port QoS Roles*

Nexus 7000 Overview

The Cisco Nexus 7000 series data center switches support more than 17 terabits per second (Tbps) of aggregate switching capacity, making them highly capable and effective in the role of data center core switches. The Nexus 7000 series switches are modular and are available in 4-, 9-, 10-, and 18-slot chassis. These chassis can be outfitted with the following types of modules:

- **Supervisor modules:** Supervisor modules provide scalable control plane and management functions for the switch. The Supervisor controls the Layer 2 and 3 services, redundancy capabilities, configuration management, status monitoring, power and environmental management, and transparent upgrades to I/O and fabric modules.
- **Fabric modules:** Fabric modules provide the central switching element for fully distributed forwarding on the I/O modules. The addition of each Fabric Module increases the bandwidth to all module slots up to the system limit of five modules, with Cisco Nexus 7000 Series Fabric-2 modules delivering up to 550 Gbps per slot.

- **M2-Series I/O modules:** M2-Series modules are full-featured, high-performance modules with support for high-density 10-, 40-, and 100-Gigabit Ethernet interfaces. Larger forwarding tables can be enabled to support a full IPv4 and IPv6 Internet route table. Furthermore, M2-Series supports a comprehensive feature set including Cisco Overlay Transport Virtualization (OTV), Cisco Locator/ID Separation Protocol (LISP), Multiprotocol Label Switching (MPLS), full Cisco NetFlow, and Virtual Private LAN Services (VPLS). In addition, these modules maintain architectural consistency with the Cisco Catalyst 6500 series switches.
- **F2/F2e-Series I/O modules:** F2-Series modules are built with switch-on-a-chip (SoC) architecture, which increases performance while reducing the power use, latency, and cooling requirements of the module. Each port can be used at 1- or 10-Gigabit Ethernet speeds. The F2-Series modules support Layer 3 functions and unified fabric capabilities such as Cisco FabricPath, Fibre Channel over Ethernet (FCoE), and integration with Cisco Nexus 2000 Series Fabric Extenders (FEX). In addition, enhanced F2e modules help to enable full Layer 2 and Layer 3 functionality and interoperability with M2-Series modules in the same virtual device context (VDC).

Figure 25-2 illustrates common software features—and differences—between M2-Series and F2-Series modules.

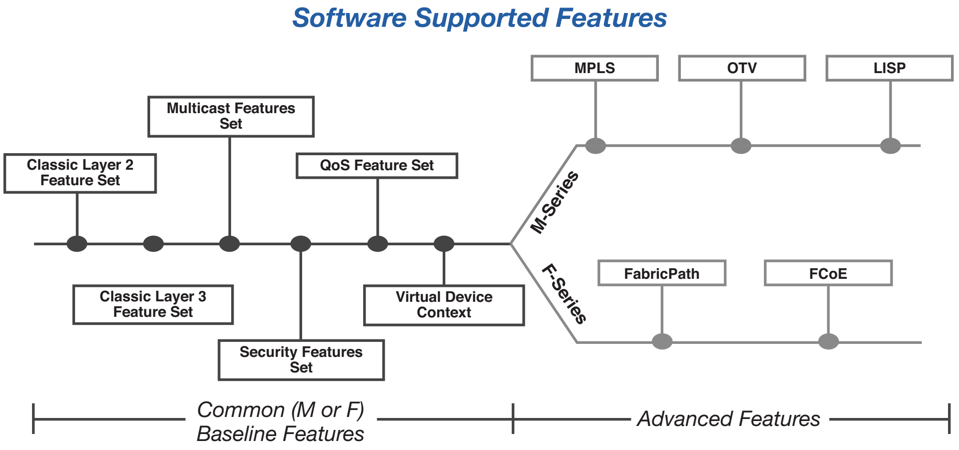


Figure 25-2 *Cisco Nexus 7000 M2 and F2 Series Module Comparisons*

Although both M2- and F2-Series modules support QoS features, there are significant differences in architecture and capabilities between these module families to merit discussing QoS design recommendations for these separately within this chapter, following a brief review of QoS features and functionality common to both module families.

As with previously discussed Nexus switches, the Nexus 7000 supports the following types of QoS policies:

- **qos:** Defines Modular QoS command-line interface (MQC) objects used for marking and policing.
- **network-qos (required on F-Series modules only):** Defines the characteristics of network-wide QoS properties (such as used in DCB networks) and should be applied consistently on all switches participating in the network.
- **queuing:** Defines MQC objects used for queuing and scheduling and a limited set of the marking objects.
- **control plane:** Defines MQC objects used for control plane policing (CoPP).

Table 25-1 outlines the order in which QoS policies are applied on the Nexus 7000 modules in the ingress and egress direction.

Table 25-1 *Cisco Nexus 7000 QoS Action Order of Operation by Traffic Direction*

Sequencing of Ingress QoS Actions	Sequencing of Egress QoS Actions
1. Queuing and scheduling	1. Classification
2. Mutation	2. Marking
3. Classification	3. Policing
4. Marking	4. Mutation
5. Policing	5. Queuing and scheduling

The Nexus 7000 trusts traffic by default, according to the rules summarized in Table 25-2.

Table 25-2 *Cisco Nexus 7000 Default Trust States*

Trust DSCP/CoS by Default	Ingress	Egress (After Traffic Is Routed)[1]
Switch virtual interface (SVI)	CoS	DSCP
Routed interface	DSCP	DSCP
Layer 2 interface	CoS[2]	DSCP

[1] When traffic is routed, the DSCP value is used (by default) to derive the egress queue. If the egress interface is a trunk, the CoS is derived from the DSCP value of the routed packet.

[2] When the Layer 2 Interface is an access port, it is considered as no CoS. Also, CoS is set to 0 in the case of a trunk interface with bridged traffic, even if DSCP bits are set.

The key takeaways relating to Nexus 7000 trust default behavior are these:

- For (L2) *bridged* traffic, CoS is used for ingress queue selection; DSCPs—while received and transmitted unmodified—are ignored from a QoS perspective; received CoS markings are used for egress queue selection and are also transmitted unmodified on egress bridged ports.
- For (L3) *routed* traffic, CoS is used for ingress queue selection; DSCP markings are not modified, but are used to generate new CoS markings derived from the 3 Most Significant Bits (MSB) of the DSCP; these newly generated CoS markings are then used for egress queue selection *even if the egress interface is not an 802.1Q trunk and so not carrying CoS within the frame.*

Of course, these default behaviors can be modified with explicit configuration policies.

For example, perhaps the bridged traffic already has DSCP values marked at the IP layer and you want to use these DSCP markings to generate egress CoS markings. You can do so via the policy shown in Example 25-1.

Example 25-1 *Nexus 7000 Table Map Policy to Drive DSCP-based CoS on Egress for Bridged Traffic*

```
 ! This section configures a DSCP-to-CoS table map
N7K(config)# table-map DSCP-TO-COS-BRIDGED
N7K(config-tmap)# default copy
 ! By default the 3 MSBs of the DSCP will be copied to CoS

 ! This section configures the policy map to apply the table-map
N7K(config-tmap)# policy-map type qos DSCP-TO-COS-BRIDGED-PMAP
N7K(config-pmap-qos)# class class-default
N7K(config-pmap-c-qos)# set dscp dscp table DSCP-TO-COS-BRIDGED
 ! All traffic will have the DSCP-to-CoS table map applied

 ! This section applies the policy map to an Ethernet interface
N7K(config)# interface Ethernet 2/10
N7K(config-if)# service-policy type qos input DSCP-TO-COS-BRIDGED-PMAP

! This section applies the policy map to a VLAN interface
N7K(config-if)# vlan configuration 110
N7K(config-vlan-config)# service-policy type qos input DSCP-TO-COS-BRIDGED-PMAP
```

Note When you use a **type qos** policy to **set dscp** (either directly to explicit value or indirectly via a table map, as shown in Example 25-1), the following will apply:

- The **set dscp** value will set the DSCP.

- The **set dscp** statement will also drive CoS re-marking, such that a new CoS value is generated for the packet with its value derived from the 3 MSB of the new DSCP; this newly generated CoS markings is used for egress queue selection *even if the egress interface is not an 802.1Q trunk and so not carrying CoS within the frame.*
- This is true for both (L2) bridged and (L3) routed traffic.

Note Type QoS policies can be applied to physical interfaces (or interface ranges) or to logical interfaces, such as VLANs—both of which are shown in Example 25-1.

You can verify the configuration in Example 25-1 with the following commands:

- **show table-map**
- **show policy-map**
- **show policy-map interface**

Nexus 7000 M2 Modules: Architecture and QoS Design

The hardware architecture of the Nexus 7000 switch with M2-Series modules is shown in Figure 25-3, highlighting the points at which QoS policies are applied. As shown in this figure, the M2 architecture features a hybrid mix of physical port queues combined with virtual output queues.

QoS policies are applied at the following points on M2 modules:

1. Ingress (**type queuing**) policies are applied at the ingress physical port.
2. Classification, marking and policing (**type qos**) policies are implemented within the forwarding engine.
3. Fabric QoS policies are applied in the VOQs *before* the packet traverses the three-stage switching fabric.
4. Egress (**type queuing**) policies are applied in the VOQs *after* the packet traverses the three-stage switching fabric.
5. Egress (**type queuing**) policies are applied again in the egress physical port.

Ingress and egress queuing policies may either be default or explicitly defined, as may **type qos** policies. In contrast, fabric QoS policies are nonconfigurable.

Fabric QoS policies determine the scheduling of the ingress virtual output queues (VOQs) because these contend for access to the three-stage switch fabric. Arbitration of fabric QoS scheduling is controlled by the central arbiter on the Supervisor (as shown in

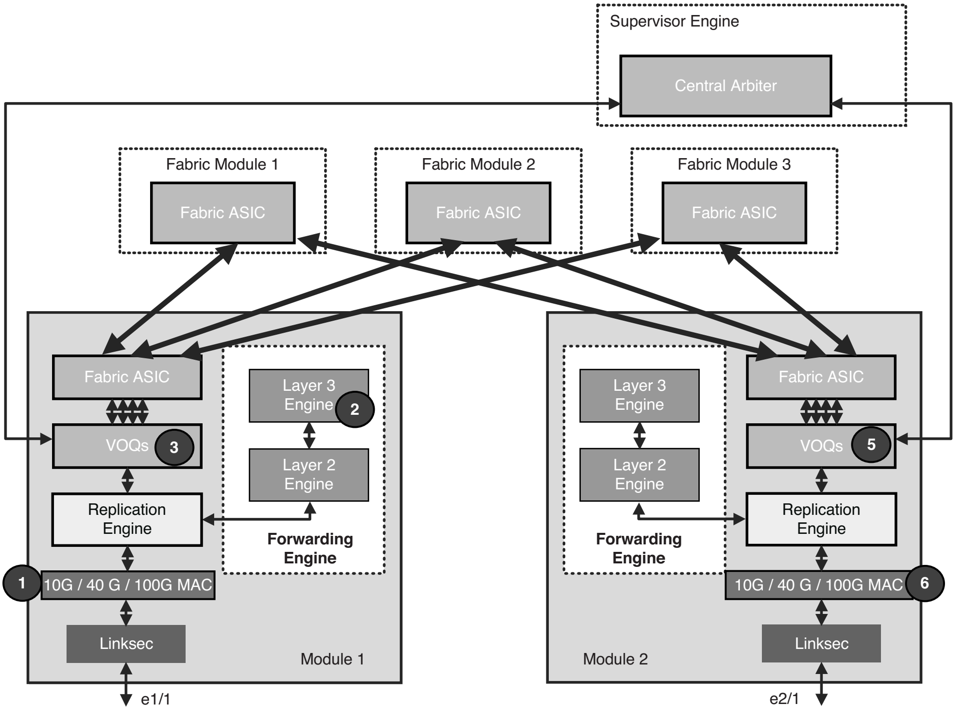


Figure 25-3 *Cisco Nexus 7000 M2 Module Hardware Architecture and QoS Enforcement Points*

Figure 25-3). The central arbiter grants permission to the ingress VOQs to access the fabric by considering two factors:

- Virtual queuing indexes (VQIs) of the egress module
- Traffic priority (CoS)

A VQI is a logical destination address on the egress module (and is module dependent). Each VQI represents 10- to 12-Gbps of traffic flow, and therefore would correspond to the requirements of the egress interface: A 10-GE interface would be assigned a single VQI; a 40 GE interface would be assigned 4 VQIs; a 100 GE interface would be assigned 10 VQIs. The central arbiter thus considers the bandwidth capacity of the egress module in its scheduling decision such that egress VOQs and egress physical port queues are not overwhelmed.

In addition, the central fabric arbiter considers the traffic priority (CoS) values of the packets in the VOQs. For M2 modules, the COS-to-queue mapping for the fabric VOQs is controlled separately from the port queuing (that is, port-queuing policies may be configured differently than these nonconfigurable fabric queuing policies). Table 25-3 shows the CoS-to-fabric VOQ mappings for M1/M2 modules.

Table 25-3 *Nexus 7000 M1/M2 Fabric VOQ CoS Mappings*

Fabric VOQ	CoS Values
PQ1	5-7
Q2	3-4
Q3	2
Q4	0-1

Q1 (PQ1) is scheduled on a strict-priority basis; Q2–Q4 are scheduled with deficit weighted round-robin (DWRR) and are assigned equal weights. The mapping and weights are used for fabric VOQs are static, and there is no configuration associated with this behavior (at the time of this writing). However, it is beneficial for administrators to be aware of these internal fabric queuing mechanisms when considering their queuing designs.

It might seem redundant to have queuing policies repeated several times in the packet flow. However, each stage of queuing (whether physical or virtual) has a separate function, as shown in Figure 25-4.

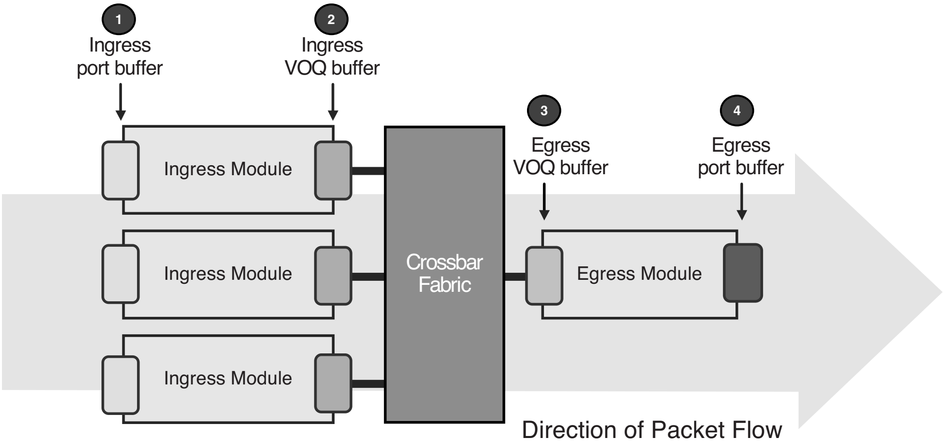


Figure 25-4 *Cisco Nexus 7000 M2 Modules: Physical and Virtual Queuing Buffers*

As shown in Figure 25-4, the various M2 queuing buffers and their corresponding functions are as follows:

1. **Ingress port buffer:** Manages congestion for ingress forwarding/replication engines
2. **Ingress VOQ buffer (fabric queue):** Manages congestion toward the switching fabric
3. **Egress VOQ buffer:** Receives frames from the fabric and manages congestion for multidestination frames

4. **Egress port buffer:** Manages congestion at the egress interface

Having considered the architecture of the Nexus 7000 with M2 modules, let's take a look at how QoS performed within it.

M2 QoS Design Steps

As shown in Figure 25-1, the three primary QoS requirements of a data center core switch are to

- Trust CoS and DSCP on ingress
- Queue on ingress
- Queue on egress

Because the default behavior of the Nexus 7000 is to trust CoS and DSCP on ingress, there are effectively only two steps to configuring QoS on a Nexus 7000 with M2-Series modules in the role of a data center core switch:

1. Configure ingress queuing.
2. Configure egress queuing.

In addition, M2 modules may be used as OTV edges, which presents another design option that is discussed later in this chapter.

M2 Queuing Models

Each Nexus 7000 M2-Series module supports an 8Q2T ingress queuing structure and a 1P7Q4T egress queuing structure. At the time of this writing, both queuing structures use CoS-to-Queue mappings; therefore, no more than eight queuing classes are supported (and therefore only the four-class and eight-class strategic enterprise QoS models are presented in the following sections).

Note 10GE M1 modules also support 8Q2T ingress and 1P7Q4T egress queuing models, and therefore the design recommendations presented in the following sections directly apply to these modules also.

The Nexus 7000 uses predefined queuing class map names, as summarized in Table 25-4 (for the 8Q2T ingress queuing model) and Table 25-5 (for the 1P7Q4T egress queuing model).

Table 25-4 *Nexus 7000 M2-Series Modules Predefined Queuing Class Map Names and Descriptions: 8Q2T Ingress Model*

8Q2T (8 Nonpriority Queues with 2 Thresholds per Queue)	
Queuing Class Map Name	*Queue Description*
8q2t-in-q1	Ingress queue 1 of 8q2t type
8q2t-in-q2	Ingress queue 2 of 8q2t type
8q2t-in-q3	Ingress queue 3 of 8q2t type
8q2t-in-q4	Ingress queue 4 of 8q2t type
8q2t-in-q5	Ingress queue 5 of 8q2t type
8q2t-in-q6	Ingress queue 6 of 8q2t type
8q2t-in-q7	Ingress queue 7 of 8q2t type
8q2t-in-q-default	Ingress default queue of 8q2t type

Table 25-5 *Nexus 7000 M2-Series Modules Predefined Queuing Class Map Names and Descriptions: 1P7Q4T Egress Model*

1P7Q4T (1 Strict-Priority Queue and 7 Nonpriority Queues with 4 Thresholds per Queue)	
Queuing Class Map Name	*Queue Description*
1p7q4t-out-pq1	Egress priority queue of 1p7q4t type
1p7q4t-out-q2	Egress queue 2 of 1p7q4t type
1p7q4t-out-q3	Egress queue 3 of 1p7q4t type
1p7q4t-out-q4	Egress queue 4 of 1p7q4t type
1p7q4t-out-q5	Egress queue 5 of 1p7q4t type
1p7q4t-out-q6	Egress queue 6 of 1p7q4t type
1p7q4t-out-q7	Egress queue 7 of 1p7q4t type
1p7q4t-out-q-default	Egress default queue of 1p7q4t type

M2 Default Queuing Models

Queuing features are enabled by default on the Nexus 7000 via a system-defined queuing policy map applied on each port and port channel. The default 8Q2T ingress queuing model and the default 1P7Q4T egress queuing model are illustrated in the combined Figure 25-5.

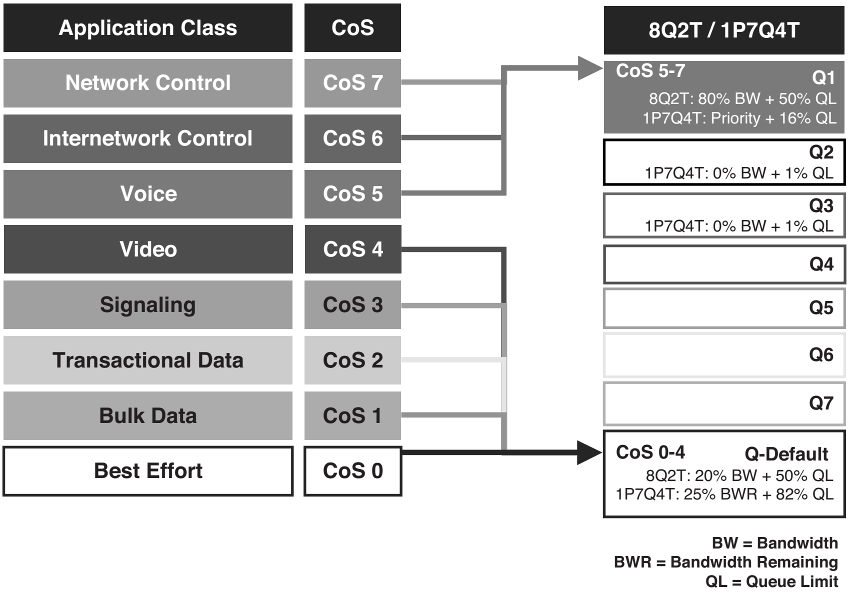


Figure 25-5 *Cisco Nexus 7000 M2 Default (8Q2T Ingress / 1P7Q4T Egress) Queuing Model*

You can verify this default policy by running a **show policy-map interface** command on an interface that has no policies attached to it, as shown in Example 25-2.

Example 25-2 *Verifying Default System Queuing Policies:* ***show policy-map interface***

```
N7K# show policy-map interface ethernet 1/2
Global statistics status :   enabled
Ethernet1/24
  Service-policy (queuing) input:   default-in-policy
    policy statistics status:   enabled (current status: enabled)
    Class-map (queuing):   in-q1 (match-any)
      queue-limit percent 50
      bandwidth percent 80
      queue dropped pkts : 0
    Class-map (queuing):   in-q-default (match-any)
      queue-limit percent 50
      bandwidth percent 20
      queue dropped pkts : 0
  Service-policy (queuing) output:   default-out-policy
    policy statistics status:   enabled (current status: enabled)
    Class-map (queuing):   out-pq1 (match-any)
      priority level 1
      queue-limit percent 16
      queue dropped pkts : 0
    Class-map (queuing):   out-q2 (match-any)
      queue-limit percent 1
      queue dropped pkts : 0
    Class-map (queuing):   out-q3 (match-any)
      queue-limit percent 1
      queue dropped pkts : 0
    Class-map (queuing):   out-q-default (match-any)
      queue-limit percent 82
      bandwidth remaining percent 25
      queue dropped pkts : 0
N7K#
```

Any explicitly defined queuing policy (such as the ones detailed in the following sections) applied to an interface will—of course—override this default system-defined queuing policy.

Queuing policies can be modified by changing the CoS values matched by the predefined queuing class maps, and the policy map actions per queuing class, as shown in the following sections.

Note Any modified queuing policies that assign CoS values to ingress or egress queues 2 through 7 should be sure to likewise assign adequate buffers to these queues (via the **queue-limit** policy map action command); otherwise, traffic may tail drop because of the lack of buffers. Remember that by default buffer allocations on these queues are 0 percent (except for egress queue 2 and 3, which are set to only 1 percent).

Note Changing CoS-to-queue assignments in the system-specified queuing class maps should be performed on the primary/administrative VDC, because this will affect CoS-to-queue assignments on all ports (in any/all VDCs) of a given queuing model type.

Note It is critically important to note that changing queuing policies is disruptive on the Nexus 7000 and therefore should be done during scheduled downtime on production networks.

M2 Four-Class (4Q2T Ingress / 1P3Q4T Egress) Queuing Model

In the four-class model (illustrated in Figure 11-3, but adapted to the data center), the application class to queue mappings are as follows:

- Real-time traffic (marked EF, mapped by default to CoS 5) is allocated 25 percent bandwidth in the ingress 4Q2T model, but is enabled with strict-priority queuing in the egress 1P3Q4T model. In addition, CoS values 6 and 7 will be left mapped to the real-time queue to protect data center network control traffic.

Note By default, CoS 6 is assigned to the PQ (as shown in Figure 25-5) and therefore provides QoS for internetwork control traffic within the data center. Be careful if you create a policy assigning *only* CoS 5 to the PQ, because this will have the undesired effect of reassigning CoS 6 to the default class. In such a case, it is recommended that CoS 6 be explicitly mapped to a (preferably dedicated) nondefault queue. Nexus 7000 Supervisors mark internnetwork control protocols—such as Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), Protocol-Independent Multicast (PIM), Label Distribution Protocol (LDP), and Overlay Transport Virtualization (OTV)—to CS6 (which translates to CoS 6). In addition, Nexus 1000V platforms mark their control plane traffic—and virtual machine (VM) control traffic—to CS6 by default.

Note By default, CoS 7 is also assigned to the PQ. This CoS marking is sometimes used for various network or VM control plane protocols in the data center. Therefore, it is recommended to leave CoS 7 mapped to the PQ to support these protocols.

- Signaling traffic (marked CS3, mapped by default to CoS 3) is assigned to a dedicated nonpriority queue with a 25 percent bandwidth allocation in the Ingress 4Q2T model and a 35 percent bandwidth-remaining allocation in the egress 1P3Q4T model.

Note FCoE traffic is not present in this model because M2 modules do not support FCoE.

Note It is recommended to use M-Series modules to connect to Cisco CallManagers/ Cisco Unified Communications Managers (CUCM) in data center environments where FCoE is present. In this manner, F-Series cards will have CoS 3 dedicated for servicing FCoE traffic with a no-drop service (as discussed later in this chapter).

- Transactional data traffic (marked AF2, mapped by default to CoS 2) is assigned to another dedicated nonpriority queue with a 25 percent bandwidth allocation in the ingress 4Q2T model and a 35 percent bandwidth-remaining allocation in the egress 1P3Q4T model.

- Best-effort traffic (marked DF, mapped by default to CoS 0) is assigned to the default queue with a 25 percent bandwidth allocation in the ingress 4Q2T model and a 30 percent bandwidth-remaining allocation in the egress 1P3Q4T model.

All bandwidth allocations will vary according to data center traffic profiles; for the sake of simplicity, (roughly) equal values have been allocated to these classes (except for the PQ and class default on egress).

Queue limits (buffer allocations) are assigned proportionally to bandwidth in the ingress queuing model, but are assigned inversely proportionally to the PQ/default queue in the egress queuing model. This is following the logic that the PQ is serviced in real time, and therefore it needs fewer buffers (whereas the default-queue—which has the least bandwidth allocated to it—can benefit from deeper buffers to prevent drops).

To simplify the design, (CoS-based) weighted random early detection (WRED) is not enabled in this model, but is enabled on the following one (eight-class 8Q2T/1P7Q4T model).

These four class mappings, along with bandwidth and buffer allocations, are illustrated in Figure 25-6.

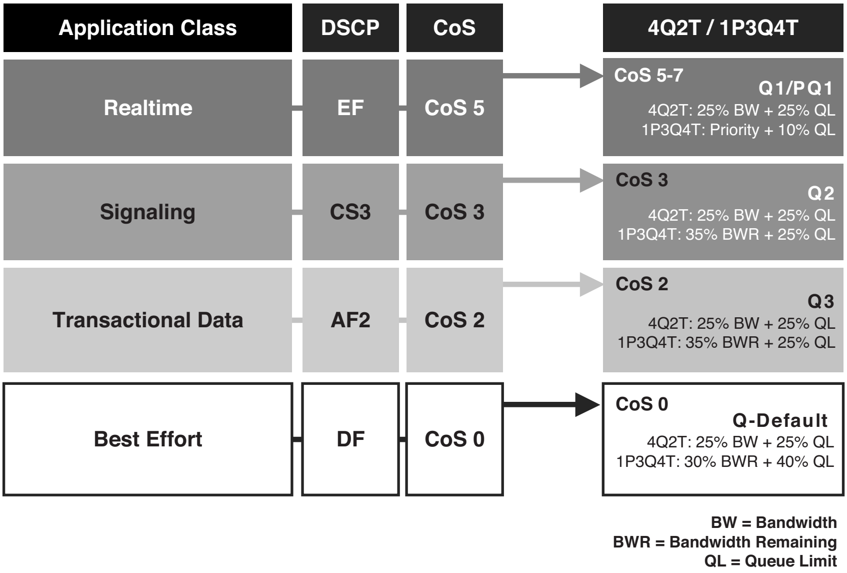


Figure 25-6 *Cisco Nexus 7000 M2 Four-Class (4Q2T Ingress / 1P3Q4T Egress) Queuing Model*

Example 25-3 shows the corresponding configuration for four-class (4Q2T ingress / 1P3Q4T egress) queuing.

Example 25-3 *Nexus 7000 M2 Four-Class (4Q2T Ingress / 1P3Q4T Egress) Queuing Model Examples*

```
! This section configures the ingress queuing class maps
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q1
N7K(config-cmap-que)#  match cos 5-7
 ! Realtime traffic (DSCP EF/CoS 5) is mapped to ingress Q1
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q2
N7K(config-cmap-que)#  match cos 3
 ! Signaling (DSCP CS3/CoS3) is mapped to ingress Q2
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q3
N7K(config-cmap-que)#  match cos 2
 ! Transactional Data (DSCP AF2/CoS2) is mapped to ingress Q3

 ! This section configures the egress queuing class maps
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-pq1
N7K(config-cmap-que)#  match cos 5-7
 ! Realtime traffic (DSCP EF/CoS 5) is mapped to egress PQ
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q2
N7K(config-cmap-que)#  match cos 3
 ! Signaling (DSCP CS3/CoS3) is mapped to egress Q2
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q3
N7K(config-cmap-que)#  match cos 2
 ! Transactional Data (DSCP AF2/CoS2) is mapped to egress Q3

 ! This section configures the 4Q2T ingress queuing policy map
N7K(config-cmap-que)# policy-map type queuing 4Q2T-INGRESS
N7K(config-pmap-que)#  class type queuing 8q2t-in-q1
N7K(config-pmap-c-que)#   bandwidth percent 25
N7K(config-pmap-c-que)#   queue-limit percent 25
 ! Ingress Q1 (Realtime) is allocated 25% bandwidth and 25% buffers
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q2
N7K(config-pmap-c-que)#   bandwidth percent 25
N7K(config-pmap-c-que)#   queue-limit percent 25
 ! Ingress Q2 (Signaling) is allocated 25% bandwidth and 25% buffers
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q3
N7K(config-pmap-c-que)#   bandwidth percent 25
N7K(config-pmap-c-que)#   queue-limit percent 25
 ! Ingress Q3 (Trans-Data) is allocated 25% bandwidth and 25% buffers
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q-default
N7K(config-pmap-c-que)#   bandwidth percent 25
N7K(config-pmap-c-que)#   queue-limit percent 25
 ! Ingress default queue is allocated 25% bandwidth and 25% buffers

 ! This section configures the 1P3Q4T egress queuing policy map
```

```
N7K(config-cmap-que)# policy-map type queuing 1P3Q4T-EGRESS
N7K(config-pmap-que)#  class type queuing 1p7q4t-out-pq1
N7K(config-pmap-c-que)#   priority
 ! Strict priority is enabled on egress PQ1
N7K(config-pmap-c-que)#   queue-limit percent 10
 ! Egress PQ (Realtime) is allocated 10% of buffers
N7K(config-pmap-c-que)#  class type queuing 1p7q4t-out-q2
N7K(config-pmap-c-que)#   bandwidth remaining percent 35
N7K(config-pmap-c-que)#   queue-limit percent 25
 ! Egress Q2 (Signaling) is allocated 35% BWR and 25% buffers
N7K(config-pmap-c-que)#  class type queuing 1p7q4t-out-q3
N7K(config-pmap-c-que)#   bandwidth remaining percent 35
N7K(config-pmap-c-que)#   queue-limit percent 25
 ! Egress Q3 (Trans-Data) is allocated 35% BWR and 25% buffers
N7K(config-pmap-c-que)#  class type queuing 1p7q4t-out-q-default
N7K(config-pmap-c-que)#   bandwidth remaining percent 30
N7K(config-pmap-c-que)#   queue-limit percent 40
 ! Egress default queue is allocated 30% BWR and 40% buffers

 ! This section applies the queuing policies to the interface(s)
N7K(config)# interface Ethernet 1/20
N7K(config-if)# service-policy type queuing input 4Q2T-INGRESS
 ! 4Q2T ingress queuing policy is attached to the interface (s)
N7K(config-if)# service-policy type queuing output 1P3Q4T-EGRESS
 ! 1P3Q4T egress queuing policy is attached to the interface (s)
```

You can verify the configuration in Example 25-4 with the following commands:

- **show class-map** [**type queuing**]
- **show policy-map** [**type queuing**]
- **show policy-map interface**

Note On M2 modules, only a single priority level is supported (at the time of this writing). Furthermore, when the **priority** command is enabled, the strict-priority queue is unbounded. However, the priority queue can be limited by adding the **shape** command to the priority queue class as a policy map action—which will enable shaped round-robin (SRR) to shape the priority traffic to the given rate.

Note If the **priority** keyword is not applied as the policy map action for this (**1p7q4t-out-pq1**) class, the queue is serviced as a nonpriority queue.

Note When the priority queue is configured, only **bandwidth remaining percent** is accepted on other classes (guaranteeing a minimum bandwidth for other classes from the remaining bandwidth of what is left after using the priority queue).

Note Configuring ingress queue limits is not supported on the 32-port 10-GE M1 module (N7K-M132XP-12L).

M2 Eight-Class (8Q2T Ingress / 1P3Q4T Egress) Queuing Model

In the eight-class model (illustrated in Figure 11-5, but adapted to the data center), the application class to queue mappings are as follows:

- Real-time traffic (marked EF, mapped by default to CoS 5) is allocated 10 percent bandwidth in the ingress 8Q2T model, but is enabled with strict-priority queuing in the egress 1P7Q4T model.
- Network control traffic (marked CoS 7) is allocated 5 percent bandwidth in the ingress 8Q2T model and a 5 percent bandwidth-remaining allocation in the egress 1P7Q4T model.
- Internetwork control traffic (marked CS6, mapped by default to CoS 6) is allocated 5 percent bandwidth in the ingress 8Q2T model and a 5 percent bandwidth-remaining allocation in the egress 1P7Q4T model.
- Video traffic (marked AF4 and mapped by default to CoS 4 or marked AF31 and explicitly mapped to CoS 4) is allocated 25 percent bandwidth in the ingress 8Q2T model and a 25 percent bandwidth-remaining allocation in the egress 1P7Q4T model.
- Signaling traffic (marked CS3, mapped by default to CoS 3) is assigned to a dedicated nonpriority queue with a 10 percent bandwidth allocation in the ingress 8Q2T model and a 10 percent bandwidth-remaining allocation in the egress 1P7Q4T model.

Note FCoE traffic is not present in this model because M2 modules do not support FCoE. However, FCoE is covered in the F2 module design section of this chapter.

- Transactional data (marked AF2, mapped by default to CoS 2) is assigned to dedicated nonpriority queue with a 10 percent bandwidth allocation in the ingress 8Q2T model and a 10 percent bandwidth-remaining allocation in the egress 1P7Q4T model.

- Bulk data (marked AF1, mapped by default to CoS 1) is assigned to dedicated nonpriority queue with a 10 percent bandwidth allocation in the ingress 8Q2T model and a 10 percent bandwidth-remaining allocation in the egress 1P7Q4T model.
- Best-effort traffic (marked DF, mapped by default to CoS 0) is assigned to the default queue with a 25 percent bandwidth allocation in the ingress 8Q2T model and a 35 percent bandwidth-remaining allocation in the egress 1P7Q4T model.

All bandwidth allocations will vary according to data center traffic profiles; for the sake of simplicity, (roughly) equal values have been allocated to these classes (except for the PQ and class default on egress).

As with the previous model, queue limits (buffer allocations) are assigned proportionally to bandwidth in the ingress queuing model, but are assigned inversely proportionally to the PQ/default queue in the egress queuing model.

CoS-based WRED is enabled only on nonpriority and noncontrol queues, namely: the video queue, the transactional data queue, the bulk data queue, and best-effort queue. On these queues, the minimum WRED threshold is set to 80 percent, and the maximum WRED threshold is set to 100 percent (the tail of the queue).

Figure 25-7 illustrates these eight class mappings, along with bandwidth and buffer allocations and congestion avoidance policies.

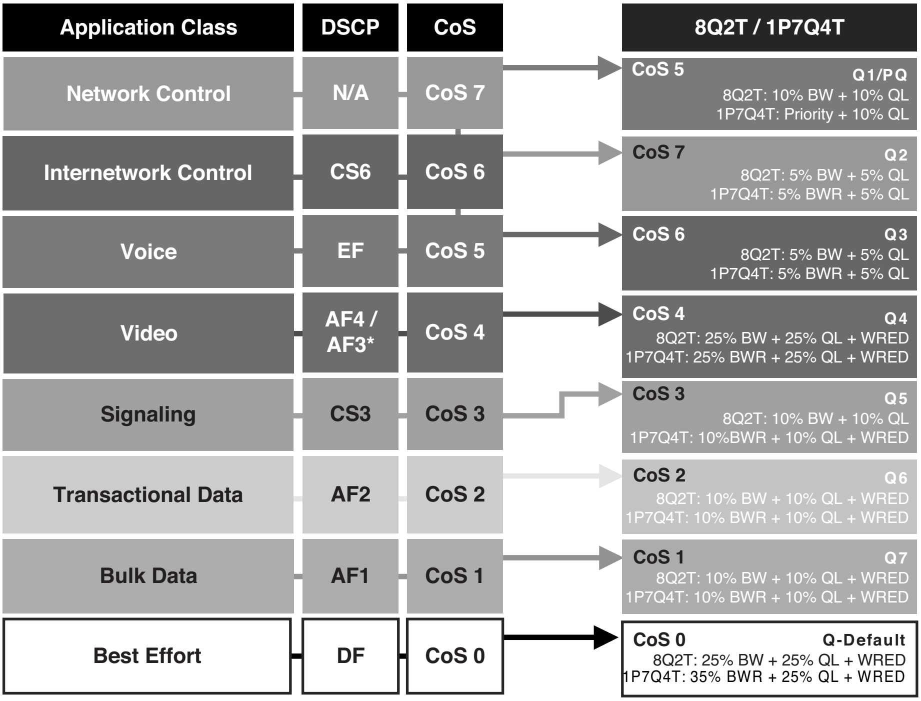


Figure 25-7 *Cisco Nexus 7000 M2 Eight-Class (8Q2T Ingress / 1P7Q4T Egress) Queuing Model*

Note This model assumes that multimedia-streaming video has been explicitly mapped to CoS 4, such as can be done by an ingress **type qos** policy matching on DSCP AF31/AF32/AF33 and setting CoS to 4.

Example 25-4 shows the corresponding configuration for eight-class (8Q2T ingress / 1P7Q4T egress) queuing.

Example 25-4 *Nexus 7000 M2 Eight-Class (8Q2T Ingress / 1P7Q4T Egress) Queuing Model Examples*

```
 ! This section configures the ingress queuing class maps
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q1
N7K(config-cmap-que)#  match cos 5
 ! Voice traffic (DSCP EF/CoS 5) is mapped to ingress Q1
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q2
N7K(config-cmap-que)#  match cos 7
 ! Network Control traffic (CoS 7) is mapped in ingress Q2
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q3
N7K(config-cmap-que)#  match cos 6
 ! Network control traffic (DSCP CS6/CoS 6) is mapped in ingress Q3
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q4
N7K(config-cmap-que)#  match cos 4
 ! Video traffic (DSCP AF4/CoS 4) is mapped in ingress Q4
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q5
N7K(config-cmap-que)#  match cos 3
 ! Signaling (DSCP CS3/CoS 3) is mapped to ingress Q5
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q6
N7K(config-cmap-que)#  match cos 2
 ! Transactional-Data (DSCP AF2/CoS 2) is mapped to ingress Q6
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q7
N7K(config-cmap-que)#  match cos 1
 ! Bulk-Data (DSCP AF1/CoS 1) is mapped to ingress Q7

 ! This section configures the egress queuing class maps
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-pq1
N7K(config-cmap-que)#  match cos 5
 ! Voice traffic (DSCP EF/CoS 5) is mapped to egress PQ1
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q2
N7K(config-cmap-que)#  match cos 7
 ! Network Control traffic (CoS 7) is mapped in egress Q2
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q3
```

```
N7K(config-cmap-que)#  match cos 6
 ! Network control traffic (DSCP CS6/CoS 6) is mapped in egress Q3
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q4
N7K(config-cmap-que)#  match cos 4
 ! Video traffic (DSCP AF4/CoS 3) is mapped in egress Q4
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q5
N7K(config-cmap-que)#  match cos 3
 ! Signaling (DSCP CS3/CoS 3) is mapped to egress Q5
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q6
N7K(config-cmap-que)#  match cos 2
 ! Transactional-Data (DSCP AF2/CoS 2) is mapped to egress Q6
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q7
N7K(config-cmap-que)#  match cos 1
 ! Bulk-Data (DSCP AF1/CoS 1) is mapped to egress Q7

 ! This section configures the 8Q2T ingress queuing policy map
N7K(config)# policy-map type queuing 8Q2T-INGRESS
N7K(config-pmap-que)#  class type queuing 8q2t-in-q1
N7K(config-pmap-c-que)#  bandwidth percent 10
N7K(config-pmap-c-que)#  queue-limit percent 10
 ! Ingress Q1 (Voice) is allocated 10% bandwidth and 10% queue-limit
N7K(config-pmap-c-que)# class type queuing 8q2t-in-q2
N7K(config-pmap-c-que)#  bandwidth percent 5
N7K(config-pmap-c-que)#  queue-limit percent 5
 ! Ingress Q2 (OAM) is allocated 5% bandwidth and 5% queue-limit
N7K(config-pmap-c-que)# class type queuing 8q2t-in-q3
N7K(config-pmap-c-que)#  bandwidth percent 5
N7K(config-pmap-c-que)#  queue-limit percent 5
 ! Ingress Q3 (Routing) is allocated 5% bandwidth and 5% queue-limit
N7K(config-pmap-c-que)# class type queuing 8q2t-in-q4
N7K(config-pmap-c-que)#  bandwidth percent 25
N7K(config-pmap-c-que)#  queue-limit percent 25
 ! Ingress Q4 (Video) is allocated 25% bandwidth and 25% queue-limit
N7K(config-pmap-c-que)#  random-detect cos-based
N7K(config-pmap-c-que)#  random-detect cos 4 minimum-threshold percent 80 maximum-
threshold percent 100
 ! CoS-Based WRED is enabled and tuned on video queue (Q4)
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q5
N7K(config-pmap-c-que)#   bandwidth percent 10
N7K(config-pmap-c-que)#   queue-limit percent 10
 ! Ingress Q5 (Signaling) gets 10% bandwidth and 10% queue-limit
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q6
N7K(config-pmap-c-que)#   bandwidth percent 10
N7K(config-pmap-c-que)#   queue-limit percent 10
 ! Ingress Q6 (Trans-Data) gets 10% bandwidth and 10% queue-limit
```

```
N7K(config-pmap-c-que)#   random-detect cos-based
N7K(config-pmap-c-que)#   random-detect cos 2 minimum-threshold percent 80 maximum-
threshold percent 100
 ! CoS-Based WRED is enabled and tuned on Trans-Data queue (Q6)
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q7
N7K(config-pmap-c-que)#   bandwidth percent 10
N7K(config-pmap-c-que)#   queue-limit percent 10
 ! Ingress Q7 (Bulk-Data) gets 10% bandwidth and 10% queue-limit
N7K(config-pmap-c-que)#   random-detect cos-based
N7K(config-pmap-c-que)#   random-detect cos 1 minimum-threshold percent 80 maximum-
threshold percent 100
 ! CoS-Based WRED is enabled and tuned on Bulk-Data queue (Q7)
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q-default
N7K(config-pmap-c-que)#   bandwidth percent 25
N7K(config-pmap-c-que)#   queue-limit percent 25
 ! Default Ingress Queue is allocated 25% bandwidth and 25% queue-limit
N7K(config-pmap-c-que)#   random-detect cos-based
N7K(config-pmap-c-que)#   random-detect cos 0 minimum-threshold percent 80 maximum-
threshold percent 100
 ! CoS-Based WRED is enabled and tuned on default queue

! This section configures the 1P7Q4T egress queuing policy map
N7K(config)# policy-map type queuing 1P7Q4T-EGRESS
N7K(config-pmap-que)# class type queuing 1p7q4t-out-pq1
N7K(config-pmap-c-que)#  priority
 ! Strict priority is enabled on egress PQ1
N7K(config-pmap-c-que)#  queue-limit percent 10
 ! Egress PQ (Realtime) is allocated 10% of buffers
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q2
N7K(config-pmap-c-que)#  bandwidth remaining percent 5
N7K(config-pmap-c-que)#  queue-limit percent 5
 ! Egress Q2 (OAM) is allocated 5% BWR and 5% queue-limit
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q3
N7K(config-pmap-c-que)#  bandwidth remaining percent 5
N7K(config-pmap-c-que)#  queue-limit percent 5
 ! Egress Q3 (Routing) is allocated 5% BWR and 5% queue-limit
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q4
N7K(config-pmap-c-que)#  bandwidth remaining percent 25
N7K(config-pmap-c-que)#  queue-limit percent 25
 ! Egress Q4 (Video) is allocated 25% BWR and 25% queue-limit
N7K(config-pmap-c-que)#  random-detect cos-based
N7K(config-pmap-c-que)#  random-detect cos 4 minimum-threshold percent 80 maximum-
threshold percent 100
 ! CoS-Based WRED is enabled and tuned on video queue (Q4)
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q5
N7K(config-pmap-c-que)#  bandwidth remaining percent 10
```

```
N7K(config-pmap-c-que)#  queue-limit percent 10
 ! Egress Q5 (Signaling) gets 10% BWR and 10% queue-limit
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q6
N7K(config-pmap-c-que)#  bandwidth remaining percent 10
N7K(config-pmap-c-que)#  queue-limit percent 10
 ! Egress Q6 (Trans-Data) gets 10% BWR and 10% queue-limit
N7K(config-pmap-c-que)#  random-detect cos-based
N7K(config-pmap-c-que)#  random-detect cos 2 minimum-threshold percent 80 maximum-
threshold percent 100
 ! CoS-Based WRED is enabled and tuned on Trans-Data queue (Q6)
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q7
N7K(config-pmap-c-que)#  bandwidth remaining percent 10
N7K(config-pmap-c-que)#  queue-limit percent 10
 ! Egress Q7 (Bulk-Data) gets 10% BWR and 10% queue-limit
N7K(config-pmap-c-que)#  random-detect cos-based
N7K(config-pmap-c-que)#  random-detect cos 1 minimum-threshold percent 80 maximum-
threshold percent 100
 ! CoS-Based WRED is enabled and tuned on Bulk-Data queue (Q7)
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q-default
N7K(config-pmap-c-que)#  bandwidth remaining percent 35
N7K(config-pmap-c-que)#  queue-limit percent 25
 ! Default Egress Queue is allocated 35% BWR and 25% queue-limit
N7K(config-pmap-c-que)#  random-detect cos-based
N7K(config-pmap-c-que)#  random-detect cos 0 minimum-threshold percent 80 maximum-
threshold percent 100
 ! CoS-Based WRED is enabled and tuned on default queue

 ! This section applies the queuing policies to the interface(s)
N7K(config)#  interface Ethernet 1/20
N7K(config-if)#  service-policy type queuing input 8Q2T-INGRESS
 ! 8Q2T ingress queuing policy is attached to the interface (s)
N7K(config-if)#  service-policy type queuing output 1P7Q4T-EGRESS
 ! 1P7Q4T egress queuing policy is attached to the interface (s)
```

You can verify the configuration in Example 25-4 with the following commands:

- **show class-map** [**type queuing**]
- **show policy-map** [**type queuing**]
- **show policy-map interface**

Note These M2 ingress and egress queuing models can also be adapted to support M1-Series 48-port 10/100/1000 modules. M1 10/100/1000 modules use 2Q4T and 1P3Q4T queuing structures. To do so, the 2Q4T can be left in the default configuration (which is almost the same as the model shown in Figure 25-5), and the four-class 1P3Q4T egress queuing model can be applied, as shown in Figure 25-6 and in Example 25-3—with the exception that the queuing class map names are changed to **1p3q4t-out-** (as opposed to **1p7q4T-out-**, as used in Example 25-4).

Note Configuring ingress queue limits is not supported on the 32-Port 10-GE M1 module (N7K-M132XP-12L), nor is ingress WRED.

M2 OTV Edge Device QoS Design

Cisco Overlay Transport Virtualization (OTV) is a technology that allows for Data Center Interconnect (DCI) between sites without changing or reconfiguring existing network designs. OTV provides an overlay that enables Layer 2 connectivity over an IP tunnel between separate Layer 2 domains This keeps the L2 domains independent—preserving the fault isolation and resiliency of separate broadcast domains—while gaining the extended-reach and load-balancing benefits of an IP-based interconnection. OTV was first introduced on the Cisco Nexus 7000 platform, but is also supported on the ASR 1000 family.

OTV introduces the concept of “MAC routing,” which means a control plane protocol is used to exchange MAC reachability information between network devices providing LAN extension functionality.

OTV also introduces the concept of dynamic encapsulation for Layer 2 flows that need to be sent to remote locations. Each Ethernet frame is individually encapsulated into an IP packet—with an OTV “shim” inserted between the IP header and the original Ethernet frame—and delivered across the transport network. This eliminates the need to establish virtual circuits, (pseudo wires) between the data center locations.

Nexus 7000 series switches functioning as OTV edge devices (between separate broadcast domains/data centers) receive Layer 2 traffic for all VLANs that need to be extended to remote locations and dynamically encapsulate the Ethernet frames into IP packets and forward these across the transport infrastructure.

From a QoS perspective, it is important to understand this encapsulation process, particularly how it relates to CoS and DSCP values, which is as follows:

1. Before the Layer 2 frame is encapsulated, the OTV edge device copies the CoS bits (802.1p) from the original Layer 2 header to the OTV shim header. Also, if the original frame is an IP packet, the original (inner) DSCP value is also copied to the outer DSCP. This allows for consistent QoS policies to be applied to OTV traffic across the transport infrastructure.

Note Untagged packets (that is, packets with no 802.1p CoS value) will have the outer DSCP value of the OTV packet set to 0. However, the inner DSCP of the encapsulated packet is always preserved.

2. Once the packet is received on the remote OTV edge device and decapsulated, the CoS value is recovered from the OTV shim and added to the 802.1Q header, allowing for preservation of both original CoS and DSCP values.

Note If the DSCP value in the outer IP header is modified in transit, these changes are not reflected in the inner DSCP value, which is exposed only after decapsulation.

Figure 25-8 illustrates DSCP and CoS preservation over OTV.

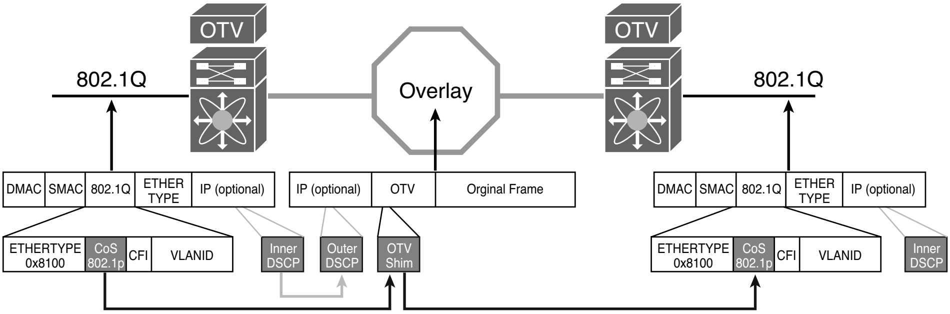


Figure 25-8 *Cisco Nexus 7000 OTV CoS and DSCP Preservation*

In addition to the functionality previously described, it is possible to map the CoS of the original packets to the DSCP value of the outer IP header for OTV packets by applying a specific policy map. This is important to uniquely mark and identify the OTV packets and apply QoS policy accordingly.

Note This behavior of preserving CoS and DSCP values over an OTV overlay was introduced in NX-OS Release 5.2(1).

OTV edge devices send control traffic to one another that is marked by default to DSCP CS6. When deploying OTV, it is highly recommended that one of two approaches to protecting such control plane traffic be used:

- Leave control plane traffic marked CoS 6 mapped to the egress priority queue (as is the default for Nexus 7000 egress queuing structures); incidentally, this is the approach used in the M2 four-class queuing model presented earlier.

- Assign CoS values 6 to a dedicated queue (if the assigned queues are dedicated for control plane flows, nonpriority queues will suffice); this is the approach used in the M2 eight-class queuing model.

Note In addition, it is recommended that network or VM control traffic—marked CoS 7—be serviced in similar manners as CoS 6 traffic in OTV designs, which has also been done in both the M2 four-class and eight-class models.

Therefore, both the M2 four-class and M2 eight-class queuing models presented in this section include OTV support.

Nexus 7000 F2 Modules: Architecture and QoS Design

The Nexus 7000 F2/F2e modules (hereafter referred to simply as F2 modules) can be used to extend the data center fabric to the core layer and include a different architecture from M2 modules, as shown in Figure 25-9.

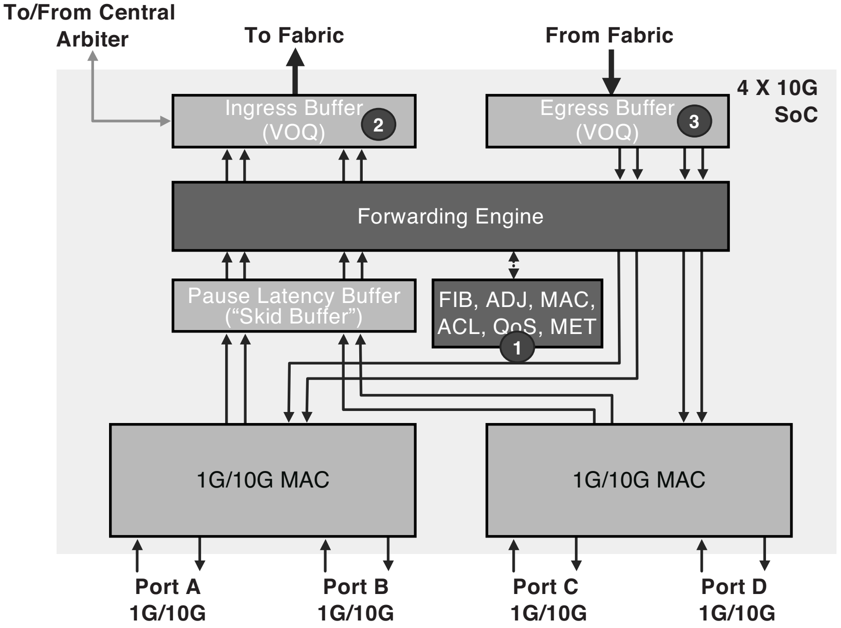


Figure 25-9 *Cisco Nexus 7000 F2 Module Hardware Architecture and QoS Enforcement Points*

QoS policies are applied at the following points on F2 modules:

1. Classification, marking and policing (**type qos**) policies are implemented within the forwarding engine.
2. Ingress and egress (**type queuing**) policies are applied in the VOQs *before* the packet traverses the three-stage switching fabric; the ingress queuing policies define the buffers allocations for these VOQs. However, it is the egress queuing policies define the scheduling of these VOQs.
3. Egress (**type queuing**) policies are applied in the VOQs *after* the packet traverses the three-stage switching fabric; buffer allocations for these post-fabric VOQs are fixed, but the egress queuing policy defines the scheduling of these VOQs.

As noted earlier, the scheduling of VOQ policies controlling access to the fabric is controlled by *egress* **type queuing** policies on F2 modules (rather than nonconfigurable fabric QoS policies, as on the M2 modules, as previously discussed).

Another key difference is that F2 modules do not have any physical port queuing buffers, as can be noted by contrasting Figure 25-10 (F2 virtual queues) with Figure 25-4 (M2 physical and virtual queues).

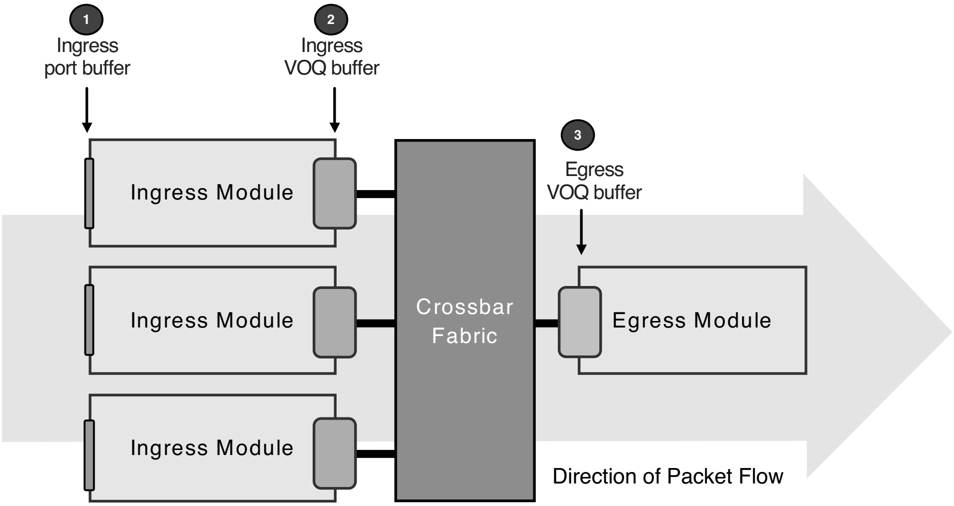


Figure 25-10 *Cisco Nexus 7000 F2 Modules: Virtual Queuing Buffers*

As shown in Figure 25-10, the F2 virtual queuing buffers and their corresponding functions are as follows:

1. **Ingress pause latency ("skid") buffer:** Absorbs packets in flight after external flow control is asserted and manages congestion for ingress forwarding/replication engines
2. **Ingress VOQ buffer (fabric queue):** Manages congestion toward the switching fabric

3. **Egress VOQ buffer:** Receives frames from the fabric and manages congestion for multidestination frames

F2 QoS Design Steps

When an F2 module is deployed within a Nexus 7000 core data center switch, then—in addition to trusting DSCP and provisioning ingress/egress queuing—it has the additional requirement of extending the DCB network to the core layer. Therefore, it must be configured with all the same **network-qos** parameters as all other devices participating in the DCB network. Furthermore, the ingress and egress queuing policies are dependent on the **network-qos** policy that has been assigned to the system. Therefore, this important first step must be taken before defining queuing policies.

Therefore, the three steps to configuring QoS on a Nexus 7000 with F2-Series modules in the role of a data center core switch are as follows:

1. Apply the correct **network-qos** policy to the system.
2. Configure ingress queuing.
3. Configure egress queuing.

F2 Network QoS Policy Design

When an F-Series module is inserted and becomes operational, five **network-qos** policies templates are automatically added to the system, as shown in Table 25-6. The **default-nq-7e-policy** is highlighted because is be the network QoS template featured in the designs that follow.

Table 25-6 *Default Network QoS Policy Templates*

Network QoS Template	CoS Values	
	CoS Values That May Be Dropped	*No Drop CoS Values*
default-nq-8e-policy	0,1,2,3,4,5,6,7	-
default-nq-8e-4q4q-policy	0,1,2,3,4,5,6,7	-
default-nq-7e-policy	0,1,2,4,5,6,7	3
default-nq-6e-policy	0,1,2,5,6,7	3,4
default-nq-4e-policy	0,5,6,7	1,2,3,4

Network QoS policy template names contain the abbreviation **nq** to refer to **network-qos.** In addition, the numbers 4, 6, 7, and 8 denote the number of the drop CoS values (CoS values that may be dropped during congestion) that is defined within the policy template, and finally **e** denotes Ethernet. The **4Q4T** in **default-nq-8e-4q4q-policy** refers to the fact that both the ingress and egress queuing structures use four queues (as opposed to the **default-nq-8e-policy** template that uses four queues on egress, but only two on ingress).

By default, the **default-nq-8e-policy** is applied to the system. These policy templates can be copied and modified or they can be simply applied globally by attaching them to **system qos**.

Note Unlike the Nexus 5500, the Nexus 7000 only allows **network-qos** policies to be applied to **system qos**.

Note Applying a **network-qos** policy to **system qos** should be done only on the primary/administrative VDC, because these changes affect all interfaces in all VDCs and are disruptive.

Not all **network-qos** templates will be able to be adapted to four-class and eight-class queuing models in this design chapter. Therefore, the **default-nq-7e-policy network qos** template is featured because it meets the minimum requirements for enabling FCoE. Therefore, in Example 25-10, the highlighted **network-qos** template **default-nq-7e-policy** is applied globally (that is, to **system qos**) because this policy autoconfigures a no-drop service for FCoE.

Example 25-10 *Nexus 7000 F2 FCoE (7e) Network QoS Policy Template Application Example*

```
 ! This section applies the default-nq-7e-policy to system qos
N7K(config)# system qos
N7K(config-sys-qos)# service-policy type network-qos default-nq-7e-policy
 ! Applies the default-nq-7e-policy globally
2013 May 30 09:47:02 N7K -QOS %$ VDC-3 %$ %IPQOSMGR-2-QOSMGR_NETWORK_QOS_POLICY_
CHANGE: Policy default-nq-7e-policy is now active
2013 May 30 09:47:02 N7K%$ VDC-1 %$ %IPQOSMGR-2-QOSMGR_NETWORK_QOS_POLICY_CHANGE:
Policy default-nq-7e-policy is now active
N7K (config-sys-qos)#
```

You can verify the configuration in Example 25-10 with the following commands:

- **show class-map type network-qos** [**c-nq-7e-drop** | **c-nq-7e-ndrop-fcoe**] (as shown in Example 25-11)
- **show policy-map type network-qos** [**default-nq-7e-policy**] (as shown in Example 25-12)
- **show policy-map system** (as shown in Example 25-13)

Example 25-11 *Verifying FCoE Class Map Configuration:* ***show class-map type network-qos [c-nq-7e-drop | c-nq-7e-ndrop-fcoe]***

```
N7K# show class-map type network-qos c-nq-7e-drop

  Type network-qos class-maps
  ===========================
  class-map type network-qos match-any c-nq-7e-drop
      Description: 7E Drop CoS map
    match cos 0-2,4-7
N7K# show class-map type network-qos c-nq-7e-ndrop-fcoe
Type network-qos class-maps
=============================================
class-map type network-qos match-any c-nq-7e-ndrop-fcoe
Description: 7E No-Drop FCoE CoS map
match cos 3
match protocol fcoe
```

As shown in Example 25-11, all CoS values except 3 are assigned to the Drop class (**c-nq-7e-drop)**, while FCoE is matched on CoS 3 and by **protocol fcoe** and assigned to the No Drop class (**c-nq-7e-ndrop-fcoe).** Incidentally, the **match protocol fcoe** command also advertises **fcoe** in DCB Exchanges (DCBX).

Example 25-12 *Verifying Network QoS Policy Map Configuration:* ***show policy-map type network-qos [default-nq-7e-policy]***

```
N7K# show policy-map type network-qos default-nq-7e-policy
  Type network-qos policy-maps
  ============================
  policy-map type network-qos default-nq-7e-policy
    class type network-qos c-nq-7e-drop
      congestion-control tail-drop
      mtu 1500
    class type network-qos c-nq-7e-ndrop-fcoe
      pause
      mtu 2112
```

As shown in Example 25-12, FCoE is given a no-drop (IEEE 802.1Qbb) Priority Flow Control (PFC) service via the **pause** command. Additionally the MTU has been automatically set to 2112 to accommodate FCoE.

Note Some meticulous readers may note that the default maximum transmission unit (MTU) specified by the Nexus 5500 for FCoE (of 2158 bytes) differs from the MTU defined by the Nexus 7000 (of 2112 bytes). This difference is based on how these respective systems calculate the MTUs: The Nexus 5500 includes 46 bytes of Ethernet overhead on top of the 2112 bytes required by Fibre Channel; whereas the Nexus 7000 ignores the Ethernet overhead within the MTU definition, implicitly accounting for it later.

Example 25-13 *Verifying System Network QoS Policy:* ***show policy-map system***

```
N7K# show policy-map system
N7K (config-sys-qos)# show policy-map system
  Type network-qos policy-maps
  ============================
  policy-map type network-qos default-nq-7e-policy
    class type network-qos c-nq-7e-drop
      match cos 0-2,4-7
      congestion-control tail-drop
      mtu 1500
    class type network-qos c-nq-7e-ndrop-fcoe
      match cos 3
      match protocol fcoe
      pause
      mtu 2112

  Service-policy (queuing) input:   default-4q-7e-in-policy
    policy statistics status:   disabled (current status: disabled)
    Class-map (queuing):   c-4q-7e-drop-in (match-any)
      Match: class-map type queuing match-any 4q4t-7e-in-q1
      Match: class-map type queuing match-any 4q4t-7e-in-q-default
      Match: class-map type queuing match-any 4q4t-7e-in-q3
      service-policy type queuing default-4q-7e-drop-in-policy
         Service-policy (queuing) input:   default-4q-7e-drop-in-policy
            policy statistics status:   disabled (current status: disabled)
            Class-map (queuing):   4q4t-7e-in-q1 (match-any)
              queue-limit percent 10
              bandwidth percent 25
            Class-map (queuing):   4q4t-7e-in-q-default (match-any)
              queue-limit percent 45
              bandwidth percent 25
            Class-map (queuing):   4q4t-7e-in-q3 (match-any)
              queue-limit percent 45
              bandwidth percent 25
      queue-limit percent 70
    Class-map (queuing):   c-4q-7e-ndrop-in (match-any)
      Match: class-map type queuing match-any 4q4t-7e-in-q4
      service-policy type queuing default-4q-7e-ndrop-in-policy
        Service-policy (queuing) input:   default-4q-7e-ndrop-in-policy
            policy statistics status:   disabled (current status: disabled)
            Class-map (queuing):   4q4t-7e-in-q4 (match-any)
              queue-limit percent 100
              bandwidth percent 25
      queue-limit percent 30
  Service-policy (queuing) output:   default-4q-7e-out-policy
    policy statistics status:   disabled (current status: disabled)
    Class-map (queuing):   c-4q-7e-drop-out (match-any)
      Match: class-map type queuing match-any 1p3q1t-7e-out-pq1
      Match: class-map type queuing match-any 1p3q1t-7e-out-q3
      Match: class-map type queuing match-any 1p3q1t-7e-out-q-default
      service-policy type queuing default-4q-7e-drop-out-policy
       Service-policy (queuing) output:   default-4q-7e-drop-out-policy
            policy statistics status:   disabled (current status: disabled)
            Class-map (queuing):   1p3q1t-7e-out-pq1 (match-any)
              priority level 1
            Class-map (queuing):   1p3q1t-7e-out-q3 (match-any)
              bandwidth remaining percent 50
            Class-map (queuing):   1p3q1t-7e-out-q-default (match-any)
              bandwidth remaining percent 50
      bandwidth remaining percent 80
    Class-map (queuing):   c-4q-7e-ndrop-out (match-any)
      Match: class-map type queuing match-any 1p3q1t-7e-out-q2
      service-policy type queuing default-4q-7e-ndrop-out-policy
      Service-policy (queuing) output:   default-4q-7e-ndrop-out-policy
            policy statistics status:   disabled (current status: disabled)
            Class-map (queuing):   1p3q1t-7e-out-q2 (match-any)
              bandwidth remaining percent 100
      bandwidth remaining percent 20
N7K(config-sys-qos)#
```

As shown in Example 25-13, the **default-nq-7e-policy** has been applied to the system (globally), complete with its default ingress and egress queuing policies.

It may be of interest to note that the ingress and egress queuing policies are actually hierarchical QoS policies (with queuing policies nested within queuing policies). Specifically, the **default-4q-7e-drop-in-policy** and **default-4q-7e-ndrop-in-policy** are nested within the **default-4q-7e-in-policy** on ingress, and the **default-4q-7e-drop-out-policy** and **default-4q-7e-ndrop-out-policy** are nested within the **default-4q-7e-out-policy** on egress. The egress hierarchical queuing policies are illustrated in Figure 25-13.

F2 Queuing Models

Each **network-qos** policy template includes its own preconfigured, default ingress and egress queuing models. Each of these can be cloned and modified, as will be shown.

Table 25-6 summarizes the default ingress and egress queuing models by **network-qos** policy-template.

Table 25-6 *F2 Default Ingress and Egress Queuing Models by Network QoS Policy Template*

Policy Template	Default Ingress Queuing Policy	Default Egress Queuing Policy
default-nq-8e-policy	default-nq-8e-in-policy	default-nq-8e-out-policy
default-nq-8e-4q4q-policy	default-8e-4q4q-in-policy	default-8e-4q4q-out-policy
default-nq-7e-policy	default-nq-7e-in-policy	default-nq-7e-out-policy
default-nq-6e-policy	default-nq-6e-in-policy	default-nq-6-out-policy
default-nq-4e-policy	default-nq-4e-in-policy	default-nq-4e-out-policy

Each default ingress and egress queuing policy has corresponding system-defined class map names, as summarized in Table 25-7 (for ingress queuing policies) and Table 25-8 (for egress queuing policies).

Table 25-7 *F2 Default Ingress Queuing Policy Class Map Names*

F2 Default Ingress Policy Map	F2 Default Ingress Class Map Names
default-nq-8e-in-policy	2q4t-8e-in-q1 and 2q4t-8e-in-q-default
default-8e-4q4q-in-policy	4q1t-8e-4q4q-in-q1, 4q1t-8e-4q4q-in-q-default, 4q1t-8e-4q4q-in-q3, and 4q1t-8e-4q4q-in-q4
default-nq-7e-in-policy	4q4t-7e-in-q1, 4q4t-7e-in-q-default, 4q4t-7e-in-q3, and 4q4t-7e-in-q4
default-nq-6e-in-policy	4q4t-6e-in-q1, 4q4t-6e-in-q-default, 4q4t-6e-in-q3, and 4q4t-6e-in-q4
default-nq-4e-in-policy	4q4t-4e-in-q1, 4q4t-4e-in-q-default, 4q4t-4e-in-q3, and 4q4t-4e-in-q4

Note Although some of these class map names seem to indicate four thresholds per nonpriority ingress queue (that is, 2q4t-/4q4t-) this is actually a misnomer. At the time of this writing, all F2 ingress queues structures support only a single drop threshold per queue (that is, the tail). Therefore in the diagrams and text that follow, all F2 ingress queuing structures will be more correctly referenced as 2Q1T or 4Q1T.

Note You may notice that this misnomer has been already been corrected in the newest policy map template (**default-8e-4q4q-in-policy**, which was introduced in NX-OS 6.1(3)) which lists all class maps as 4q1t-.

Table 25-8 *F2 Default Egress Queuing Policy Class Map Names*

F2 Default Egress Policy Names	F2 Default Egress Class Map Names
default-nq-8e-out-policy	1p3q1t-8e-out-pq1, 1p3q1t-8e-out-q2, 1p3q1t-8e-out-q3, and 1p3q1t-8e-out-q-default
default-8e-4q4q-out-policy	1p3q1t-8e-4q4q-out-pq1, 1p3q1t-8e-4q4q-out-q2, 1p3q1t-8e-4q4q-out-q3, and 1p3q1t-8e-4q4q-out-q-default
default-nq-7e-out-policy	1p3q1t-7e-out-pq1, 1p3q1t-7e-out-q2, 1p3q1t-7e-out-q3, and 1p3q1t-7e-out-q-default
default-nq-6-out-policy	3p1q1t-6e-out-pq1, 3p1q1t-6e-out-pq2, 3p1q1t-6e-out-pq3, and 3p1q1t-6e-out-q-default
default-nq-4e-out-policy	2p2q1t-4e-out-pq1, 2p2q1t-4e-out-pq2, 2p2q1t-4e-out-q3, and 2p2q1t-4e-out-q-default

F2 Default Queuing Models

Figure 25-11 through Figure 25-13 illustrate the F2 default ingress and egress queuing models.

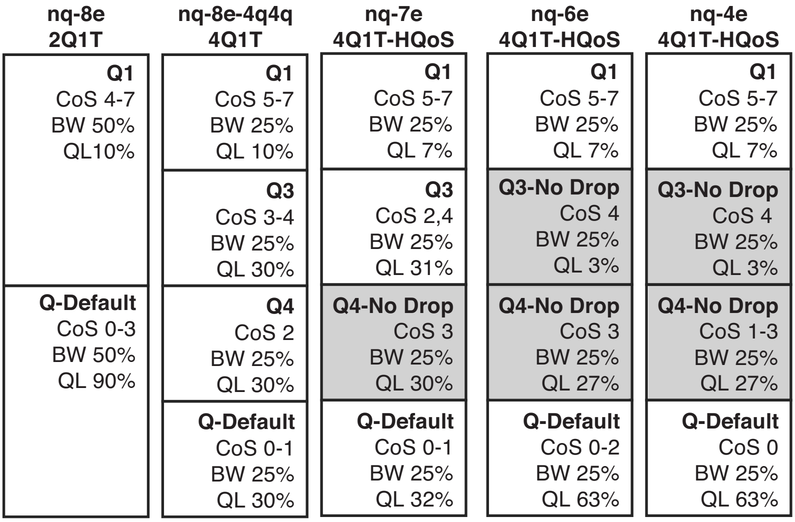
Ingress queuing policies control the buffer allocations to the pre-fabric VOQs, as shown in Figure 25-9 and Figure 25-10. Any **bandwidth** values assigned to these ingress do not actually affect any internal switch scheduling (because the internal scheduling of these queues is determined by the *egress* queuing policies), but rather are used in DCBX advertisements to peers to drive Enhanced Transmission Selection (ETS).

Note Starting with the Cisco NX-OS 6.1 release, DSCP-to-queue mapping is supported on F2 modules, in the ingress direction (for IPv4 traffic); this is done by using the **match dscp** command with the **2q4t-8e-in-q1** class map and the **2q4t-8e-in-q-default** class map.

In addition, starting with NX-OS 6.1(2) release, DSCP-to-queue mapping is supported for F2 ingress queues for IPv6 traffic.

However, to maintain compatibility with the other queuing models in this chapter, CoS-to-queue mapping is used in the examples that follow.

Egress queue buffer allocations are fixed and nonconfigurable. However, Figure 25-12 summarizes the various per-template egress queuing policies.

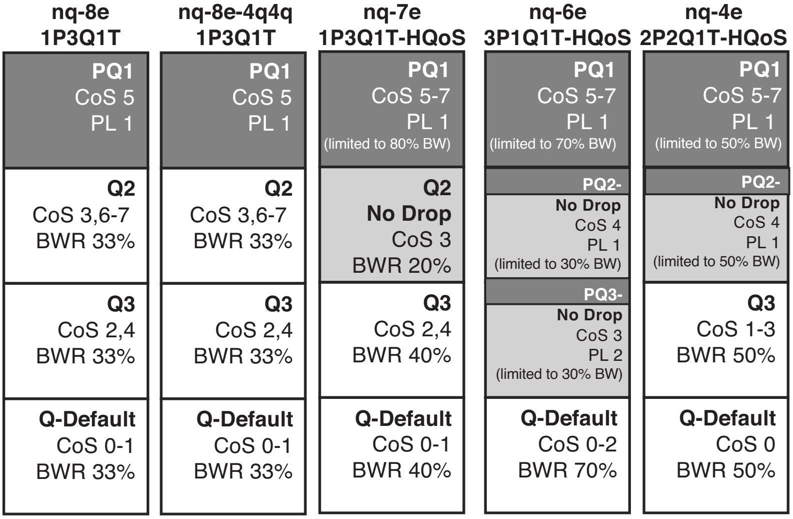


nq-8e 2Q1T	nq-8e-4q4q 4Q1T	nq-7e 4Q1T-HQoS	nq-6e 4Q1T-HQoS	nq-4e 4Q1T-HQoS
Q1 CoS 4-7 BW 50% QL10%	Q1 CoS 5-7 BW 25% QL 10%	Q1 CoS 5-7 BW 25% QL 7%	Q1 CoS 5-7 BW 25% QL 7%	Q1 CoS 5-7 BW 25% QL 7%
	Q3 CoS 3-4 BW 25% QL 30%	Q3 CoS 2,4 BW 25% QL 31%	Q3-No Drop CoS 4 BW 25% QL 3%	Q3-No Drop CoS 4 BW 25% QL 3%
Q-Default CoS 0-3 BW 50% QL 90%	Q4 CoS 2 BW 25% QL 30%	Q4-No Drop CoS 3 BW 25% QL 30%	Q4-No Drop CoS 3 BW 25% QL 27%	Q4-No Drop CoS 1-3 BW 25% QL 27%
	Q-Default CoS 0-1 BW 25% QL 30%	Q-Default CoS 0-1 BW 25% QL 32%	Q-Default CoS 0-2 BW 25% QL 63%	Q-Default CoS 0 BW 25% QL 63%

BW= DCBX Advertised Bandwidth
QL = Queue Limit

Note: Q2 is the default queue in all models and is out-of-sequence in these diagrams

Figure 25-11 *Cisco Nexus 7000 F2 Default Ingress Queuing Models by Network QoS Policy Template*



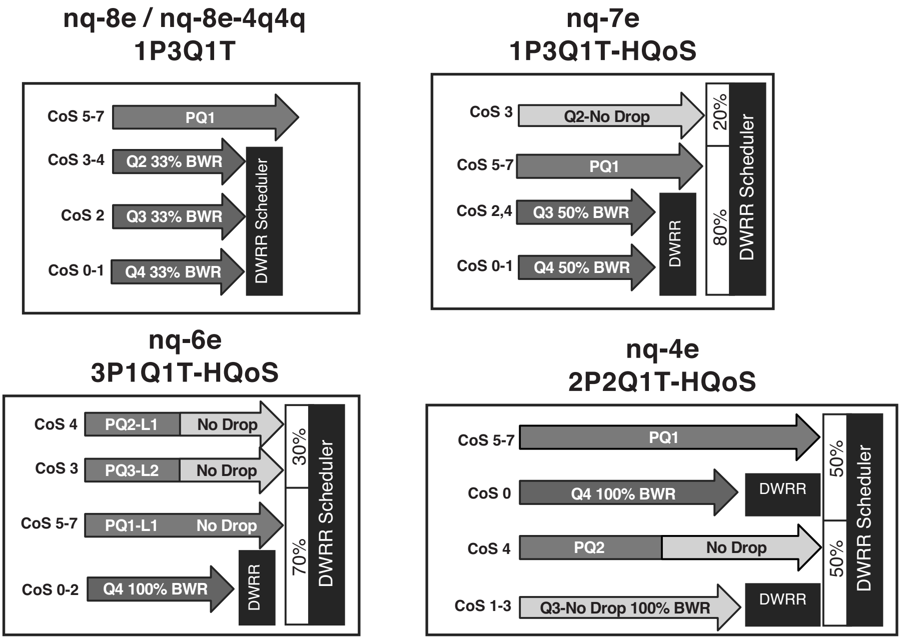
nq-8e 1P3Q1T	nq-8e-4q4q 1P3Q1T	nq-7e 1P3Q1T-HQoS	nq-6e 3P1Q1T-HQoS	nq-4e 2P2Q1T-HQoS
PQ1 CoS 5 PL 1	PQ1 CoS 5 PL 1	PQ1 CoS 5-7 PL 1 (limited to 80% BW)	PQ1 CoS 5-7 PL 1 (limited to 70% BW)	PQ1 CoS 5-7 PL 1 (limited to 50% BW)
Q2 CoS 3,6-7 BWR 33%	Q2 CoS 3,6-7 BWR 33%	Q2 No Drop CoS 3 BWR 20%	PQ2- No Drop CoS 4 PL 1 (limited to 30% BW)	PQ2- No Drop CoS 4 PL 1 (limited to 50% BW)
Q3 CoS 2,4 BWR 33%	Q3 CoS 2,4 BWR 33%	Q3 CoS 2,4 BWR 40%	PQ3- No Drop CoS 3 PL 2 (limited to 30% BW)	Q3 CoS 1-3 BWR 50%
Q-Default CoS 0-1 BWR 33%	Q-Default CoS 0-1 BWR 33%	Q-Default CoS 0-1 BWR 40%	Q-Default CoS 0-2 BWR 70%	Q-Default CoS 0 BWR 50%

PQ = Priority Queue / PL = Priority Level
BWR= Bandwidth Remaining (after all PQ servicing)

Figure 25-12 *Cisco Nexus 7000 F2 Default Egress Queuing Models by Network QoS Policy Template*

Note For the hierarchical QoS (HQoS) policies shown in Figure 25-11 and 25-12, the queue limits and bandwidth-remaining values were calculated by factoring the inner (child) policy bandwidth or queue-limit values by the outer (parent) policy values.

In addition, Figure 25-13 shows the scheduling policies—highlighting the hierarchical scheduling policies of the 7e, 6e, and 4e egress queuing policies.



PQ = Priority Queue & L1/2 = Priority Level 1/2
(unless specified otherwise PQs are L1)
BWR = Bandwidth Remaining
DWRR = Deficit Weighted Round Robin
Q4 is the Default Queue in all examples

Figure 25-13 *Cisco Nexus 7000 F2 Default Egress Scheduling Policies by Network QoS Policy Template*

Note The ingress and egress queuing configurations summarized and illustrated above are valid as of NX-OS 6.1(3). However, these default CoS-to-queue mappings, bandwidth, and buffer allocations have all been known to change from one software version to another. It is recommended to verify the current queuing settings by performing a **show policy-map type queuing default-***policy-map-name* command.

F2 Four-Class (4Q1T Ingress / 1P3Q1T Egress) Queuing Model

To achieve four-class application class to queue mapping functionally compatible with the four-class model illustrated earlier in this chapter (in Figure 25-6), all you need to do is to apply the nq-7e policy template (**default-nq-7e-policy**) to **system qos**—as was shown in Example 25-10. The default ingress and egress queuing settings (shown in Figure 25-11, 25-12, and 25-13) of this nq-7e policy-template take care of all the mapping and provisioning requirements of this model automatically.

You can verify this configuration with the following commands:

- **show policy-map system** (as shown in Example 25-13)
- **show class-map type queuing** [**4q4t-7e-in-q1** | **4q4t-7e-in-q3** | **4q4t-7e-in-q4** | **4q4t-7e-in-q-default** | **1p3q1t-7e-out-pq1** | **1p3q1t-7e-out-q2** | **1p3q1t-7e-out-q3** | **1p3q1t-7e-out-q-default**]
- **show policy-map type queuing** [**default-4q-7e-in-policy** | **default-4q-7e-drop-in-policy** | **default-4q-7e-ndrop-in-policy** | **default-4q-7e-out-policy** | **default-4q-7e-drop-out-policy** | **default-4q-7e-ndrop-out-policy**]
- **show policy-map interface**

F2 Eight-Class (4Q1T Ingress / 1P3Q1T Egress) Queuing Model

As with the four-class model, the nq-7e policy template provides a highly functional adaptation of the eight-class model—without requiring any additional configuration. Figure 25-14 illustrates this eight-class model mapped to the default nq-7e policy-template.

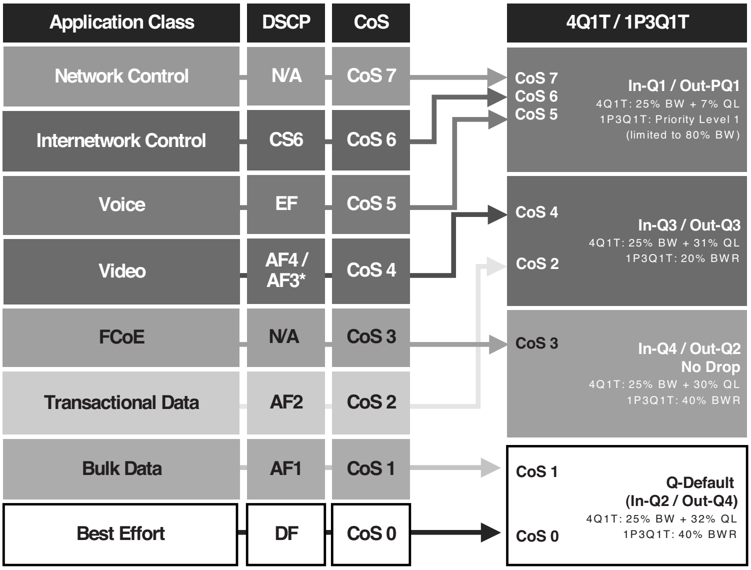


Figure 22-14 *Cisco Nexus 7000 F2 Eight-Class (nq-7e: 4Q1T Ingress / 1P3Q1T Egress) Default Queuing Model*

Note This model assumes that multimedia-streaming video has been explicitly mapped to CoS 4, such as can be done by an ingress **type qos** policy matching on DSCP AF31/AF32/AF33 and setting CoS to 4.

However, for the sake of demonstration purposes, let's consider an example where you might want to modify this nq-7e policy template to tune it more tightly to your requirements and environment.

You first have to clone the default queuing policy—either prepending it with a prefix (with the **prefix** keyword) or appending it with a suffix (with the **suffix** keyword). This is shown in Example 25-14, where the default policy is prepended with the prefix MODIFIED-.

Example 25-14 *Nexus 7000 F2 Eight-Class (7e: 4Q1T Ingress / 1P3Q1T Egress) Modified Queuing Model Examples: Part 1, Cloning the Default Policy*

```
 ! This section clones the default 7e policy-template
N7K# qos copy policy-map type queuing default-4q-7e-in-policy prefix MODIFIED-
```

After the policy has been cloned and prepended, it will appear within the configuration, as shown in Example 25-15.

Example 25-15 *Nexus 7000 F2 Eight-Class (nq-7e: 4Q1T Ingress / 1P3Q1T Egress) Modified Queuing Model Examples: Part 2, Verifying the Cloning Policy*

```
N7K# show run | begin MODIFIED
policy-map type queuing MODIFIED-4q-7e-drop-in
  class type queuing 4q4t-7e-in-q1
    queue-limit percent 10
    bandwidth percent 25
  class type queuing 4q4t-7e-in-q-default
    queue-limit percent 45
    bandwidth percent 25
  class type queuing 4q4t-7e-in-q3
    queue-limit percent 45
    bandwidth percent 25

policy-map type queuing MODIFIED-4q-7e-ndrop-in
  class type queuing 4q4t-7e-in-q4
    queue-limit percent 100
    bandwidth percent 25

policy-map type queuing MODIFIED-4q-7e-in
  class type queuing c-4q-7e-drop-in
```

```
    service-policy type queuing MODIFIED-4q-7e-drop-in
    queue-limit percent 70
  class type queuing c-4q-7e-ndrop-in
    service-policy type queuing MODIFIED-4q-7e-ndrop-in
    queue-limit percent 30
```

As you may notice, the ingress queue-limit allocation is not evenly balanced among the three drop queues—Q1, Q2/default, and Q3, as highlighted in Example 25-15. These queue limits are 10 percent, 45 percent, and 45 percent, respectively, and are applied against the parent policy's queue limit of 70 percent (for effective queue limits of 7 percent, 31.5 percent, and 31.5 percent, respectively).

Suppose that you to balance these queue limits more evenly. If so, you could make the policy changes shown in Example 25-16.

Example 25-16 *Nexus 7000 F2 Eight-Class (7e: 4Q1T Ingress / 1P3Q1T Egress) Modified Queuing Model Examples: Part 3, Modifying the Policy*

```
 ! This section modifies the cloned ingress queuing policy
N7K(config)# policy-map type queuing MODIFIED-4q-7e-drop-in
N7K(config-pmap-que)# class type queuing 4q4t-7e-in-q1
N7K(config-pmap-c-que)# queue-limit percent 33
 ! Changes queue-limit for In-Q1 to 33% (of 70%) ≈ 23%
N7K(config-pmap-c-que)# class type queuing 4q4t-7e-in-q-default
N7K(config-pmap-c-que)# queue-limit percent 34
 ! Changes queue-limit for the default queue to 34% (of 70%) ≈ 24%
N7K(config-pmap-c-que)# class type queuing 4q4t-7e-in-q3
N7K(config-pmap-c-que)# queue-limit percent 33
 ! Changes queue-limit for In-Q3 to 33% (of 70%) ≈ 23%
```

With these changes made, you can verify the modified policy, as shown in Example 25-17.

Example 25-17 *Nexus 7000 F2 Eight-Class (7e: 4Q1T Ingress / 1P3Q1T Egress) Modified Queuing Model Examples: Part 4, Verifying the Modified Policy*

```
N7K# show run | begin MODIFIED
policy-map type queuing MODIFIED-4q-7e-drop-in
  class type queuing 4q4t-7e-in-q1
    queue-limit percent 33
    bandwidth percent 25
  class type queuing 4q4t-7e-in-q-default
    queue-limit percent 34
    bandwidth percent 25
  class type queuing 4q4t-7e-in-q3
    queue-limit percent 33
    bandwidth percent 25

policy-map type queuing MODIFIED-4q-7e-ndrop-in
  class type queuing 4q4t-7e-in-q4
    queue-limit percent 100
    bandwidth percent 25

policy-map type queuing MODIFIED-4q-7e-in
  class type queuing c-4q-7e-drop-in
    service-policy type queuing MODIFIED-4q-7e-drop-in
    queue-limit percent 70
  class type queuing c-4q-7e-ndrop-in
    service-policy type queuing MODIFIED-4q-7e-ndrop-in
    queue-limit percent 30
```

As you can see from Example 25-17, the desired changes have been made to the default ingress queuing policy. The policy can now be applied to an interface (or interface range) and will then replace the default ingress queuing policy on the interfaces, as shown in Example 25-18. When applying the policy, make sure that it is the parent queuing policy that is applied (which in this case is **MODIFIED-4q-7e-in**, as highlighted earlier) and not the child-policies (which in this case are **MODIFIED-4q-7e-drop-in** and **MODIFIED-4q-7e-ndrop-in**).

Example 25-18 *Nexus 7000 F2 Eight-Class (7e: 4Q1T Ingress / 1P3Q1T Egress) Modified Queuing Model Examples: Part 5. Applying the Modified (HQoS-Parent) Ingress Queuing Policy to an Interface*

```
N7K(config)# interface Ethernet3/44
N7K(config-if)# service-policy type queuing input MODIFIED-4q-7e-in
 ! Attaches the modified parent ingress queuing policy to the int(s)
```

You can verify this configuration with the following commands:

- **show policy-map type queuing** [**MODIFIED-4q-7e-in** | **MODIFIED-4q-7e-drop-in** | **MODIFIED-4q-7e-ndrop-in**]
- **show policy-map interface**

Similar changes can be made to egress queuing policies by following the same steps outlined in Example 25-15 through Example 25-18.

FEX QoS Design

The Cisco Nexus 2000 Series Fabric Extender (FEX) is a remote linecard that you can connect to the Cisco Nexus 7000 Series switch to extend the fabric. The FEX has 48 1-Gbps server-facing ports, which are satellite ports, and four uplink ports that you can use to connect it to the Cisco Nexus 7000 Series switch. The four ports on the Cisco Nexus 7000 Series switch that connect to the uplink ports are fabric ports. Only QoS policies can be configured on the server-facing FEX ports. Currently, queuing on the FEX interfaces is not supported. QoS design for Nexus 2000 FEX was discussed extensively in the previous chapter.

Additional M2/F2 QoS Design Options

These QoS designs represent a generic building-block for the Nexus 7000 in a data center core switch role, but they are by no means the only design options available to you. For example, the Nexus 7000 may connect directly to servers—playing the role of a data center access switch. Therefore, some additional design models are presented in the following sections:

- Trusted Server Model
- Untrusted Server Model
- Single-Application Server Model
- Multi-Application Server Model
- Application-Policing Server Model

Each of these additional QoS design options are discussed in turn. Because most of these models feature **type qos** policies, they apply equally to M2- and F2-Series modules. However, any module-specific differences are highlighted in the respective sections.

Trusted Server Model

In many cases in the data center network administrators will be able to trust L2 and L3 markings set on application servers, especially as a certain level of administrative control has already been exercised by allowing the servers to be physically installed within the data center.

The Nexus 7000 trusts CoS and DSCP by default. Therefore, no explicit configuration is needed on interfaces/ports connecting to trusted application servers.

Untrusted Server Model

If the networking team does not trust an application server (or the team administering it—as has been known to happen), they may configure the policies shown in this section to disable trust on an ingress port.

The untrusted server policy may be comprised of two parts:

1. The first policy is an ingress queuing (**type queuing**) policy to rewrite CoS to 0; unfortunately, DSCP cannot be re-marked within a queuing policy (at the time of this writing), only CoS. Furthermore, CoS can only be re-marked on the default class when configured within an ingress queuing policy.

Note This policy is only required if the port is an 802.1Q trunk.

2. The second policy is a **type qos** policy to re-mark DSCP to 0.

Both policies are combined within Example 25-19. To cover additional ground, this example has been modified to illustrate application on a M1-Series 10/100/1000 linecard—which has been done by referencing the **2q4t-in-q-default** class map within the queuing policy map.

Note To apply this policy to a M2 series module, the default class reference of **2q4t-in-q-default** would be replaced with **8q2t-in-q-default**.

Example 25-19 *Nexus 7000 Untrusted Server Model: M1 (10/100/1000) Module Example*

```
 ! This section configures the Untrusted Server Queuing policy map
N7K(config)# policy-map type queuing M1-UNTRUSTED-SERVER-QUEUING-POLICY
N7K(config-pmap-que)# class type queuing 2q4t-in-q-default
 ! Applies the policy map actions that follow
 ! to the M1 10/100/1000 2Q4T default ingress queue class map
N7K(config-pmap-c-que)# bandwidth percent 100
N7K(config-pmap-c-que)# queue-limit percent 100
 ! All bandwidth and buffers are assigned to the default queue
N7K(config-pmap-c-que)# set cos 0
 ! CoS is set to 0 on ingress

 ! This section configures the Untrusted Server QoS policy map
N7K(config)# policy-map type qos UNTRUSTED-SERVER-QOS-POLICY
N7K(config-pmap-qos)# class class-default
N7K(config-pmap-c-qos)# set dscp 0
 ! All traffic is re-marked to DSCP 0

 ! This section applies both policies to the Untrusted Server port
N7K(config)# interface ethernet 2/40
N7K(config-if)# service-policy type queuing input M1-UNTRUSTED-SERVER-QUEUING-POLICY
```

```
 ! Attaches the Untrusted Server Queuing policy to the port
N7K(config-if)# service-policy type qos input UNTRUSTED-SERVER-QOS-POLICY
 ! Attaches the Untrusted Server QoS policy to the port
```

You can verify the configuration in Example 25-19 with the following commands:

- **show class-map** [**type** { **qos** |**queuing** }]
- **show policy-map** [**type** { **qos** |**queuing** }]
- **show policy-map interface**

Example 25-19 presents the recommended model for setting a port to an untrusted state on a Nexus 7000. However, some advanced users may wonder at the logic of this policy, because it might seem to re-mark only CoS values 0 and 1. This may seem so because—by default—only CoS values 0 and 1 are mapped to the **2q4t-in-q-default** default-class (and again, only the default-class supports re-marking within an ingress queuing policy). You can verify the CoS-to-queue mapping with the internal **show** command **show system internal qos queuing config interface**, as shown in Example 25-20.

Example 25-20 *Verifying Internal CoS-to-Queue Mappings*

```
N7K# show system internal qos queuing config interface e 2/40 | begin COS2Q
<snip>
COS2Q Config
Direction: ingress
COS 0 Queue: 2q4t-in-q-default
COS 1 Queue: 2q4t-in-q-default
COS 2 Queue: 2q4t-in-q1
COS 3 Queue: 2q4t-in-q1
COS 4 Queue: 2q4t-in-q1
COS 5 Queue: 2q4t-in-q1
COS 6 Queue: 2q4t-in-q1
COS 7 Queue: 2q4t-in-q1
<snip>
N7K#
```

From Example 25-20, it might seem that only CoS values 0 and 1 will be subject to the CoS re-marking policy within the ingress queuing policy (and that CoS values 2–7 will escape this policy, because they are mapped to a nondefault queuing class [in this case, **2q4t-in-q1**]).

However, there is a bit of under-the-hood logic that needs to be explained to better understand how this policy works. Namely, re-marking takes place on *all incoming packets prior to the CoS-to-queue mapping*. This is why re-marking (as part of an ingress queuing policy) is supported only on the default class (that is, so that the re-marking action is applied to *all* ingress packets). It is only *after* re-marking has taken place that CoS-to-queue mapping takes place. Therefore—in this case—it is only after all incoming packets have been re-marked to CoS 0 that these will then be mapped exclusively to the **2q4t-in-q-default** ingress queue, having all the ingress bandwidth and buffers allocated to it.

This untrusted server model—with the same caveats—can be applied similarly to F2-Series modules. The key changes are that the ingress queuing policy needs to be first cloned, then modified, as shown in Example 25-21 based on the F2 default (**default-4q-8e-in-policy)** policy template.

Example 25-21 *Nexus 7000 Untrusted Server Model: F2 Module Example (Based on the Default Policy Template)*

```
 ! This section clones the F2 default ingress policy map
N7K(config)# qos copy policy-map type queuing default-4q-8e-in-policy prefix
UNTRUSTED-
 ! Clones the default ingress queuing policy and prepends a prefix

 ! This section configures the ingress queuing policy map
N7K(config)# policy-map type queuing UNTRUSTED-4q-8e-in
N7K(config-pmap-que)#  class type queuing 2q4t-8e-in-q1
N7K(config-pmap-c-que)#  bandwidth percent 1
N7K(config-pmap-c-que)#  queue-limit percent 1
 ! F2 modules require BW and QL for all queuing classes
 ! Therefore 1% BW and 1% QL represent the minimum provisions
N7K(config-pmap-c-que)# class type queuing 2q4t-8e-in-q-default
N7K(config-pmap-c-que)#  bandwidth percent 99
N7K(config-pmap-c-que)#  queue-limit percent 99
 ! All remaining BW and buffers are assigned to the default class
N7K(config-pmap-c-que)#  set cos 0
 ! CoS is re-marked for all traffic hitting the default class

 ! This section configures the type qos re-marking policy map
N7K(config)# policy-map type qos UNTRUSTED-SERVER-QOS-POLICY
N7K(config-pmap-qos)# class class-default
N7K(config-pmap-c-qos)# set dscp 0
 ! All traffic is re-marked to DSCP 0

 ! Both re-marking policies are applied to the untrusted interface(s)
N7K(config)# interface ethernet 3/45
N7K(config-if)# service-policy type queuing input UNTRUSTED-4q-8e-in
N7K(config-if)# service-policy type qos input UNTRUSTED-SERVER-QOS-POLICY
```

Note This network QoS template requires a bandwidth allocation and queue limit to be assigned to the **2q4t-8e-in-q1** queue. Therefore, to meet this configuration requirement, the minimum allocations are used for each. Effectively, however, this queue is disabled, and all traffic is serviced by the default ingress queue in this model.

You can verify the configuration in Example 25-21 with the following commands:

- **show class-map** [**type** { **qos** |**queuing** }]
- **show policy-map** [**type** { **qos** |**queuing** }]
- **show policy-map interface**

Single-Application Server Marking Model

The Single-Application Server Model is the same as the Untrusted Server Model, except that all traffic is marked to a nonzero CoS/DSCP value.

Note If an ingress queuing policy is to be used to re-mark to a CoS value that will be mapped to a nondefault ingress queue, be sure to assign adequate bandwidth and buffers to the respective ingress queue. Alternatively, you may elect to simplify by only re-marking DSCP via the **type qos** policy and skip the ingress queuing CoS re-marking policy altogether.

Multi-Application Server Classification and Marking Model

In the Multi-Application Server Classification and Marking Model, access lists are used for classification, and traffic is marked to multiple code points. This might be because the application server does not mark traffic or marks traffic to different code points (as compared with the enterprise's QoS model).

Consider a generic example where the network team chooses to explicitly mark traffic from a multimedia application server such that media traffic is identified and marked AF41, signaling traffic is identified and marked CS3, conferencing data (transactional data) flows are identified and marked AF21, and all remaining flows are marked DSCP 0—as shown in Example 25-22.

Example 25-22 *Nexus 7000 M2/F2 Multi-Application Server Classification and Marking Model*

```
 ! This section configures the class maps
N7K(config-cmap-qos)# class-map type qos MEDIA-FLOWS
N7K(config-cmap-qos)#  match access-group name MEDIA-FLOWS
```

```
 ! Media flows are identified via MEDIA-FLOWS ACL
N7K(config-cmap-qos)# class-map type qos SIGNALING
N7K(config-cmap-qos)#  match access-group name SIGNALING
 ! Signaling traffic is identified via SIGNALING ACL
N7K(config-cmap-qos)# class-map type qos CONFERENCING-DATA
N7K(config-cmap-qos)#  match access-group name CONFERENCING-DATA
 ! Conferencing data traffic is identified via CONFERENCING-DATA ACL

 ! This section configures the classification and marking policy map
N7K(config-cmap-qos)# policy-map type qos MULTI-APP-SERVER
N7K(config-pmap-qos)#  class type qos MEDIA-FLOWS
N7K(config-pmap-c-qos)# set dscp af41
 ! Media flows are marked DSCP AF41
N7K(config-pmap-c-qos)# class type qos SIGNALING
N7K(config-pmap-c-qos)#  set dscp cs3
 ! Signaling flows are marked DSCP CS3
N7K(config-pmap-c-qos)# class type qos CONFERENCING-DATA
N7K(config-pmap-c-qos)#  set dscp af21
 ! Conferencing data flows are marked DSCP AF21
N7K(config-pmap-c-qos)# class class-default
N7K(config-pmap-c-qos)#  set dscp default

 ! This section applies the marking policy to the multi-app server int
N7K(config)# interface Ethernet 2/40
N7K(config-if)# service-policy type qos input MULTI-APP-SERVER
 ! Attaches the marking policy to the interface
```

You can verify the configuration in Example 25-22 with the following commands:

- **show class-map** [**type qos**]
- **show policy-map** [**type qos**]
- **show policy-map interface**

Server Policing Model

The Nexus 7000 supports full policing features on the M1, M2, and F2 modules, where policing is performed within the forwarding engine. These modules include support for the following:

- One-rate two-color and two-rate three-color aggregate policing
- Shared policers
- Color-aware policing

Note F1 modules do *not* support any policing functionality

In the Server Policing Model, one or more application classes are metered via one- or two-rate policers, with conforming, exceeding, and (optionally) violating traffic all marked to different DSCP values.

In Example 25-23, a multimedia-conferencing server's media traffic is metered by an RFC 2698 two-rate three-color marking, and AF4 DSCP values are marked according to RFC 2597 markdown rules. The conforming marking can be set directly, but exceeding and violating re-marking values are set via the **cir-markdown-map** and **pir-markdown-map** table-maps (which by default are set to reflect RFC 2597 markdown rules).

Example 25-23 *Nexus 7000 M2/F2 Server Policing Model*

```
 ! This section configures the policing policy map
N7K(config)# policy-map type qos SERVER-POLICING
N7K(config-pmap-qos)# class class-default
N7K(config-pmap-c-qos)# police cir percent 50 pir percent 75 conform set-dscp
-transmit af41 exceed set dscp dscp table cir-markdown-map violate set dscp dscp
table pir-markdown-map
 ! Traffic is policed a CIR of 50% and a PIR of 75%
 ! Conforming traffic is marked AF41
 ! Exceeding traffic is re-marked per the cir-markdown-map
 ! Violating traffic is re-marked per the pir-markdown-map

 ! This section applies the policing policy to the server int
N7K(config)# interface Ethernet 2/40
N7K(config-if)# service-policy type qos input SERVER-POLICING
 ! Attaches the marking policy to the interface
```

You can verify the configuration in Example 25-23 with the following commands:

- **show table-map** [**cir-markdown-map**] (as shown in Example 25-24)
- **show table-map [pir-markdown-map]** (as shown in Example 25-25)
- **show class-map [type qos]**
- **show policy-map [type qos]**
- **show policy-map interface**

Example 25-24 *Verifying CIR Markdown Values:* ***show table-map cir-markdown-map***

```
N7K# show table-map cir-markdown-map
  Table-map cir-markdown-map
```

```
    default copy
    from 10,12 to 12
    from 18,20 to 20
    from 26,28 to 28
    from 34,36 to 36

N7K#
```

As highlighted in Example 25-24, AF41 (DSCP 34)/AF42 (DSCP 36) are re-marked to AF42 (DSCP 36) if found to be *exceeding* the CIR.

Example 25-25 *Verifying PIR Markdown Values:* ***show table-map pir-markdown-map***

```
N7K# show table-map pir-markdown-map
  Table-map pir-markdown-map
    default copy
    from 10,12 to 14
    from 18,20 to 22
    from 26,28 to 30
    from 34,36 to 38

N7K#
```

As highlighted in Example 25-25, AF41 (DSCP 34)/AF42 (DSCP 36) are re-marked to AF43 (DSCP 38) if found to be *violating* the peak information rate (PIR).

DSCP-Mutation Model

It may be that you want to sidestep the entire marking-conflict issues between signaling (CS3/CoS 3), multimedia-streaming (AF3/CoS 3), and FCoE (CoS 3) traffic by simply mutating the DSCP markings for signaling and multimedia-streaming traffic as this traffic enters and exits the data center, such that CS3 and AF3 markings are mapped to/from the CoS 4 range to nonstandard DSCP values on a 1:1 basis. You can do this via a pair of complementary **table-map** commands, as shown in Example 25-26. These policies would be applied on (likely M2) interfaces connecting the data center with the campus in the ingress and egress directions, respectively.

Note If you follow this approach, you have to mark DSCP values for signaling and multimedia streaming on server ports on ingress to these nonstandard DSCP values for DC-sourced flows.

Example 25-26 *Nexus 7000 DSCP Mutation Policies to/from Campus Networks*

```
 ! This section configures the DC-FROM-CAMPUS DSCP-mutation table map
N7K(config)# table-map DC-FROM-CAMPUS-TABLE-MAP
N7K(config-tmap)#  default copy
 ! All DSCP values are copied 1:1 to themselves
 ! with the following modifications made:
N7K(config-tmap)#  from 24 to 33
 ! CS3 (24) is coped to nonstandard DSCP 33 (that will map to CoS 4)
N7K(config-tmap)#  from 26 to 35
 ! AF31 (26) is coped to nonstandard DSCP 35 (that will map to CoS 4)
N7K(config-tmap)#  from 28 to 37
 ! AF32 (28) is coped to nonstandard DSCP 37 (that will map to CoS 4)
N7K(config-tmap)#  from 30 to 39
 ! AF33 (30) is coped to nonstandard DSCP 39 (that will map to CoS 4)

 ! This section configures the DC-TO-CAMPUS DSCP-mutation table map
N7K(config)# table-map DC-TO-CAMPUS-TABLE-MAP
N7K(config-tmap)#  default copy
 ! All DSCP values are copied 1:1 to themselves
 ! with the following modifications made:
N7K(config-tmap)#  from 33 to 24
 ! CS3 (24) is restored from nonstandard DSCP 33
N7K(config-tmap)#  from 35 to 26
 ! AF31 (26) is restored from nonstandard DSCP 35
N7K(config-tmap)#  from 37 to 28
 ! AF32 (28) is restored from nonstandard DSCP 37
N7K(config-tmap)#  from 39 to 30
 ! AF33 (30) is restored from nonstandard DSCP 39

 ! This section configures the DC-FROM-CAMPUS policy map
N7K(config)# policy-map type qos DC-FROM-CAMPUS-POLICY-MAP
N7K(config-pmap-qos)# class class-default
N7K(config-pmap-c-qos)# set dscp dscp table DC-FROM-CAMPUS-TABLE-MAP
 ! All DSCPs from campus are mutated by DC-FROM-CAMPUS-TABLE-MAP

 ! This section configures the DC-TO-CAMPUS policy map
N7K(config-pmap)# policy-map type qos DC-TO-CAMPUS-POLICY-MAP
N7K(config-pmap-qos)# class class-default
N7K(config-pmap-c-qos)# set dscp dscp table DC-TO-CAMPUS-TABLE-MAP
 ! All DSCPs to campus are mutated by DC-TO-CAMPUS-TABLE-MAP

 ! This section attaches the policy maps to DC-to-Campus interface(s)
N7K(config)# interface ethernet 1/32
N7K(config-if)# service-policy type qos input EQ82-DC-FROM-CAMPUS-POLICY-MAP
N7K(config-if)# service-policy type qos output EQ82-DC-TO-CAMPUS-POLICY-MAP
```

You can verify the configuration in Example 25-26 with the following commands:

- **show table-map**
- **show policy-map** [**type qos**]
- **show policy-map interface**
- **show system internal ipqos table-map** (as shown in Examples 25-27)

Example 25-27 *Verifying Table-Mapping Values:* ***show system internal ipqos table-map***

```
N7K# show system internal ipqos table-map
=======================================================
table-map: DC-TO-CAMPUS-TABLE-MAP (len: 27)
        default copy
        ppf node id: 0x4500333
        ref_count: 1
        Bit array: 33,35,37,39
        Values set:
             0  1  2  3  4  5  6  7
             8  9 10 11 12 13 14 15
            16 17 18 19 20 21 22 23
            24 25 26 27 28 29 30 31
            32 24 34 26 36 28 38 30
            40 41 42 43 44 45 46 47
            48 49 50 51 52 53 54 55
            56 57 58 59 60 61 62 63
=======================================================
table-map: DC-FROM-CAMPUS-TABLE-MAP (len: 29)
        default copy
        ppf node id: 0x4500332
        ref_count: 1
        Bit array: 24,26,28,30
        Values set:
             0  1  2  3  4  5  6  7
             8  9 10 11 12 13 14 15
            16 17 18 19 20 21 22 23
            33 25 35 27 37 29 39 31
            32 33 34 35 36 37 38 39
            40 41 42 43 44 45 46 47
            48 49 50 51 52 53 54 55
            56 57 58 59 60 61 62 63
=======================================================
N7K#
```

Example 25-27 lists the DSCP-DSCP mappings in table format, with each set of eight DSCP values listed on a separate row in ascending order. The *from* DSCP values are implied by the respective positions in the table, and the *to* DSCP values are the ones that are explicitly listed in the table. For example, in the first table, it is the fifth row that is of interest (listing all DSCP values that are mapped to CoS 4 because of sharing the 3 MSB of 4): The first column of this row corresponds to DSCP 32, the second corresponds to DSCP 33, and so on. As highlighted, DSCP 33 is not mapped to itself, but to 24 (CS3). Also DSCPs 35, 37, and 39 are mapped to 26 (AF31), 28 (AF32), and 30 (AF33), respectively. In addition, in the second table (DC-FROM-CAMPUS-TABLE-MAP), the fourth row is the one of interest that shows position 24 (CS3) being mapped to 33 and AF31 (26), AF32 (28), and AF33 (30) being mapped to 35, 37, and 39, respectively—as highlighted.

CoPP Design

The Nexus 7000 supports control plane policing. Best-practice designs for control plane policing are covered in Appendix B, "Control Plane Policing."

Note For additional platform-specific details on configuring CoPP on the Nexus 7000, see http://www.cisco.com/en/US/docs/switches/datacenter/sw/6_x/nx-os/security/configuration/guide/b_Cisco_Nexus_7000_NX-OS_Security_Configuration_Guide__Release_6.x_chapter_011001.html.

Summary

This design chapter discussed the best-practice QoS design recommendations for a Cisco Nexus 7000 Series switch in the role of a data center core switch. The discussion began with an overview of the Nexus 7000 system, the differences between M2- and F2-Series modules, the order of QoS operations, and the default trust states. Because of the major differences in architecture and capabilities between the M2- and F2-Series modules, these were approached separately.

Therefore, the next section discussed QoS designs for M2 modules, beginning with a consideration of their hardware architecture and QoS capabilities. Four-class and eight-class queuing models were presented for these M2 modules. In addition, it was recommended in this section to use M-Series modules (that do not support FCoE) to connect to Cisco Unified Communication Managers—which use CoS 3 for signaling (so that F-series modules that do support FCoE can have CoS 3 dedicated to FCoE). Also, OTV was overviewed, along with QoS design recommendations for supporting this service.

Next, F2-Series modules were discussed, overviewing and contrasting their hardware architectures with M2 modules. Network QoS policy templates were analyzed in detail, as was their respective default queuing modules. Examples of enabling the four-class and eight-class queuing models, along with FCoE, were presented in this section.

After this, additional QoS designs common to both series of modules were presented, including trusted/untrusted server models and server marking and policing models. Finally, a DSCP-mutation model was also presented to sidestep the potential marking conflict of signaling and multimedia-streaming and FCoE traffic—all of which map to CoS 3 by default.

Further Reading

Cisco Nexus 7000 Series NX-OS Quality of Service Configuration Guide, Release 6.x: http://www.cisco.com/en/US/docs/switches/datacenter/sw/6_x/nx-os/qos/configuration/guide/nx-os_qos_book.html

Massively Scalable Data Center (MSDC) Design and Implementation Guide: http://www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/MSDC/1.0/MSDC1_C.html

Cisco Overlay Transport Virtualization Technology Introduction and Deployment Considerations: http://www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/DCI/whitepaper/DCI3_OTV_Intro.html

NX-OS and Cisco Nexus Switching: Next-Generation Data Center Architectures (Second Edition) (Fuller, Jansen, McPherson; Cisco Press, 2013)

This page intentionally left blank

Chapter 26

Data Center QoS Design Case Study

With their campus and WLAN QoS network designs based on their strategic eight-class end-to-end QoS model (detailed in Chapter 12, "Strategic QoS Design Case Study," and as illustrated in Figure 12-2), Tifosi's networking team is now ready to adapt and extend these policies into their virtualized multiservice data center.

Tifosi recognizes that they will have to amend their strategic eight-class model (on a per-platform basis) to adapt to the data center environment to include support for the following:

- Virtual machine control protocols
- vMotion
- FCoE

To accommodate these additional classes, yet map efficiently into fixed eight-queue hardware queuing models, Tifosi's networking team has elected to remove the Scavenger class from their queuing models within the data center.

Tifosi's multitier data center network consists of the following:

- Cisco Nexus 1000V switches at the virtual access edge
- Cisco Nexus 5500s with Nexus 2000 Fabric Extenders at the access/aggregation layer
- Cisco Nexus 7000s with both M2 and F2 series modules at the core layer

Tifosi has considered various server models and has settled on the following:

- **Trusted Server model**, which will be applied to most servers (and to all interswitch links)

- **Single-Application Server model**, which will be applied to only a few single-application servers that either do not mark CoS/DSCP or mark these to values that differ from their strategic QoS model
- **Multi-Application Server model**, which again will be applied to a few multi-application servers for the same reasons

They have decided not to implement a blanket untrusted server model because they strictly control access to the data center and only allow approved servers to be hosted therein; therefore, approved servers must align with one (or more) of their strategic application classes.

In addition, they have decided not to implement a policed server model because these only police on an aggregate basis—rather than on a microflow basis, as in the campus. Therefore, random flows may be penalized with markdown, rather than single flows that exceed predefined limits. They plan to revisit this policy in the future, especially if microflow policing becomes available on the Nexus switching family.

Tifosi supports FCoE in their data center—which is assigned by default to CoS 3. Furthermore, they've made the decision to dedicate the CoS 3 virtual lane to FCoE. This separation will be achieved in the following manner:

- Cisco Unified Communication Servers and Multimedia Conferencing Servers (such as their Microsoft Lync servers) and any virtual machines using signaling traffic will mark these flows as DSCP CS3, which will be trusted.
- Nexus 1000V ingress policies will asymmetrically mark signaling traffic to CoS 4 and queue it along with multimedia-conferencing traffic.
- Nexus 5500/2000 queuing policies will classify signaling traffic from servers by DSCP (CS3) and assign these to qos-group 4, which will in turn will map to the video queue (and which will be marked Cos 4 by the **network-qos** policy). In contrast, FCoE traffic will be classified by CoS 3 and serviced in a dedicated system queue.
- Nexus 7000 data center/campus-edge policies will re-mark CS3 from the campus to the nonstandard DSCP value of 33 so that southbound signaling flows will be mapped to CoS 4 once these hit the DCB network on the F2 series modules.

Finally, Tifosi is connecting their two primary data centers via OTV and wants their QoS policies to support this overlay.

Figure 26-1 illustrates Tifosi's combined data center QoS policies.

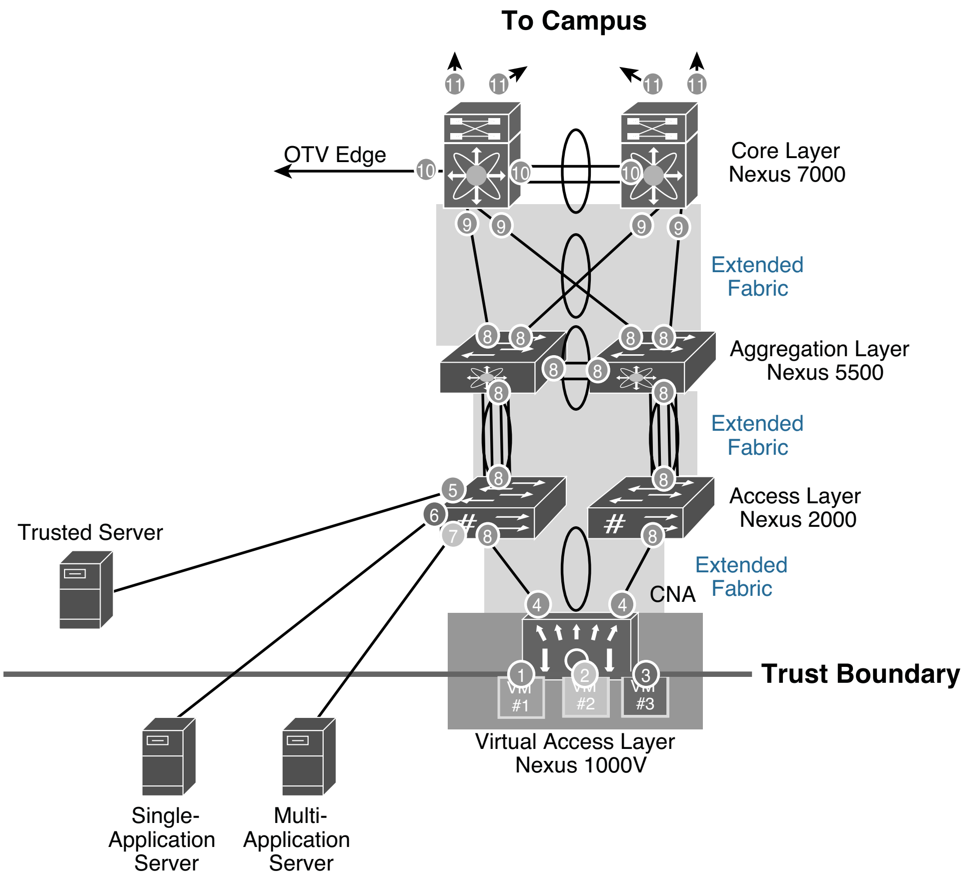


Figure 26-1 *Tifosi Software Case Study: Data Center Network QoS Design*

These data center QoS policies are detailed as follows:

Data Center Virtual Access-Edge QoS Policies

- **Policy 1: Trusted Virtual Machines (Nexus 1000V):** Enable CoS and DSCP trust (default setting)
- **Policy 2: Single-Application Virtual Machine (Nexus 1000V):** Explicitly set CoS/DSCP
- **Policy 3: Multi-Application Virtual Machine (Nexus 1000V):** Classify traffic by ACLs and explicitly set CoS/DSCP values by application class
- **Policy 4: Network-Edge Queuing (Nexus 1000V):** Enable egress protocol- or CoS-based CBWFQ policies for Tifosi's amended eight-class strategic model

Data Center Access/Aggregation Layer QoS Policies

- **Policy 5: Trusted Servers (Nexus 5500/2000):**
  - Enable CoS and DSCP trust (default setting)
  - Enable a no-drop service for FCoE (default)
  - Enable ingress queuing for Tifosi's amended eight-class strategic model
  - Enable egress queuing for Tifosi's amended eight-class strategic model
- **Policy 6: Single-Application Servers (Nexus 5500/2000):**
  - Explicitly set CoS and DSCP
  - Enable a no-drop service for FCoE (default)
  - Enable ingress queuing for Tifosi's amended eight-class strategic model
  - Enable egress queuing for Tifosi's amended eight-class strategic model
- **Policy 7: Multi-Application Servers (Nexus 5500/2000):**
  - Classify traffic by ACLs and explicitly set CoS/DSCP values by application class
  - Enable a no-drop service for FCoE (default)
  - Enable ingress queuing for Tifosi's amended eight-class strategic model
  - Enable egress queuing for Tifosi's amended eight-class strategic model
- **Policy 8: Network-Edge Queuing (Nexus 5500/2000):**
  - Enable CoS and DSCP trust (default setting)
  - Enable a no-drop service for FCoE (default)
  - Enable ingress queuing for Tifosi's amended eight-class strategic model
  - Enable egress queuing for Tifosi's amended eight-class strategic model

Data Center Core-Layer QoS Policies

- **Policy 9: Network-Edge Queuing (Nexus 7000 F2 modules):**
  - Enable CoS and DSCP trust (default setting)
  - Enable a no-drop service for FCoE
  - Enable ingress queuing for Tifosi's amended eight-class strategic model
  - Enable egress queuing for Tifosi's amended eight-class strategic model
- **Policy 10: Network-Edge Queuing (including OTV edges) (Nexus 7000 M2 modules):**
  - Enable CoS and DSCP trust (default setting)

  - Enable ingress queuing for Tifosi's amended eight-class strategic model
  - Enable egress queuing for Tifosi's amended eight-class strategic model
- **Policy 11: DSCP-Mutation Between Data Center and Campus (Nexus 7000 M2 modules):**
  - Map DSCP CS3 from campus to DSCP 33 in the data center
  - Enable ingress queuing for Tifosi's amended eight-class strategic model
  - Enable egress queuing for Tifosi's amended eight-class strategic model

Tifosi Data Center Virtual Access Layer Nexus 1000V QoS Design

Tifosi's data center virtual access layer consists of Cisco Nexus 1000V and includes combinations of the following policies:

- Trusted Virtual Machine Policy
- Single-Application Virtual Machine Policy
- Multi-Application Virtual Machine Policy
- Network-Edge Queuing Policy

The configuration for each of these virtual access layer policies is shown in turn.

Policy 1: Trusted Virtual Machines

The Trusted Virtual Machines model trusts CoS/DSCP markings set by the virtual machine. No explicit policy is required to enable this model because the Nexus 1000V trusts both CoS and DSCP by default. However, noted that the vEthernet interfaces are not usually trunked ports, so only DSCP markings are present.

Policy 2: Single-Application Virtual Machine

The Single-Application Virtual Machine policy is explicitly set CoS/DSCP to all traffic originating from the VM and is shown in Example 26-1.

Example 26-1 *Tifosi Case Study: Virtual Access-Edge Design for Single-Application Virtual Machines on a Nexus 1000V*

```
 ! This section configures a match-any IP ACL
N1KV(config)# ip access-list MATCH-ANY-ACL
N1KV(config-acl)# permit ip any any

 ! This section configures a match-any class map
```

```
N1KV(config)# class-map type qos MATCH-ANY
N1KV(config-cmap-qos)# match access-group name MATCH-ANY-ACL

 ! This section configures the policy map
N1KV(config)# policy-map type qos SINGLE-APP-VM
N1KV(config-pmap-qos)# class MATCH-ANY
N1KV(config-pmap-c-qos)# set dscp af21
N1KV(config-pmap-c-qos)# set cos 2

 ! This section applies the policy to a vEthernet port profile
N1KV(config)# port-profile type vethernet SINGLE-APP-VM
N1KV(config-port-prof)# service-policy type qos input SINGLE-APP-VM
```

You can verify the configuration in Example 26-1 with the following commands:

- **show class-map type qos**
- **show policy-map type qos**
- **show port-profile name**
- **show policy-map interface vethernet** [*interface number*]

Policy 3: Multi-Application Virtual Machine

The Multi-Application Virtual Machine policy classifies traffic by ACLs and explicitly sets CoS and DSCP values by application class, as shown in Example 26-2. In this example, CoS values are set to align with DSCP for the Multimedia Conferencing, Data, and Best Effort classes; however, for the Signaling class, CoS is marked asymmetrically (to CoS 4) to avoid upstream interference with FCoE over the DCB network.

Example 26-2 *Tifosi Case Study: Virtual Access-Edge Design for Multi-Application Virtual Machines on a Nexus 1000V*

```
 ! IP ACLs are assumed to be configured to identify
 ! media, signaling, and data flows
 ! This section configures the class maps
N1KV(config-cmap-qos)# class-map type qos MEDIA-FLOWS
N1KV(config-cmap-qos)#  match access-group name MEDIA-FLOWS
N1KV(config-cmap-qos)# class-map type qos SIGNALING
N1KV(config-cmap-qos)#  match access-group name SIGNALING
N1KV(config-cmap-qos)# class-map type qos CONFERENCING-DATA
N1KV(config-cmap-qos)#  match access-group name CONFERENCING-DATA
 ! This section configures the policy map
N1KV(config)# policy-map type qos MULTI-APP-VM
N1KV(config-pmap-qos)# class type qos MEDIA-FLOWS
N1KV(config-pmap-c-qos)#  set dscp af41
```

```
N1KV(config-pmap-c-qos)#  set cos 4
N1KV(config-pmap-c-qos)# class type qos SIGNALING
N1KV(config-pmap-c-qos)#  set dscp cs3
N1KV(config-pmap-c-qos)#  set cos 4
N1KV(config-pmap-c-qos)# class type qos CONFERENCING-DATA
N1KV(config-pmap-c-qos)#  set dscp af21
N1KV(config-pmap-c-qos)#  set cos 2
N1KV(config-pmap-c-qos)# class class-default
N1KV(config-pmap-c-qos)#  set dscp 0
N1KV(config-pmap-c-qos)#  set cos 0

 ! This section applies the policy to a vEthernet port profile
N1KV(config)# port-profile type vethernet MULTI-APP-VM
N1KV(config-port-prof)# service-policy type qos input MULTI-APP-VM
```

You can verify the configuration in Example 26-2 with the following commands:

- **show class-map type qos**
- **show policy-map type qos**
- **show port-profile name**
- **show policy-map interface vethernet** [*interface number*]

Policy 4: Network-Edge Queuing

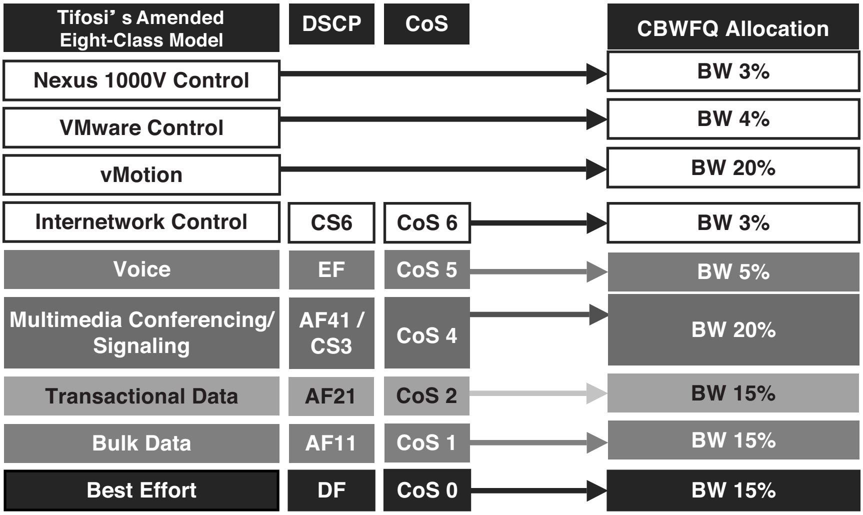
The virtual access layer Network-Edge Queuing policy enables CBWFQ for Tifosi's amended eight-class strategic model to provision for the following:

- Nexus 1000V control traffic
- VM control and management traffic
- vMotion
- Tifosi's strategic application classes

As previously noted, the Scavenger class has been removed from Tifosi's data center queuing models; in addition, in this instance so has the Real-Time-Interactive traffic class, because this applies only to traffic traversing to/from Cisco TelePresence conferencing servers, (which are located on physical servers at the access/aggregation layer).

In this example, signaling traffic (which has been previously marked to CoS 4) will be matched along with multimedia conferencing traffic (on CoS 4) and assigned to the same egress queue (because the Nexus 1000V does not support DSCP-to-queue mapping, at the time of writing).

Figure 26-2 illustrates Tifosi's virtual access Network-Edge Queuing policy configuration.



Tifosi' s Amended Eight-Class Model	DSCP	CoS	CBWFQ Allocation
Nexus 1000V Control			BW 3%
VMware Control			BW 4%
vMotion			BW 20%
Internetwork Control	CS6	CoS 6	BW 3%
Voice	EF	CoS 5	BW 5%
Multimedia Conferencing/ Signaling	AF41 / CS3	CoS 4	BW 20%
Transactional Data	AF21	CoS 2	BW 15%
Bulk Data	AF11	CoS 1	BW 15%
Best Effort	DF	CoS 0	BW 15%

Figure 26-2 *Tifosi Software Case Study: Nexus 1000V Egress CBWFQ Model*

Example 26-3 shows the virtual access layer Network-Edge Queuing policy configuration.

Example 26-3 *Tifosi Case Study: Virtual Access-Layer Network-Edge Queuing on a Nexus 1000V*

```
 ! This section configures the class maps
N1KV(config)# class-map type queuing match-any N1KV-CONTROL
N1KV(config-cmap-que)#  match protocol n1k_control
N1KV(config-cmap-que)#  match protocol n1k_packet
N1KV(config-cmap-que)#  match protocol n1k_mgmt
N1KV(config-cmap-que)# class-map type queuing match-all VM-CONTROL
N1KV(config-cmap-que)#  match protocol vmw_mgmt
N1KV(config-cmap-que)# class-map type queuing match-all VMOTION
N1KV(config-cmap-que)#  match protocol vmw_vmotion
N1KV(config-cmap-que)# class-map type queuing match-all NETWORK-CONTROL
N1KV(config-cmap-que)#  match cos 6
N1KV(config-cmap-que)# class-map type queuing match-all VOICE
N1KV(config-cmap-que)#  match cos 5
N1KV(config-cmap-que)# class-map type queuing match-all MULTIMEDIA-CONFERENCING-
SIGNALING
N1KV(config-cmap-que)#  match cos 4
N1KV(config-cmap-que)# class-map type queuing match-all TRANSACTIONAL-DATA
N1KV(config-cmap-que)#  match cos 2
N1KV(config-cmap-que)# class-map type queuing match-all BULK-DATA
N1KV(config-cmap-que)#  match cos 1
```

```
 ! This section configures the CBWFQ policy map
N1KV(config)# policy-map type queuing N1KV-CBWFQ
N1KV(config-pmap-que)# class type queuing N1KV-CONTROL
N1KV(config-pmap-c-que)#  bandwidth percent 3
N1KV(config-pmap-c-que)# class type queuing VM-CONTROL
N1KV(config-pmap-c-que)#  bandwidth percent 4
N1KV(config-pmap-c-que)# class type queuing VMOTION
N1KV(config-pmap-c-que)#  bandwidth percent 20
N1KV(config-pmap-c-que)# class type queuing NETWORK-CONTROL
N1KV(config-pmap-c-que)#  bandwidth percent 3
N1KV(config-pmap-c-que)# class type queuing VOICE
N1KV(config-pmap-c-que)#  bandwidth percent 5
N1KV(config-pmap-c-que)# class type queuing MULTIMEDIA-CONFERENCING-SIGNALING
N1KV(config-pmap-c-que)#  bandwidth percent 20
N1KV(config-pmap-c-que)# class type queuing TRANSACTIONAL-DATA
N1KV(config-pmap-c-que)#  bandwidth percent 15
N1KV(config-pmap-c-que)# class type queuing BULK-DATA
N1KV(config-pmap-c-que)#  bandwidth percent 15

 ! This section attaches the policy to the uplink port profile
N1KV(config)# port-profile type ethernet UPLINK
N1KV(config-port-prof)# service-policy output N1KV-CBWFQ
```

You can verify the configuration in Example 26-3 with the following commands:

- **show class-map** [**type queuing**]
- **show policy-map** [**type queuing**]
- **show port-profile name**
- **show policy-map interface**
- **module vem** [*module-number*] **execute vemcmd show qos node**
- **module vem** [*module-number*] **execute vemcmd show qos pinst** [**uplink LTL**]

Tifosi Data Center Access/Aggregation Layer Nexus 5500/2000 QoS Design

Tifosi's data center access/aggregation layer consists of Cisco Nexus 5500 switches with Nexus 2000 Fabric Extenders and includes combinations of the following policies:

- Trusted Server
- Single-Application Server

- Multi-Application Server
- Network-Edge Queuing

The configuration for each of these access/aggregation layer policies is shown in turn.

Policy 5: Trusted Server

The Trusted Server policy enables CoS and DSCP trust, and a no-drop service for FCoE (all of which is enabled by default on the Nexus 5500). The only element of the policy needing explicit configuration is the queuing policies, which will be covered in Policy 8 (to minimize redundancy, as these are not only lengthy, but common for Policies 5-8).

Policy 6: Single-Application Server

The Single-Application Server policy, as shown in Example 26-4, explicitly sets the CoS and DSCP markings for all traffic sourced from the server, in addition to enabling a lossless service for FCoE (which is enabled by default) and queuing (which is shown in Policy 8).

Example 26-4 *Tifosi Case Study: Access/Aggregation Edge Design for Single-Application Server on a Nexus 5500/2000*

```
 ! This section configures a match-any IP ACL
N5K(config)# ip access-list MATCH-ANY-ACL
N5K(config-acl)# permit ip any any

 ! This section configures a type qos match-any class map
N5K(config)# class-map type qos MATCH-ANY
N5K(config-cmap-qos)# match access-group name MATCH-ANY-ACL

 ! This section configures the marking policy map
N5K(config)# policy-map type qos SINGLE-APP-SERVER
N5K(config-pmap-qos)# class type qos MATCH-ANY
N5K(config-pmap-c-qos)# set qos-group 2
N5K(config-pmap-c-qos)# set dscp af21

 ! This section attaches the type qos policy to the interface(s)
N5K(config)# interface ethernet 1/15-18
N5K(config-if-range)# service-policy type qos input SINGLE-APP-SERVER

 ! This section configures the network-qos class map
N5K(config)# class-map type network-qos CLASS-2
N5K(config-cmap-nq)# match qos-group 2
```

```
 ! This section configures the network-qos policy map
N5K(config)# policy-map type network-qos NETWORK-QOS-POLICY
N5K(config-pmap-nq)# class type network-qos CLASS-2
N5K(config-pmap-nq)#  set cos 2

 ! This section applies the network-qos policy map to system qos
N5K(config)# system qos
N5K(config-sys-qos)# service-policy type network-qos NETWORK-QOS-POLICY
```

You can verify the configuration in Example 26-5 with the following commands:

- **show class-map type** [**qos** | **network-qos**]
- **show policy-map type** [**qos** | **network-qos**]
- **show policy-map** [**interface** | **system**]

Policy 7: Multi-Application Server

The Multi-Application Server policy, as shown in Example 26-5, classifies applications by ACLs, explicitly sets the CoS and DSCP markings for matching traffic classes, and enables a lossless service for FCoE (by default) and queuing (which is shown in Policy 8).

Note that although signaling traffic is marked to DSCP CS3, its CoS value is asymmetrically marked to 4 so as not to interfere with FCoE traffic.

Example 26-5 *Tifosi Case Study: Access/Aggregation-Edge Design for Single-Application Server on a Nexus 5500/2000*

```
 ! IP ACLs are assumed to be configured to identify
 ! media, signaling and data flows

 ! This section configures the type qos class-maps
N5K(config-cmap-qos)# class-map type qos MEDIA-FLOWS
N5K(config-cmap-qos)#  match access-group name MEDIA-FLOWS
N5K(config-cmap-qos)# class-map type qos SIGNALING
N5K(config-cmap-qos)#  match access-group name SIGNALING
N5K(config-cmap-qos)# class-map type qos TRANSACTIONAL-DATA
N5K(config-cmap-qos)#  match access-group name TRANSACTIONAL-DATA

 ! This section configures the type qos marking policy map
N5K(config)# policy-map type qos MULTI-APP-SERVER
N5K(config-pmap-qos)#  class type qos MEDIA-FLOWS
N5K(config-pmap-c-qos)#  set qos-group 4
N5K(config-pmap-c-qos)#  set dscp af41
N5K(config-pmap-c-qos)# class type qos SIGNALING
```

```
N5K(config-pmap-c-qos)#  set qos-group 3
N5K(config-pmap-c-qos)#  set dscp cs3
N5K(config-pmap-c-qos)# class TRANSACTIONAL-DATA
N5K(config-pmap-c-qos)#  set qos-group 2
N5K(config-pmap-c-qos)#  set dscp af21
N5K(config-pmap-c-qos)# class type qos class-default
N5K(config-pmap-c-qos)#  set dscp 0

 ! This section attaches the policy map to the interface(s)
N5K(config)# interface ethernet 1/1-4
N5K(config-if-range)# service-policy type qos input MULTI-APP-SERVER

 ! This section configures the network-qos class-maps
N5K(config-cmap-nq)# class-map type network-qos CLASS-2
N5K(config-cmap-nq)#  match qos-group 2
N5K(config-cmap-nq)# class-map type network-qos CLASS-3
N5K(config-cmap-nq)#  match qos-group 3
N5K(config-cmap-nq)# class-map type network-qos CLASS-4
N5K(config-cmap-nq)#  match qos-group 4

 ! This section configures the network-qos policy map
N5K(config)# policy-map type network-qos NETWORK-QOS-POLICY
N5K(config-pmap-nq)# class type network-qos CLASS-2
N5K(config-pmap-nq-c)#  set cos 2
N5K(config-pmap-nq-c)# class type network-qos CLASS-3
N5K(config-pmap-nq-c)#  set cos 4
N5K(config-pmap-nq-c)# class type network-qos CLASS-4
N5K(config-pmap-nq-c)#  set cos 4

 ! This section applies the network-qos policy map to system qos
N5K(config)# system qos
N5K(config-sys-qos)# service-policy type network-qos NETWORK-QOS-POLICY
```

You can verify the configuration in Example 26-5 with the following commands:

- **show class-map type** [**qos** | **network-qos**]
- **show policy-map type** [**qos** | **network-qos**]
- **show policy-map** [**interface** | **system**]

Policy 8: Network-Edge Queuing Policy

The access/aggregation Network-Edge Queuing policy enables ingress and egress queuing for network control traffic, no-drop FCoE traffic, vMotion traffic, and Tifosi's strategic application classes (less the Scavenger class). Figure 26-3 illustrates the resulting queuing policy.

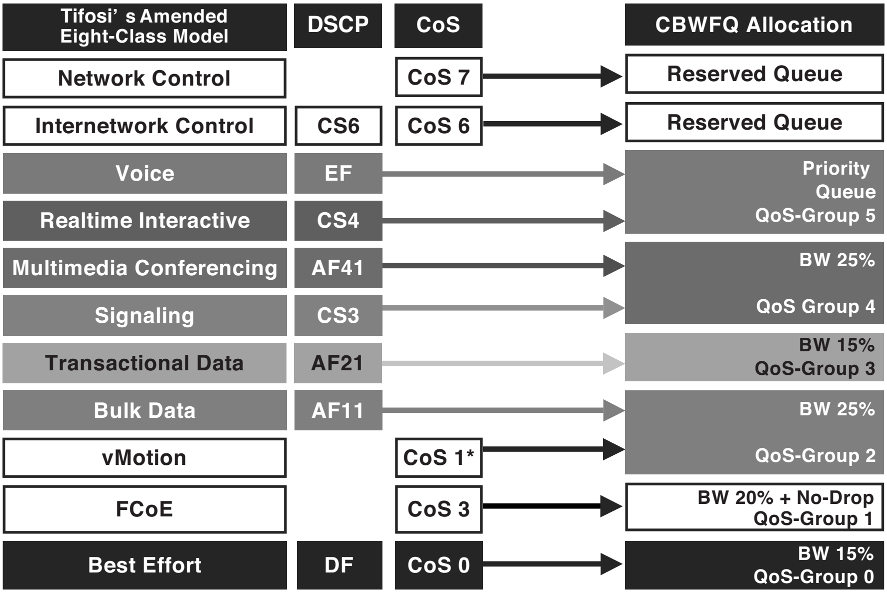


Figure 26-3 *Tifosi Software Case Study: Nexus 5500 Ingress/Egress Queuing Model*

In this example, signaling traffic is matched on CS3 (for northbound signaling) or DSCP 33 (for southbound signaling) and assigned to the video/multimedia-conferencing queue.

In addition, all application traffic classes are marked to their aligned CoS values by the **network-qos** policy, with two exceptions:

- Real-time interactive traffic (CS4) is marked to CoS 5 so that it will receive priority service.
- Signaling traffic (CS3) is marked to CoS 4 so that it will not interfere with upstream policies for FCoE.

Note It has been assumed that vMotion traffic has been previously marked to CoS 1 via a Cisco UCS Manager QoS policy.

Example 26-6 shows the access/aggregation Network-Edge Queuing policy configuration.

Example 26-6 *Tifosi Case Study: Access/Aggregation Layer Network-Edge Queuing on a Nexus 5500*

```
 ! This section configures the type qos class maps
N5K(config-cmap-qos)# class-map type qos match-any VOICE
N5K(config-cmap-qos)#  match dscp ef
N5K(config-cmap-qos)# class-map type qos match-any REALTIME-INTERACTIVE
N5K(config-cmap-qos)#  match dscp cs4
N5K(config-cmap-qos)# class-map type qos match-any MULTIMEDIA-CONFERENCING
N5K(config-cmap-qos)#  match dscp af41 af42 af43
N5K(config-cmap-qos)# class-map type qos match-any SIGNALING
N5K(config-cmap-qos)#  match dscp cs3 33
N5K(config-cmap-qos)# class-map type qos match-any TRANSACTINOAL-DATA
N5K(config-cmap-qos)#  match dscp af21 af22 af23
N5K(config-cmap-qos)# class-map type qos match-any BULK-DATA
N5K(config-cmap-qos)#  match dscp af11 af12 af13
N5K(config-cmap-qos)# class-map type qos match-any VMOTION
N5K(config-cmap-qos)#  match cos 1

 ! This section configures the type qos policy map
N5K(config)# policy-map type qos MODIFIED-EIGHT-CLASS-QOS-POLICY
N5K(config-pmap-qos)# class VOICE
N5K(config-pmap-c-qos)#  set qos-group 5
N5K(config-pmap-c-qos)# class REALTIME-INTERACTIVE
N5K(config-pmap-c-qos)#  set qos-group 5
N5K(config-pmap-c-qos)# class MULTIMEDIA-CONFERENCING
N5K(config-pmap-c-qos)#  set qos-group 4
N5K(config-pmap-c-qos)# class SIGNALING
N5K(config-pmap-c-qos)#  set qos-group 4
N5K(config-pmap-c-qos)# class TRANSACTIONAL-DATA
N5K(config-pmap-c-qos)#  set qos-group 3
N5K(config-pmap-c-qos)# class BULK-DATA
N5K(config-pmap-c-qos)#  set qos-group 2
N5K(config-pmap-c-qos)# class VMOTION
N5K(config-pmap-c-qos)#  set qos-group 2
N5K(config-pmap-c-qos)# class class-fcoe
N5K(config-pmap-c-qos)#  set qos-group 1

 ! This section configures the network-qos class maps
N5K(config-cmap-nq)# class-map type network-qos PRIORITY-CLASS
N5K(config-cmap-nq)#  match qos-group 5
N5K(config-cmap-nq)# class-map type network-qos VIDEO-CLASS
N5K(config-cmap-nq)#  match qos-group 4
N5K(config-cmap-nq)# class-map type network-qos TRANS-DATA-CLASS
N5K(config-cmap-nq)#  match qos-group 3
```

```
N5K(config-cmap-nq)# class-map type network-qos BULK-DATA-CLASS
N5K(config-cmap-nq)#  match qos-group 2

 ! This section configures the network-qos policy map
N5K(config)# policy-map type network-qos MODIFIED-EIGHT-CLASS-NETWORK-QOS-POLICY
N5K(config-pmap-nq)# class type network-qos PRIORITY-CLASS
N5K(config-pmap-nq-c)#  set cos 5
N5K(config-pmap-nq-c)# class type network-qos VIDEO-CLASS
N5K(config-pmap-nq-c)#  set cos 4
N5K(config-pmap-nq-c)# class type network-qos TRANS-DATA-CLASS
N5K(config-pmap-nq-c)#  set cos 2
N5K(config-pmap-nq-c)# class type network-qos BULK-DATA-CLASS
N5K(config-pmap-nq-c)#  set cos 1
N5K(config-pmap-nq-c)# class type network-qos class-fcoe
N5K(config-pmap-nq-c)#  pause no-drop
N5K(config-pmap-nq-c)#  mtu 2158

! This section configures the queuing class maps
N5K(config-cmap-que)# class-map type queuing PRIORITY-QUEUE
N5K(config-cmap-que)#  match qos-group 5
N5K(config-cmap-que)# class-map type queuing VIDEO-QUEUE
N5K(config-cmap-que)#  match qos-group 4
N5K(config-cmap-que)# class-map type queuing TRANS-DATA-QUEUE
N5K(config-cmap-que)#  match qos-group 3
N5K(config-cmap-que)# class-map type queuing BULK-DATA-QUEUE
N5K(config-cmap-que)#  match qos-group 2

 ! This section configures the queuing policy map
N5K(config)# policy-map type queuing MODIFIED-EIGHT-CLASS-QUEUING-POLICY
N5K(config-pmap-que)# class type queuing PRIORITY-QUEUE
N5K(config-pmap-c-que)#  priority
N5K(config-pmap-c-que)# class type queuing VIDEO-QUEUE
N5K(config-pmap-c-que)#  bandwidth percent 25
N5K(config-pmap-c-que)# class type queuing TRANS-DATA-QUEUE
N5K(config-pmap-c-que)#  bandwidth percent 15
N5K(config-pmap-c-que)# class type queuing BULK-DATA-QUEUE
N5K(config-pmap-c-que)#  bandwidth percent 25
N5K(config-pmap-c-que)# class type queuing class-fcoe
N5K(config-pmap-c-que)#  bandwidth percent 20
N5K(config-pmap-c-que)# class type queuing class-default
N5K(config-pmap-c-que)#  bandwidth percent 15

 ! This section applies the network-qos and queuing policies
 ! to system qos
N5K(config)# system qos
```

```
N5K(config-sys-qos)# service-policy type network-qos MODIFIED-EIGHT-CLASS-NETWORK-
QOS-POLICY
N5K(config-sys-qos)# service-policy type qos input MODIFIED-EIGHT-CLASS-QUEUING-
POLICY
N5K(config-sys-qos)# service-policy type queuing output MODIFIED-EIGHT-CLASS-
QUEUING-POLICY
```

You can verify the configuration in Example 26-6 with the following commands:

- **show class-map type** [**qos** | **network-qos** | **queuing**]
- **show policy-map type** [**qos** | **network-qos** | **queuing**]
- **show policy-map** [**interface** | **system**]

Tifosi Data Center Core Layer Nexus 7000 QoS Design

Tifosi's data core layer consists of Cisco Nexus 7000 switches with F2 modules extending the DCB network to the core layer and with M2 modules operating at Layer 3 to provide interconnections to the campus and to OTV peers. The core layer includes the following QoS policies:

- Network-Edge Queuing (F2 Modules)
- Network-Edge Queuing (M2 Modules)
- DSCP Mutation for Signaling Traffic Between Campus and Data Center

The configuration for each of these core layer policies is shown in turn.

Policy 9: Network-Edge Queuing (F2 Modules)

In the Network-Edge Queuing policy for F2 modules, CoS and DSCP trust must be enabled (which is done by default). In addition, a no-drop service for FCoE needs to also be enabled, which can be done by applying the **default-nq-7e-policy** template to **system qos**, as shown previously in Example 26-6. Figure 26-4 illustrates the application-to-queue mappings applied by this policy template.

Note Figure 26-4 assumes that signaling traffic has been marked to CoS 4 (as has been shown in previous examples) and that vMotion has been previously marked to CoS 1 (such as by a Cisco UCS QoS policy).

Example 26-7 shows not only the application of the **nq-7e** policy template to **system qos**, but also some optional modifications of the default queuing policies (specifically, the tuning of the ingress buffers of the drop queues to more equitably distribute these).

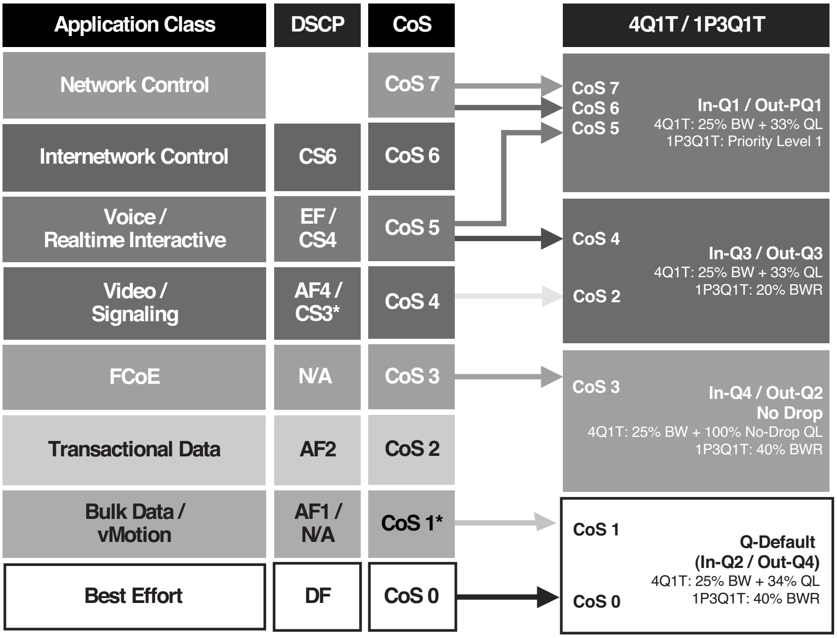


Figure 26-4 *Tifosi Software Case Study: Cisco Nexus 7000 F2 (nq-7e: 4Q1T Ingress / 1P3Q1T Egress) Queuing Model*

Example 26-7 *Tifosi Case Study: Core Network-Edge Queuing on a Nexus 7000 F2 Module (nq-7e 4Q1T Ingress / 1P3Q1T Egress) Model*

```
 ! This section applies the default-nq-7e-policy to system qos
N7K(config)# system qos
N7K(config-sys-qos)# service-policy type network-qos default-nq-7e-policy

 ! This section clones the default 7e policy template
N7K# qos copy policy-map type queuing default-4q-7e-in-policy prefix MODIFIED-

 ! This section modifies the cloned ingress queuing policy
N7K(config)# policy-map type queuing MODIFIED-4q-7e-drop-in
N7K(config-pmap-que)# class type queuing 4q4t-7e-in-q1
N7K(config-pmap-c-que)# queue-limit percent 33
N7K(config-pmap-c-que)# class type queuing 4q4t-7e-in-q-default
N7K(config-pmap-c-que)# queue-limit percent 34
N7K(config-pmap-c-que)# class type queuing 4q4t-7e-in-q3
N7K(config-pmap-c-que)# queue-limit percent 33

 ! This section applies the modified queuing policy to the interface(s)
N7K(config)# interface Ethernet3/1-8
N7K(config-if-range)# service-policy type queuing input MODIFIED-4q-7e-in
```

You can verify the configuration in Example 26-7 with the following commands:

- **show class-map type network-qos** [**c-nq-7e-drop** | **c-nq-7e-ndrop-fcoe**]
- **show policy-map type network-qos** [**default-nq-7e-policy**]
- **show policy-map system**
- **show policy-map type queuing** [**MODIFIED-4q-7e-in** | **MODIFIED-4q-7e-drop-in** | **MODIFIED-4q-7e-ndrop-in**] (assuming the prefix used is MODIFIED-)
- **show policy-map interface**

Policy 10: Network-Edge Queuing (M2 Modules)

In the Network-Edge Queuing policy for M2 modules CoS and DSCP trust must be enabled (which is done by default). No provisioning is needed for vMotion (because these modules are assumed to be operating at Layer 3), nor is any needed for FCoE (which is not supported on these modules anyway). Therefore, there is no longer any overlap for CoS 3. Furthermore, because signaling traffic has been mapped to CoS 4 (in the northbound direction at the virtual access or access/aggregation layers and in the southbound direction at the DC campus interconnect, as shown in Policy 11), there is no traffic in CoS 3 at all in Tifosi's adaption of this model. Therefore, any bandwidth and buffers assigned to this queue are reallocated to the CoS 4 queue (where signaling traffic will be serviced), as shown in Figure 26-5.

In addition, because this service policy explicitly protects network and internetwork control traffic via a dedicated class, it will also support the OTV interconnection.

Example 26-8 shows the configuration for Tifosi's M2 ingress and egress queuing model.

Example 26-8 *Tifosi Case Study: Core Network-Edge Queuing on a Nexus 7000 M2 Module (8Q2T Ingress / 1P7Q4T Egress) Model*

```
! This section configures the ingress queuing class maps
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q1
N7K(config-cmap-que)#  match cos 5
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q2
N7K(config-cmap-que)#  match cos 7
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q3
N7K(config-cmap-que)#  match cos 6
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q4
N7K(config-cmap-que)#  match cos 4
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q5
N7K(config-cmap-que)#  match cos 3
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q6
N7K(config-cmap-que)#  match cos 2
N7K(config-cmap-que)# class-map type queuing match-any 8q2t-in-q7
N7K(config-cmap-que)#  match cos 1
```

```
 ! This section configures the egress queuing class maps
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-pq1
N7K(config-cmap-que)#  match cos 5
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q2
N7K(config-cmap-que)#  match cos 7
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q3
N7K(config-cmap-que)#  match cos 6
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q4
N7K(config-cmap-que)#  match cos 4
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q5
N7K(config-cmap-que)#  match cos 3
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q6
N7K(config-cmap-que)#  match cos 2
N7K(config-cmap-que)# class-map type queuing match-any 1p7q4t-out-q7
N7K(config-cmap-que)#  match cos 1

 ! This section configures the 8Q2T ingress queuing policy map
N7K(config)# policy-map type queuing 8Q2T-INGRESS
N7K(config-pmap-que)#  class type queuing 8q2t-in-q1
N7K(config-pmap-c-que)#  bandwidth percent 10
N7K(config-pmap-c-que)#  queue-limit percent 10
N7K(config-pmap-c-que)# class type queuing 8q2t-in-q2
N7K(config-pmap-c-que)#  bandwidth percent 5
N7K(config-pmap-c-que)#  queue-limit percent 5
N7K(config-pmap-c-que)# class type queuing 8q2t-in-q3
N7K(config-pmap-c-que)#  bandwidth percent 5
N7K(config-pmap-c-que)#  queue-limit percent 5
N7K(config-pmap-c-que)# class type queuing 8q2t-in-q4
N7K(config-pmap-c-que)#  bandwidth percent 35
N7K(config-pmap-c-que)#  queue-limit percent 35
N7K(config-pmap-c-que)#  random-detect cos-based
N7K(config-pmap-c-que)#  random-detect cos 4 minimum-threshold percent 80 maximum-
threshold percent 100
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q5
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q6
N7K(config-pmap-c-que)#   bandwidth percent 10
N7K(config-pmap-c-que)#   queue-limit percent 10
N7K(config-pmap-c-que)#   random-detect cos-based
N7K(config-pmap-c-que)#   random-detect cos 2 minimum-threshold percent 80 maximum-
threshold percent 100
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q7
N7K(config-pmap-c-que)#   bandwidth percent 10
N7K(config-pmap-c-que)#   queue-limit percent 10
N7K(config-pmap-c-que)#   random-detect cos-based
N7K(config-pmap-c-que)#   random-detect cos 1 minimum-threshold percent 80 maximum-
threshold percent 100
N7K(config-pmap-c-que)#  class type queuing 8q2t-in-q-default
```

```
N7K(config-pmap-c-que)#   bandwidth percent 25
N7K(config-pmap-c-que)#   queue-limit percent 25
N7K(config-pmap-c-que)#   random-detect cos-based
N7K(config-pmap-c-que)#   random-detect cos 0 minimum-threshold percent 80 maximum-
threshold percent 100

! This section configures the 1P7Q4T egress queuing policy map
N7K(config)# policy-map type queuing 1P7Q4T-EGRESS
N7K(config-pmap-que)# class type queuing 1p7q4t-out-pq1
N7K(config-pmap-c-que)#  priority
N7K(config-pmap-c-que)#  queue-limit percent 10
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q2
N7K(config-pmap-c-que)#  bandwidth remaining percent 5
N7K(config-pmap-c-que)#  queue-limit percent 5
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q3
N7K(config-pmap-c-que)#  bandwidth remaining percent 5
N7K(config-pmap-c-que)#  queue-limit percent 5
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q4
N7K(config-pmap-c-que)#  bandwidth remaining percent 35
N7K(config-pmap-c-que)#  queue-limit percent 35
N7K(config-pmap-c-que)#  random-detect cos-based
N7K(config-pmap-c-que)#  random-detect cos 4 minimum-threshold percent 80 maximum-
threshold percent 100
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q5
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q6
N7K(config-pmap-c-que)#  bandwidth remaining percent 10
N7K(config-pmap-c-que)#  queue-limit percent 10
N7K(config-pmap-c-que)#  random-detect cos-based
N7K(config-pmap-c-que)#  random-detect cos 2 minimum-threshold percent 80 maximum-
threshold percent 100
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q7
N7K(config-pmap-c-que)#  bandwidth remaining percent 10
N7K(config-pmap-c-que)#  queue-limit percent 10
N7K(config-pmap-c-que)#  random-detect cos-based
N7K(config-pmap-c-que)#  random-detect cos 1 minimum-threshold percent 80 maximum-
threshold percent 100
N7K(config-pmap-c-que)# class type queuing 1p7q4t-out-q-default
N7K(config-pmap-c-que)#  bandwidth remaining percent 35
N7K(config-pmap-c-que)#  queue-limit percent 25
N7K(config-pmap-c-que)#  random-detect cos-based
N7K(config-pmap-c-que)#  random-detect cos 0 minimum-threshold percent 80 maximum-
threshold percent 100

 ! This section applies the queuing policies to the interface(s)
N7K(config)#  interface Ethernet 1/1-4
N7K(config-if-range)#  service-policy type queuing input 8Q2T-INGRESS
N7K(config-if-range)#  service-policy type queuing output 1P7Q4T-EGRESS
```

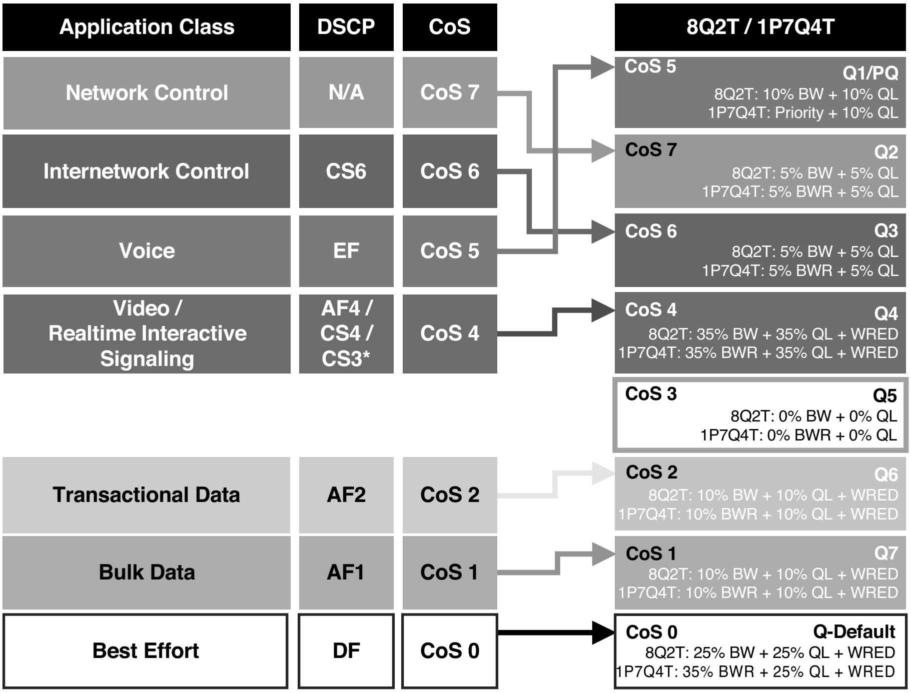


Figure 26-5 *Tifosi Software Case Study: Cisco Nexus 7000 M2 (8Q2T Ingress/1P7Q4T Egress) Queuing Model*

You can verify the configuration in Example 26-8 with the following commands:

- **show class-map** [**type queuing**]
- **show policy-map** [**type queuing**]
- **show policy-map interface**

Policy 11: DSCP Mutation for Signaling Traffic Between Campus and Data Center

Tifosi has elected to separate signaling traffic from FCoE traffic over their DCB network; therefore, they have chosen to map southbound signaling traffic received from the campus to the nonstandard DSCP value of 33 (which will map by default to CoS 4), as shown in Example 26-9.

Example 26-9 *Tifosi Case Study: DSCP Mutation for Signaling Traffic Between Campus and Data Center*

```
 ! This section configures the type qos class map
N7K(config)# class-map type qos match-any CAMPUS-SIGNALING
```

```
N7K(config-cmap-qos)#  match dscp cs3
 ! This section configures the type qos re-marking policy map
N7K(config-cmap-qos)# policy-map type qos CAMPUS-CS3-TO-DC-DSCP33
N7K(config-pmap-qos)#  class CAMPUS-SIGNALING
N7K(config-pmap-c-qos)# set dscp 33

 ! This section applies the re-marking policies to the int(s)
 ! And the queuing policies to the interface(s)
N7K(config)#  interface Ethernet 1/5-6
N7K(config-if-range)#  service-policy type qos input CAMPUS-CS3-TO-DC-DSCP33
N7K(config-if-range)#  service-policy type queuing input 8Q2T-INGRESS
N7K(config-if-range)#  service-policy type queuing output 1P7Q4T-EGRESS
```

You can verify the configuration in Example 26-9 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Summary

This chapter continued the case study example of Tifosi Software and applied their strategic eight-class end-to-end QoS model to the data center environment, which required amending it to include data center protocols such as VM control protocols, vMotion, and FCoE (which requires a lossless service).

Virtual access layer policies were presented for the Nexus 1000V switch for trusted VMs, single-application VMs and multi-application VMs, and a CBWFQ egress (uplink) queuing model.

In addition, access/aggregation layer policies were detailed for the Nexus 5500 with 2000 series FEX modules for the same ingress models (Trusted Server, Single-Application Server, and Multi-Application Server), which are combined with **network-qos** policies and ingress and egress queuing policies.

Afterward, core layer policies for the Nexus 7000 were documented to illustrate queuing configurations on F2 modules and M2 modules.

Furthermore, woven throughout these designs were tactics to achieve separation of signaling traffic from FCoE traffic over the DCB network, so as to dedicate the CoS 3 virtual lane to FCoE. These tactics included asymmetrical DSCP/CoS marking, DSCP-to-queue mapping (where supported), and DSCP mutation at the campus/DC interconnect.

Further Reading

Nexus 1000V QoS Configuration Guide, NX-OS Release 4.2(1)SV2(1.1): http://www.cisco.com/en/US/docs/switches/datacenter/nexus1000/sw/4_2_1_s_v_2_1_1/qos/configuration/guide/b_Quality_of_Service_Configuration_Guide_Release_4_2_1SV2_1_1.html

Nexus 5500 QoS Configuration Guide, NX-OS Release 6.0(2)N1(2): http://www.cisco.com/en/US/docs/switches/datacenter/nexus5500/sw/qos/602_N1_2/b_5500_QoS_Config_602N12.html

Cisco Nexus 7000 Series NX-OS Quality of Service Configuration Guide, Release 6.x: http://www.cisco.com/en/US/docs/switches/datacenter/sw/6_x/nx-os/qos/configuration/guide/nx-os_qos_book.html

This page intentionally left blank

Chapter 27

WAN and Branch QoS Design Considerations and Recommendations

QoS has a dual-role in rich-media WAN and branch networks:

- To manage packet loss (and jitter) by queuing policies
- To enhance classification granularity by leveraging deep packet inspection engines.

The effect of packet loss on real-time applications has already been discussed at length. However, packet jitter (which is most apparent at the WAN/branch edge because of the dramatic downshifts in link speeds) can result in similar effects on voice and video quality.

Jitter is the variation in delay from one packet of a given flow to another of the same flow. Excessive packet jitter can adversely affect the quality of real-time applications. In some situations, the effect of an excessively delayed packet is just as bad as a dropped packet (which it ultimately becomes if the receiver's de-jitter buffer is exceeded). Jitter is explained in detail in RFC 3393, *IP Packet Delay Variation Metric for IP Performance Metrics*.

Both packet loss and packet jitter can be mitigated and managed by deploying QoS at the WAN and branch.

In addition, because Cisco IOS offers increased classification granularity—thanks to tools such as Medianet application metadata and application visibility and control Network Based Application Recognition 2 (AVC NBAR2) classification—you can use these on the WAN and branch LAN edges to enhance application classification policies.

Therefore, several of the strategic QoS design principles (discussed in Chapter 11, "QoS Design Principles and Strategies") apply to WAN and branch QoS designs, including the following:

- **Always perform QoS in hardware rather than software when a choice exists:** Many classification, marking, and policing policies can be deployed in hardware on campus/wireless-edge platforms, which lightens the QoS processing load required of (software-based) Cisco IOS routers (like the Cisco ISR G2).

- **Classify and mark applications as close to their sources as technically and administratively feasible:** Some classification policies may require Layer 7 awareness and, therefore, may not be possible to perform on campus and branch Catalyst switches. Therefore, the ingress edge of the WAN or branch router may be the closest technically feasible point to perform such detailed classification.
- **Enable queuing policies at every node where the potential for congestion exists:** Enable queuing policies on all WAN edges according to strategic business priorities. In addition, in some rare cases, LAN edges might also require queuing policies to be deployed on them.
- **(Optional) Protect the control plane and data plane by enabling control plane policing** to harden the WAN and branch network infrastructure, and data plane policing (scavenger class QoS) to mitigate and constrain network attacks.

Before these strategic QoS design principles can be translated into platform-specific configuration recommendations, you need to consider a few additional WAN and branch-specific factors:

- WAN and branch architectures
- Hardware versus IOS Software QoS
- Latency and jitter
- Tx-Ring
- CBWFQ
- LLQ
- WRED
- RSVP
- Medianet
- AVC
- AutoQoS
- Control plane policing (CPP)
- Link types and speeds
- WAN and branch QoS models
- WAN and branch interface QoS roles

Each of these WAN and branch-specific considerations is discussed in turn.

WAN and Branch Architectures

Extending an enterprise campus network over a wide area to interconnect with other campus/branch networks usually requires the deployment of two types of routers:

- WAN aggregation routers
- Branch routers

WAN aggregation routers serve to connect large campus networks to the WAN/VPN, and branch routers serve to connect smaller branch LANs to the WAN/VPN. Figure 27-1 illustrates these WAN two main types of WAN routers.

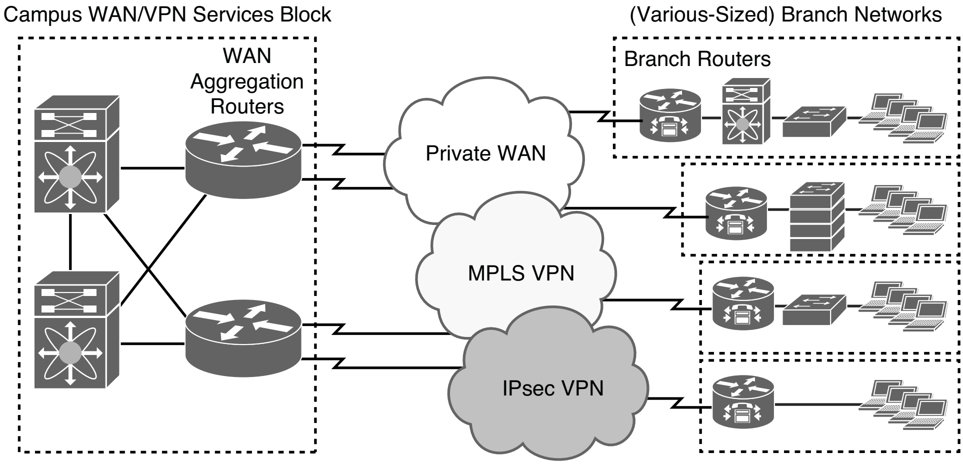


Figure 27-1 *WAN and Branch Architectures and Router Roles*

In this section, QoS designs are presented for both WAN aggregation routers and for branch routers.

Note This part of the book ("WAN and Branch QoS Design") lays a foundation for wide-area designs and therefore features private WAN media (Serial/ATM/POS). The following part (Part VII, "MPLS VPN Design") features Ethernet as a WAN/MAN access media. And the final part (Part VIII, "IPsec VPN Design") builds further and can be applied over any access media.

Hardware Versus IOS Software QoS

Recommendations:

- Be aware of marginal CPU overhead required for deploying QoS policies in IOS routers.
- Choose router platforms that match performance requirements.

Unlike Cisco Catalyst switches used within the Medianet campus (which perform quality of service (QoS) exclusively in hardware), most Cisco routers perform QoS operations in IOS Software (such as ISR/ISR G2s); although some platforms (such as the Cisco Catalyst 6500/7600 and Cisco ASRs) perform QoS in a hybrid mix of software and hardware.

IOS QoS has several advantages, including the following:

- **Cross-platform consistency in QoS features:** For example, rather than having hardware-specific queuing structures on a per-platform or per-linecard basis (as is the case for Cisco Catalyst switches), standard software queuing features, like low-latency queuing (LLQ) and class-based weighted fair queuing (CBWFQ) can be used across WAN and branch router platforms.
- **Consistent QoS configuration syntax:** The configuration syntax for IOS QoS is Modular QoS command-line interface (MQC) is (with very few exceptions) identical across these WAN and branch router platforms.
- **Richer QoS features:** Many IOS QoS features, such as NBAR/NBAR2 and Hierarchical QoS (HQoS), are not available on most Catalyst hardware platforms.

However, despite these advantages, you want to keep in mind one main caveat with respect to software QoS policies: These require a marginal CPU load to process. How much of an incremental CPU load each QoS policy requires depends on many factors, including the router platform (including any hardware-acceleration features), link speeds, policy complexity, traffic mix, and packet rates/sizes.

Table 27-1 provides a generic scalability guideline for the WAN and branch routers discussed (as per their product data sheets). However, it cannot be overemphasized that there is no substitute for testing to ensure that a given router fits the required role. Cisco recommends that WAN or branch routers not consume more than 75 percent of their CPU resources during normal operation (including all QoS policy operations), in order that the CPU always has cycles available to efficiently respond to network events. If a given WAN aggregation or branch router exceeds this 75 percent CPU utilization target during normal operation, it may be advisable to consider a platform upgrade.

Table 27-1 *Cisco WAN and Branch Router Platform Data Sheet Performance Capacities*

Cisco Router Platform	Performance Capacity
Cisco ASR1000-ESP100	100 Gbps
Cisco ASR1000-ESP40	40 Gbps
Cisco ASR1000-ESP20	20 Gbps
Cisco ASR1000-ESP10/10-N	10 Gbps
Cisco ASR1000-ESP5	5 Gbps
Cisco ASR1002-X	5 Gbps (upgradeable)
Cisco ASR1002	2.5 Gbps
Cisco ASR1001	2.5 Gbps
Cisco 3945E ISR G2	350 Mbps
Cisco 3925E ISR G2	250 Mbps
Cisco 3945 ISR G2	150 Mbps
Cisco 3925 ISR G2	100 Mbps
Cisco 2951 ISR G2	75 Mbps
Cisco 2921 ISR G2	50 Mbps
Cisco 2911 ISR G2	35 Mbps
Cisco 2901 ISR G2	25 Mbps
Cisco 1941 ISR G2	25 Mbps
Cisco 1921 ISR G2	15 Mbps

Latency and Jitter

Recommendations:

- Choose service provider paths to target 150 ms for one-way latency. If this target is unable to be met, 200 ms is also generally acceptable.
- Understand that only the queuing delay is manageable by WAN QoS policies.

Real-time applications have fixed latency budgets. For example, the ITU G.114 specification sets the target for one-way latency of real-time voice/video conversations to be 150 ms. To meet such targets, it is important for administrators to understand the components of network latency so that they know which factors they can and cannot control with network and QoS design. Network latency can be broken down into fixed and variable components:

- Serialization (fixed)
- Propagation (fixed)
- Queuing (variable)

Serialization refers to the time it takes to convert a Layer 2 frame into Layer 1 electrical or optical pulses onto the transmission media. Therefore, serialization delay is fixed and is a function of the line rate (that is, the clock speed of the link). For example, a T1 circuit (1.544 Mbps) requires about 8 ms to serialize a 1500-byte Ethernet frame onto the wire, whereas an OC-192/STM-64 circuit (9.953 Gbps) requires just 1.2 microseconds to serialize the same frame.

Usually, the most significant network factor in meeting the latency targets for over the WAN is propagation delay, which can account for over 95% of the network latency time budget. Propagation delay is also a fixed component and is a function of the physical distance that the signals have to travel between the originating endpoint and the receiving endpoint. The gating factor for propagation delay is the speed of light: which is 300,000 km/s or 186,000 miles per second in a vacuum. However, the speed of light in an optical fiber is about one-third the speed of light in a vacuum. Therefore, the propagation delay for most fiber circuits works out to be approximately 6.3 microseconds per km or 8.2 microseconds per mile.

Another point to keep in mind when calculating propagation delay is that optical fibers are not always physically placed over the shortest path between two geographic points, especially over transoceanic links. Due to installation convenience, circuits may be hundreds or even thousands of miles longer than theoretically necessary. Also, repeaters and amplifiers in the path may introduce additional delay. Furthermore, link paths and details might not even be known to the end customer, and therefore it is important for a network administrator to understand service level agreements, specifically in terms of minimum, maximum, and average end-to-end delay.

Nonetheless, the G.114 real-time communications network latency budget of 150 ms allows for nearly 24,000 km or 15,000 miles worth of propagation delay (which is approximately 60 percent of the earth's circumference); the theoretical worst-case scenario (exactly half of the earth's circumference) would require only 126 ms of latency. Therefore, this latency target is usually achievable for nearly any two locations (via a terrestrial path), given relatively direct transmission paths. In some scenarios, however, meeting this latency target may simply not be possible because of the distances involved and the relative directness of their respective transmission paths. In such scenarios, if the G.114 150-ms one-way latency target cannot be met because of the distances involved, administrators should be aware that both the ITU and Cisco Technical Marketing have shown that real-time communication quality does not begin to significantly degrade until one-way latency exceeds 200 ms, as is illustrated in the ITU G.114 graph of real-time speech quality versus absolute delay, which is reproduced as Figure 27-2.

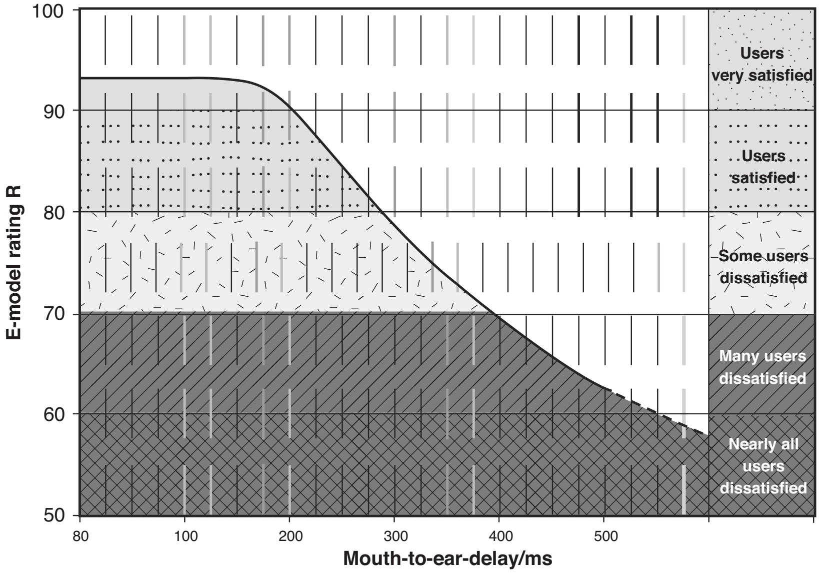


Figure 27-2 *ITU G.114 Graph of Real-Time Speech Quality Versus Latency*

Note The context so far has been for WAN circuits over terrestrial paths; for satellite circuits, the expected latency can be in the area of 250 to 900 ms. For example, signals being relayed via geostationary satellites will need to be sent to an altitude of 35,786 km (22,236 miles) above sea level (from the equator) out into space and then back to Earth again. There is nothing administrators can do to decrease latency in such scenarios because there is little they can do about increasing the speed of light or radio waves. All that can be done to address the effect of latency in these scenarios is to educate the user base so that realistic performance expectations are set.

The final network latency component to be considered is queuing delay, which is variable (variable delay is also known as jitter). Queuing delay is a function of whether a network node is congested and, if so, what scheduling policies have been applied to resolve congestion events. Real-time applications are often more sensitive to jitter than latency as a whole because packets need to be received in de-jitter buffers before being played out. If a packet is not received within the window allowed for by the de-jitter buffer, it's as good as lost and can affect the overall voice or video call quality.

Given that the majority of factors contributing to network latency are fixed, careful attention has to be given to queuing delay, because this is the only latency factor that is directly under network administrator control, via their queuing policies. Therefore, understanding the IOS queuing system, including the Tx-Ring and LLQ/CBWFQ operation, serves to assist network administrators optimize these critical policies.

Tx-Ring

Recommendation:

- Be aware of the Tx-Ring function and depth; tune only if necessary.

The Tx-Ring is the final IOS output buffer for a WAN interface (a relatively small first in, first out [FIFO] queue) that maximizes physical link bandwidth utilization by matching the outbound packet rate on the router with the physical interface rate. The operation of the Tx-Ring was previously illustrated in Figure 5-4 in Chapter 5, "Congestion Management and Avoidance Tools."

The Tx-Ring can be configured on certain platforms with the **tx-ring-limit** interface configuration command. The value of the **tx-ring-limit** number can be from 1 to 32,767 packets. The default value of the Tx-Ring varies according to platform and link type and speed. You can determine the size of the Tx-Ring with the **show controllers interface | include tx** verification command, as shown in Example 27-1.

Example 27-1 *Verifying the Tx-Ring Size: show controllers | include tx_limited*

```
Router# show controllers Serial 1/0 | include tx_limited
tx_underrun_err=0, tx_soft_underrun_err=0, tx_limited=1(64)
Router#
```

In Example 27-1, the default value of the Tx-Ring is reported to be 64 packets on this serial interface.

During Cisco Technical Marketing design validation, it was observed that the default Tx-Ring limit on some interfaces was shown to cause somewhat higher jitter values to some real-time application classes, particularly high-definition (HD) video-based real-time applications such as Cisco TelePresence traffic. The reason for this is the bursty nature of HD video traffic. For example, consider a fully congested T3 WAN link with active LLQ/CBWFQ policies. The default Tx-Ring depth in this case is 64 packets, as shown in Example 27-1. Even if TelePresence traffic is prioritized via a LLQ, if there are no TelePresence packets to send, the FIFO Tx-Ring is filled with other traffic to a default depth of 64 packets. When a new TelePresence packet arrives, even if it gets priority treatment from the Layer 3 LLQ/CBWFQ queuing system, the packets are dequeued into the FIFO Tx-Ring when space is available. However, with the default settings, there can be as many as 63 packets in the Tx-Ring in front of that TelePresence packet. In such a worst-case scenario, it could take as long as 17 ms to transmit these non-real-time packets out of this (45-Mbps) T3 interface. This 17 ms of instantaneous and variable delay (jitter) can affect the video quality for TelePresence to the point of being visually apparent to the end user.

On the one hand, lowering the value of the Tx-Ring on this link will force in the IOS Software engaging congestion management policies sooner and more often, resulting in lower overall jitter values for real-time applications, like TelePresence. On the other hand, setting the value of the Tx-Ring too low may result in significantly higher CPU utilization rates, because the processor is being continually interrupted to engage queuing policies, even when congestion rates are just momentary bursts and not sustained rates. Therefore, when tuning the Tx-Ring, a tradeoff setting is required such that jitter is minimized, but not at the expense of excessive CPU utilization rates.

CBWFQ

Recommendation:

- Understand class-based weighted fair queuing (CBWFQ) operation and design options.

CBWFQ is an IOS queuing algorithm that combines the ability to guarantee bandwidth with the ability to dynamically ensure fairness to other flows within a class of traffic. The operation of CBWFQ was previously illustrated in Figure 5-2 in Chapter 5.

An important point about bandwidth assigned to a given CBWFQ class is that the bandwidth allocated is not a static bandwidth reservation, but rather represents a minimum bandwidth guarantee to the class, provided there are packets offered to the class. If there are no packets offered to the class, the scheduler services the next queue and can dynamically redistribute unused bandwidth allocations to other queues, as necessary.

In addition, a fair-queuing presorter may be applied to specific CBWFQ queues with the **fair-queue policy-map class** configuration command. Note that this command enables a flow-based fair-queuing presorter, not a weighted fair-queuing presorter as the marketing name for this feature implies (and therefore the fair-queuing presorter does not take into account the IP Precedence values of any packets offered to a given class). For example, if a CBWFQ class were assigned 1 Mbps of bandwidth and there were four competing traffic flows contending for this class, a fair-queuing presorter would ensure that each flow receives (1 / (Total number of flows)) of bandwidth, or in this example (1/4 of 1 Mbps) 250 Kbps of bandwidth.

Note Before IOS Release 12.4(20)T, a fair-queue presorter could only be applied to class default. Since this software release, however, IOS includes the support of the Hierarchical Queuing Framework (HQF), which (among many other QoS feature enhancements) allows for a fair-queue presorter to be applied to any CBWFQ class.

The depth of a CBWFQ is defined by its queue limit, which varies according to link speeds and platforms. You can modify this queue limit with the **queue-limit policy-map class** configuration command. The queue limit is usually expressed as a number of packets, although on some platforms it may be expressed in bytes or in milliseconds. In some cases, you might have to increase the queue limit from the default value, such as when provisioning CBWFQs for bursty applications and noticing significant numbers of packet drops for the class.

LLQ

Recommendations:

- Understand LLQ operation and design options.
- Use a dual-LLQ design when deploying voice and real-time-video applications.
- Limit sum of all LLQs to 33 percent of bandwidth.
- Tune the burst parameter if required.

Low-latency queuing (LLQ) is essentially CBWFQ combined with a strict-priority queue. As discussed in Chapter 5, IOS LLQ includes an implicit policer that allows for the abstraction and presentation of "multiple-LLQs" for various traffic classes, when in reality, there is only a single LLQ. Therefore, the functionality offered by the implicit LLQ policer can be leveraged to prevent multiple types of real-time flows from interfering with each other (like, for instance, protecting voice from bursty real-time video).

Note Multi-LLQ configuration is presented in Example 5-3, and its operation is illustrated in Figure 5-3.

Traffic vying for strict-priority queuing is serviced on a first-come, first-serve basis, provided the packets are admitted by their respective policers. So, there is no operational advantage or disadvantage in the order of configuration of LLQ classes. In other words—referring back to Example 5-3—the VoIP class is not serviced ahead of Broadcast Video or Real-Time Interactive class, but rather each class is serviced on a first-come, first-serve basis until their policer limits are reached.

A final consideration with LLQ is the configuration of a burst parameter of the implicit LLQ policer, which is configured as an optional parameter of the priority **policy-map class** configuration command. Because traffic rates are not constant, especially for applications like video, packets may be presented to the LLQ implicit policers in subsecond bursts. For example, if 5 Mbps of traffic is provisioned for real-time-interactive flows, then—assuming a constant rate traffic flow—the policer would expect 625 bytes per millisecond (50 Mbps / 1000 ms per second / 8 bits per byte). However, for a bursty real-time-interactive application—such as Cisco TelePresence—this constant subsecond traffic rate is not the norm, but rather subsecond bursting is more typical. Specifically, a single Cisco TelePresence camera can produce traffic that bursts as high as 64 KB within a 33-ms video frame window. Therefore, to accommodate such applications, the burst parameter of the implicit LLQ policer may need to be adjusted. By default, the LLQ burst parameter computed as 200 ms of traffic at the configured bandwidth rate and is used when the burst argument is not specified. The range of the burst is from 32 to 2000000 bytes.

WRED

Recommendations:

- Understand DSCP-based WRED operation and benefits.
- Optionally tune WRED thresholds as required.
- Optionally enable ECN.

While congestion management mechanisms such as LLQ/CBWFQ manage the front of the queue, congestion avoidance mechanisms such as weighted random early detect (WRED) manage the tail of the queue. Congestion avoidance mechanisms work best with TCP-based applications because selective dropping of packets causes the TCP windowing mechanisms to "throttle back" and adjust the rate of flows to manageable rates.

Note Figure 5-6 provides an illustration of WRED operation.

To better match the Assured Forwarding (AF) per-hop behavior (PHB) (as defined in RFC 2597), DSCP-based WRED thresholds may be explicitly tuned to be consistent and compatible in the following manner:

- Set the minimum WRED thresholds for AFx3 to 60 percent of the queue depth.
- Set the minimum WRED thresholds for AFx2 to 70 percent of the queue depth.
- Set the minimum WRED thresholds for AFx1 to 80 percent of the queue depth.
- Set all WRED maximum thresholds to 100 percent.

WRED can also be used to set the RFC 3168 IP Explicit Congestion Notification (ECN) bits to indicate that congestion was experienced in transit. ECN functionality is enabled with the **ecn** keyword in conjunction with the **random-detect** policy map class configuration command.

RSVP

Recommendations:

- Enable RSVP for dynamic network-aware admission control requirements.
- Use the IntServ/DiffServ RSVP model to increase efficiency and scalability.
- Use application-identification RSVP policies for greater policy granularity.

Resource Reservation Protocol (RSVP) functionality and operation was discussed in detail in Chapter 6, "Bandwidth Reservation Tools." Therein, it was highlighted that RSVP is the only network-aware admission control mechanism currently available. Furthermore, if such functionality is required, the Cisco IOS RSVP agent functionality must be enabled on the WAN and branch routers.

Furthermore, to increase efficiency and scalability, it is recommended to deploy RSVP via the integrated/differentiated services (IntServ/DiffServ) model, wherein RSVP is only used for admission control decisions, while DiffServ mechanisms are used for classification, marking, policing, and scheduling functions. The RSVP IntServ/DiffServ model is enabled by configuring the **ip rsvp data-packet classification none** and **ip rsvp resource-provider none** interface commands.

In addition, you can deploy RSVP admission control policies on a more granular basis by integrating application identification (app ID) within the reservation requests (as based on RFC 2872). With such functionality, you can configure separate local (per-interface) RSVP policies to handle bandwidth requests on a per-application basis, rather than at an aggregate level alone.

A final word on RSVP: The main functionality it brings to network administrators is dynamic and network-aware admission control, which becomes increasingly relevant on voice/video networks where a single admission decision (such as deciding to admit or not a Cisco TelePresence call) can instantly add 20 Mbps or more of (potentially) real-time traffic onto a link.

Medianet

Recommendation:

- Leverage Medianet metadata awareness, where applicable, to support rich-media applications over the WAN.

Medianet tools were overviewed in Chapter 8, “Medianet.” The Medianet technology umbrella includes the following:

- Auto-Configuration tools
- Media Monitoring Tools
- Media Awareness Tools
- Media Services Interface and Proxy

The Medianet application metadata classification engine can be effectively deployed on WAN and branch routers to identify and classify rich-media applications running over WANs.

Note Other Medianet tools can also apply over the WAN and branch, such as the Media Service Proxy and Media Monitoring Tools (including Mediatrace, Performance Monitor, and IP SLA Video Operations). Because the core focus of this book is QoS design, however, these management and monitoring tools are not discussed in detail here.

AVC

Recommendation:

- Utilize NBAR2 for Layer 7 classification, as required.

Chapter 9, "Application Visibility and Control (AVC)," covers the application visibility control (AVC) toolset. AVC includes the following technologies:

- NBAR2
- Flexible NetFlow (FNF)
- Management and reporting applications

Of these AVC tools, the most relevant in the context of WAN and branch QoS design is NBAR2. As previously mentioned, with so many applications running over HTTP, a simple Layer 4 access list matching on TCP port 80 is often insufficient to achieve adequate application-based classification. Therefore, deeper packet inspection may be required, and because NBAR2 is not supported on Cisco Catalyst switches (at the time of this writing), the nearest point to perform such deep inspection may be at the WAN aggregator or branch router (at the LAN edge on ingress).

AutoQoS

Cisco AutoQoS is an administrative option in the WAN and branch and is discussed in detail in both Chapter 2, "IOS-Based QoS Architectural Framework and Syntax Structure," and in Appendix A, "AutoQoS for Medianet."

Control Plane Policing

Cisco IOS supports control plane policing (CPP), and this feature is detailed in Appendix B, "Control Plane Policing."

Link Types and Speeds

Recommendation:

- Rich media WANs require adequate bandwidth to support the bandwidth requirements of video-based applications.

Table 27-2 summarizes the link types and speeds supported on the WAN and branch platforms discussed in this part of the book and in Part VII, "MPLS VPN QoS Design."

Table 27-2 *Supported Link Types and Speeds*

Media	Line Rates
Serial	T3 (45 Mbps)
ATM	DS3 (45 Mbps) to OC48/STM16 (2.5 Gbps)
POS	OC3/STM1 (155 Mbps) to OC192/STM64 (10 Gbps)
Ethernet*	10 Mbps to 10 Gbps+

Note This part of the book ("WAN and Branch QoS Design") lays a foundation for wide-area designs and therefore features private WAN media (Serial/ATM/POS). Part VII, "MPLS VPN QoS Design," features Ethernet as a WAN/MAN access media.

Note Although these platforms can support serial links as slow as DS0 (64 Kbps) line rates, such rates are much too slow to support rich-media applications. Therefore, a minimum rate of 10 Mbps will be used in the context of these WAN and branch designs.

Note QoS designs for slower-speed links are discussed in the context of broadband scenarios in Part VIII, "IPsec QoS Design."

Note These minimum recommended line rates render the use of (computationally intensive) link-efficiency mechanisms, like IP RTP header compression (cRTP) and link-fragmentation and interleaving (LFI), not only unnecessary, but wasteful. Therefore, these tools are not included in the presented designs.

WAN and Branch QoS Models

Recommendations:

- Enable ingress and egress QoS models.
- Optionally enable CPP.

Generally speaking, two main (and one optional) steps enable you to configure QoS in the WAN and branch:

1. Apply an ingress QoS model (if required).

2. Apply an egress QoS model.
3. (Optional). Enable CPP.

Each of these models is overviewed in turn.

Ingress QoS Models

Recommendations:

- DSCP is trusted by default in IOS.
- Enable ingress NBAR2 classification, as required, on LAN edges.
- Enable ingress/internal queuing, if required.

By default, IOS trusts packet markings, whether these are Layer 2 CoS, Layer 2.5 MPLS EXP, or Layer 3 DSCP. Therefore, no explicit QoS policies are needed at the WAN and branch ingress edges if all packets have been classified and marked previously.

In the case that deep packet inspection (NBAR2) is required to perform classification, however, these policies may be applied on the LAN edges of the WAN and branch routers in the ingress direction.

In a WAN and branch context, the router backplane is rarely the bottleneck causing congestion. In some ASR/ESP combination scenarios, though, this might indeed be the case. Therefore, ingress/internal QoS recommendations and designs are presented in Chapter 28, “WAN Aggregator (Cisco ASR 1000) QoS Design,” to address these specific scenarios.

Egress QoS Models

Recommendations:

- Deploy egress queuing policies on all WAN edge interfaces.
- Egress queuing policies may not be necessary on LAN edge interfaces.

As previously discussed (in Chapter 11, at a minimum the following standards-based queuing behaviors should be supported on all platforms when deploying QoS for rich-media applications:

- Real-time queue (to support a RFC 3246 EF PHB service)
- Guaranteed-bandwidth queue (to support RFC 2597 AF PHB service)
- Default queue (to support a RFC 2474 DF service)
- Bandwidth-constrained queue (to support a RFC 3662 Scavenger service)

Note Unlike Cisco Catalyst switches, IOS QoS is not limited by a fixed number of queues, and most platforms will support up to 256 software queues, which is likely many more than most administrators will require.

Cisco offers design recommendations for each of these queue types:

- For the real-time queue, as previously discussed, extensive testing and customer deployments have shown that a general best queuing practice is to limit the amount of strict-priority queuing to 33 percent of link bandwidth capacity. This strict priority queuing rule is a conservative and safe design ratio for merging real-time applications with data applications. Also, any traffic assigned to a strict-priority queue should be governed by an admission control mechanism.

Note It is important to understand that this strict priority queuing rule is simply a best practice design recommendation, not a mandate. There might be cases where specific business objectives cannot be met while holding to this recommendation. In such cases, enterprises must provision according to their detailed requirements and constraints. However, it is important to recognize the tradeoffs involved with overprovisioning strict-priority traffic and its negative performance impact both on other real-time flows and also on non-real-time-application response times:

- Any guaranteed-bandwidth queues should be provisioned with a separate CBWFQ. Fair-queuing presorters can be enabled on multimedia and data application classes to ensure fairness to discrete flows within the class. In addition, DSCP-based WRED can be enabled (and optionally tuned) to minimize TCP global-synchronization.

  On CBWFQs used for control traffic, however, it is not recommended to enable fair-queuing presorters nor DSCP-based WRED (because traffic in these classes should neither be reordered nor dropped).

- A minimum bandwidth allocation is recommended on any deferential queue (such as the scavenger queue). Furthermore, neither fair-queuing presorters nor WRED should be enabled on this queue (to save marginal CPU cycles for managing traffic on a queue that has no implied good-faith service guarantee to begin with).

- The default/Best Effort class is the default class for all traffic that has not been explicitly assigned to another application class queue. Only if an application has been selected for preferential/deferential treatment is it removed from the default class. Because most enterprises have several thousand applications running over their networks, adequate bandwidth must be provisioned for this class as a whole to handle the sheer number and volume of applications that default to it. Therefore, it is recommended to provision at least 25 percent of link bandwidth for the default/Best Effort class. A fair-queuing presorter is recommended to be enabled on the default/Best Effort class, as is WRED.

These queue-based recommendations can be summarized as follows:

- **LLQs**
    - Limit the sum of all LLQs to 33%.
    - Govern admission to the LLQs with an admission control mechanism.
    - Do not enable WRED on the (drop-sensitive) LLQs.
- **Multimedia/Data CBWFQs**
    - Provision guaranteed-bandwidth allocations according to application requirements.
    - Enable fair-queuing presorters.
    - Enable DSCP-based WRED (tune if needed).
- **Control CBWFQs**
    - Provision guaranteed bandwidth allocations according to control traffic requirements.
    - Do not enable presorters.
    - Do not enable WRED.
- **Scavenger CBWFQ**
    - Provision with a minimum bandwidth allocation (for example, 1 percent).
    - Do not enable presorters.
    - Do not enable WRED.
- **Default/Best Effort CBWFQ**
    - Allocate at least 25 percent for the default/Best Effort queue.
    - Enable fair-queuing pre-sorters on the default/Best Effort queue.
    - Enable WRED on the default/Best Effort queue (tune if needed)

As with previous design chapters, 4-class, 8-class, and 12-class queuing models are presented for the WAN and branch.

A final note: Queuing policies are principally required on the WAN edges. Sometimes, however, these may be required on the LAN edges also (such as when the combined WAN bandwidth being serviced by a WAN aggregation router exceeds the LAN link's capacity to the campus distribution network). For example, this would be true where a WAN aggregator is servicing 110 branches at 10 Mbps each (for a potential total throughput of 1.1 Gbps), yet is connecting to the campus network via a Gigabit Ethernet (GE) link. Although such cases are rare, should these occur, the egress queuing policies would also be required on the LAN edges in the egress direction to provide service level guarantees in the rare event that all branch links were transmitting at capacity toward the campus.

Control Plane Policing

Recommendation:

- Understand the purpose and use of the control plane policing (CPP) feature as an optional QoS-for-security tool to harden the network infrastructure.

Appendix B provides design recommendations and configurations for CPP.

WAN and Branch Interface QoS Roles

Recommendation:

- Define consistent ingress and egress QoS policies for all router interface roles.

The QoS policy elements discussed so far can be grouped into roles that various router interfaces serve within the WAN and branch architecture, as illustrated in Figure 27-3.

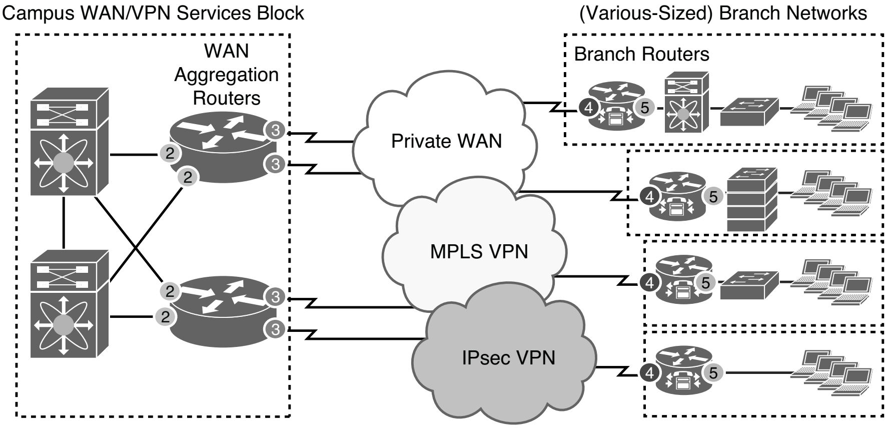


Figure 27-3 *WAN and Branch Interface QoS Roles*

The specific QoS policies for these roles are as follows:

1. WAN aggregator ingress/internal QoS (ASR-only): Internal QoS policies *may* be applied (if required).
2. WAN aggregator LAN edge:
   - Ingress DSCP-trust *should* be enabled (enabled by default).
   - Ingress NBAR2 classification and marking policies *may* be applied.

   - Ingress Medianet metadata classification and marking policies *may* be applied.
   - Egress LLQ/CBWFQ/WRED policies *may* be applied (if required).

3. WAN aggregator WAN edge:
   - Ingress DSCP-trust *should* be enabled (enabled by default).
   - Egress LLQ/CBWFQ/WRED policies *should* be applied.
   - RSVP policies *may* be applied.
   - Additional VPN-specific policies *may* be applied.
4. Branch WAN edge (may or may not be identical to WAN aggregator WAN edge):
   - Ingress DSCP-trust *should* be enabled (enabled by default).
   - Ingress LLQ/CBWFQ/WRED policies *should* be applied.
   - RSVP policies *may* be applied.
   - Additional VPN-specific policies *may* be applied.
5. Branch LAN edge (may or may not be identical to WAN aggregator LAN edge):
   - Ingress DSCP-trust *should* be enabled (enabled by default).
   - Ingress NBAR2 classification and marking policies *may* be applied.
   - Ingress Medianet Metadata classification and marking policies *may* be applied.
   - Egress LLQ/CBWFQ/WRED policies *may* be applied (if required).

Summary

This chapter discusses the QoS design considerations and recommendations for enterprise WAN and branch networks. The dual role of QoS in the WAN and branch is to manage packet loss and to manage packet jitter.

Strategic QoS design principles that apply in WAN and branch networks include the following:

- Always perform QoS in hardware rather than software when a choice exists.
- Classify and mark applications as close to their sources as technically and administratively feasible.
- Enable queuing policies at every node where the potential for congestion exists.
- (Optional) Protect the control plane and data plane by enabling control plane policing.

WAN and branch architectures were overviewed, highlighting the two main types of routers within this architecture: WAN aggregators and branch routers. The benefits and limitations of IOS Software QoS were then discussed. In addition, the performance capabilities of various WAN aggregation and branch routers were summarized.

Design considerations relating to latency and jitter were reviewed, along with various key components of the IOS QoS toolset relevant to WAN and branch design, including the following:

- Tx-Ring
- CBWFQ
- LLQ
- WRED
- RSVP
- AVC
- Medianet
- AutoQoS
- CPP

Following this, best-practice recommendations for ingress and egress WAN and branch policies were presented.

Finally, all these recommendations were summarized and grouped into various specific interface QoS roles, including the following:

- WAN aggregator LAN edge:
- WAN aggregator ingress/internal QoS (ASR-only)
- WAN aggregator WAN edge
- Branch WAN edge
- Branch LAN edge

Further Reading

Cisco Medianet WAN Aggregation QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html

Cisco Medianet WAN/VPN QoS Design At-a-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/qoswanvpnaag.html

Cisco ASR 1000 Series Embedded Services Processors Data Sheet: http://www.cisco.com/en/US/prod/collateral/routers/ps9343/data_sheet_c78-450070.html

Cisco 3900 Series Integrated Services Routers Data Sheet: http://www.cisco.com/en/US/prod/collateral/routers/ps10536/data_sheet_c78_553924.html

Cisco 2900 Series Integrated Services Routers Data Sheet: http://www.cisco.com/en/US/prod/collateral/routers/ps10537/data_sheet_c78_553896.html

Cisco 1941 Series Integrated Services Routers Data Sheet: http://www.cisco.com/en/US/prod/collateral/routers/ps10538/data_sheet_c78_556319.html

Cisco 1921 Integrated Services Router Data Sheet: http://www.cisco.com/en/US/prod/collateral/voicesw/ps6789/ps7290/ps10589/data_sheet_c78-598389.html

This page intentionally left blank

Chapter 28

WAN Aggregator (Cisco ASR 1000) QoS Design

The primary quality of service (QoS) role of the WAN aggregator router is to manage the congestion caused by the major reduction in speed from the LAN to the WAN, while minimizing packet loss and jitter. A secondary role of this router may be to perform deep packet inspection (DPI) for ingress application classification.

WAN aggregator routers connect not only to private WAN networks but may also connect to Multiprotocol Label Switching (MPLS) or IPsec-based virtual private networks (VPNs). The interface-specific QoS roles of WAN aggregator routers are illustrated in Figure 28-1.

Note This chapter focuses on private WAN connections and will serve as a base for the VPN-specific design chapters to build on. Therefore, VPN-specific design considerations and policies are deferred to later chapters, including sub-line-rate circuit designs (such as Ethernet-based WAN scenarios).

The Cisco ASR 1000 series router is well suited to the role of a WAN aggregator router. Therefore, to begin, we will review this platform's architecture, particularly as it relates to QoS.

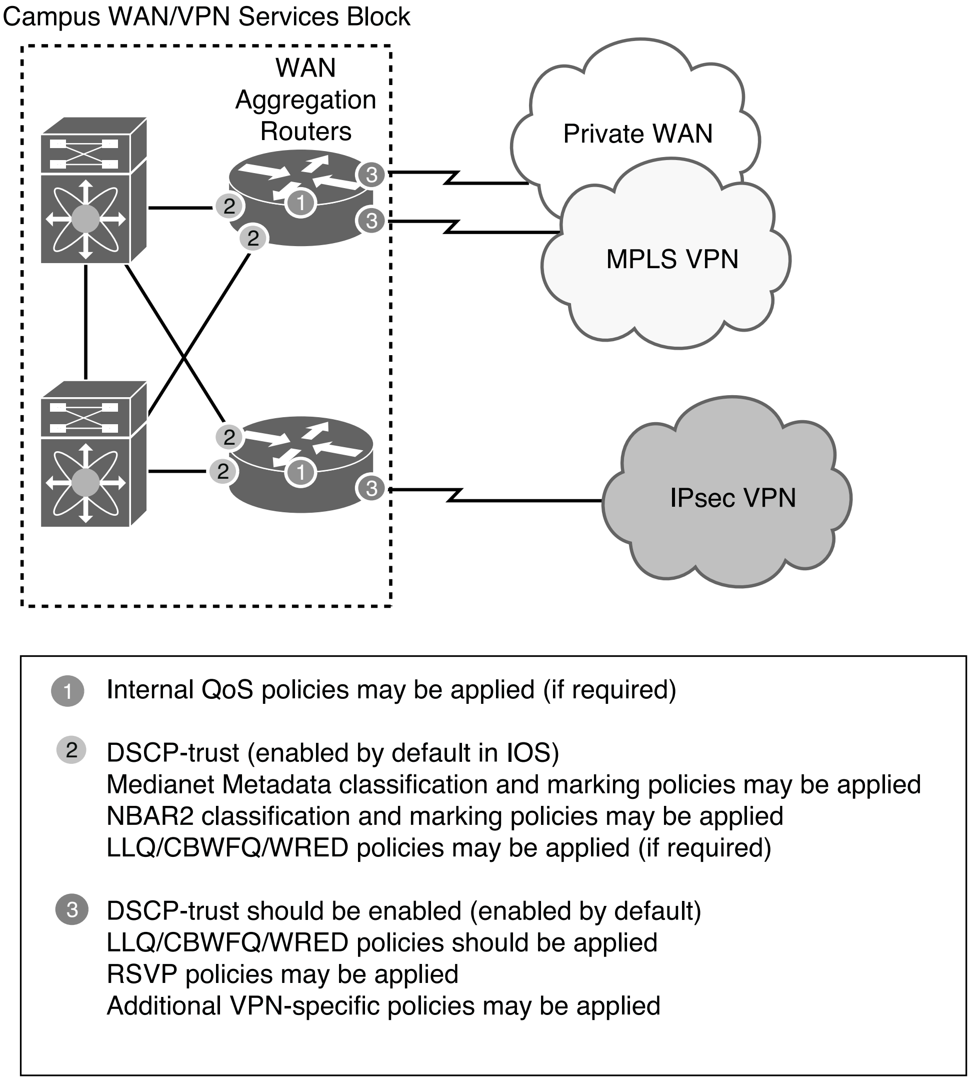

Campus WAN/VPN Services Block
WAN Aggregation Routers
1
2
3
Private WAN
MPLS VPN
IPsec VPN
1 Internal QoS policies may be applied (if required)
2 DSCP-trust (enabled by default in IOS)
Medianet Metadata classification and marking policies may be applied
NBAR2 classification and marking policies may be applied
LLQ/CBWFQ/WRED policies may be applied (if required)
3 DSCP-trust should be enabled (enabled by default)
LLQ/CBWFQ/WRED policies should be applied
RSVP policies may be applied
Additional VPN-specific policies may be applied

Figure 28-1 *WAN Aggregator Interface QoS Roles*

Cisco ASR 1000 QoS Architecture

At a high level, the architecture of the Cisco ASR 1000 series routers can be partitioned into three main elements:

- **Network control plane:** Consisting of active and standby route processors (RPs)
- **Data-plane forwarding:** Consisting of active and standby embedded services processors (ESPs)

- **Network input/output:** Consisting of shared port adapters (SPAs) controlled by one or more SPA interface processor (SIPs)

Figure 28-2 graphically depicts a redundant Cisco ASR 1000 router with two route processors, two ESPs, and three SIPs.

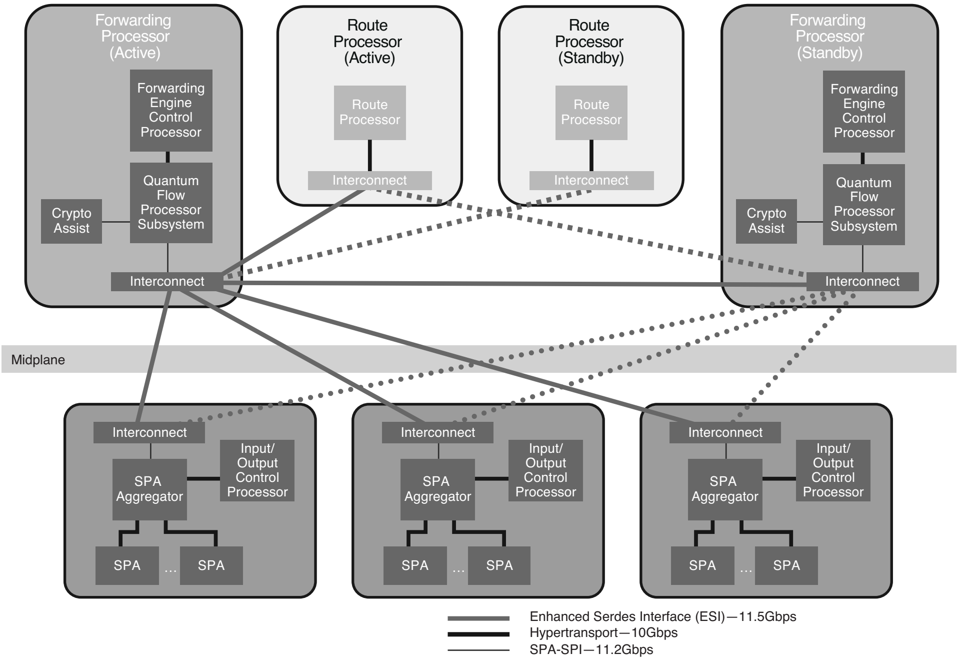
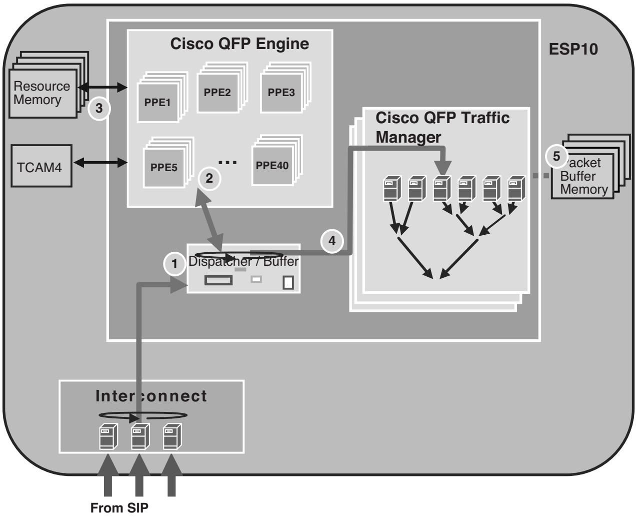

Forwarding Processor (Active)
Forwarding Engine Control Processor
Quantum Flow Processor Subsystem
Crypto Assist
Interconnect
Route Processor (Active)
Route Processor
Interconnect
Route Processor (Standby)
Route Processor
Interconnect
Forwarding Processor (Standby)
Forwarding Engine Control Processor
Quantum Flow Processor Subsystem
Crypto Assist
Interconnect
Midplane
Interconnect
SPA Aggregator
Input/ Output Control Processor
SPA ... SPA
Interconnect
SPA Aggregator
Input/ Output Control Processor
SPA ... SPA
Interconnect
SPA Aggregator
Input/ Output Control Processor
SPA ... SPA
Enhanced Serdes Interface (ESI)—11.5Gbps
Hypertransport—10Gbps
SPA-SPI—11.2Gbps

Figure 28-2 *Cisco ASR 1000 Hardware Architectural*

QoS is performed within the ESPs. The main QoS component of ESPs is the Cisco QuantumFlow Processor (QFP) and its two chips:

- **Cisco QFP Engine:** In charge of Modular QoS command-line interface (MQC) classification, marking, policing, and weighted random early detection (WRED)
- **Cisco QFP Traffic Manager:** Handles shaping, low-latency queuing (LLQ), and class-based weighted fair queuing (CBWFQ) scheduling.

Figure 28-3 illustrates these chips along with the packet flow for QoS operations within an ESP.



1. Ingress packets are temporarily stored in small internal packet buffer until processed
2. Free QFP Engine is allocated for this packet and SW begins processing packet (MAC classification, QoS classification, ACLs, forwarding lookup, police, WRED, etc.) including modifying packet contents
3. System accesses tables in resource DRAM and TCAM to perform lookups for features enabled for this packet, update statistics, update state for stateful features, etc.
4. Once packet processing is complete and packet has been modified, system issues request to enqueue packet to an output queue
5. The packet contents is copied from the internal pkt buffer to the deep output packet buffer where it is stored until scheduled for output

Figure 28-3 *Cisco ASR 1000 ESP Architecture and QoS Flow*

A significant advantage of the ASR 1000 architecture is the fact that Cisco QFP Traffic Manager is not affected by any of the features that Cisco QFP Engine needs to perform. So, QoS shaping and queuing configurations have minimum impact performance in the Cisco ASR 1000. Therefore, even though QoS performance on the ASR depends on the QoS configuration and the combination of other enabled features, it is very scalable.

QoS Design Steps

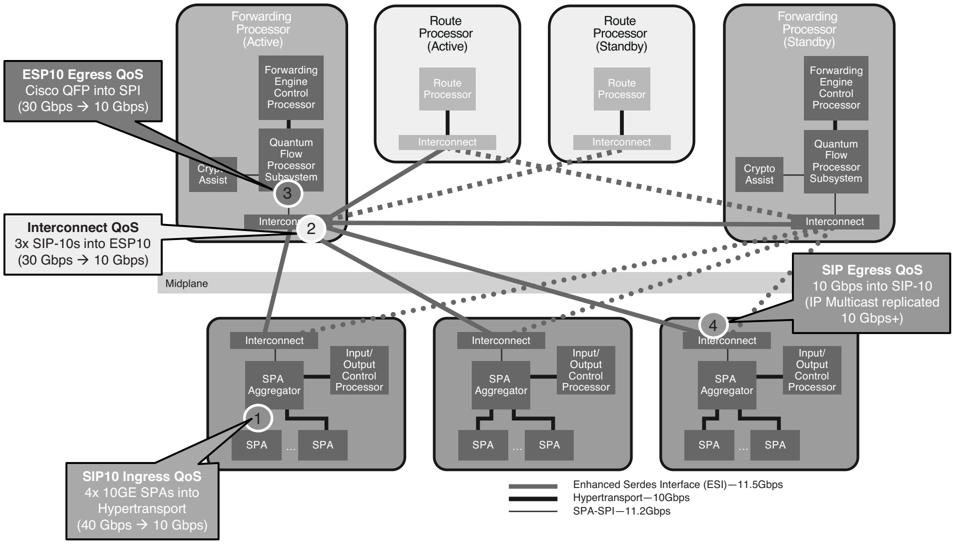
Configuring QoS on a Cisco ASR 1000 router performing the role of a WAN aggregator requires three main steps:

1. Enable internal QoS, as required.
2. Configure ingress QoS models, including the following:
   - Trust DSCP (enabled by default in IOS)
   - Medianet metadata classification and marking
   - NBAR2 classification and marking
3. Configure egress QoS policies.

Each of these design steps is covered in turn.

ASR 1000 Internal QoS

Given the Cisco ASR 1000's internal hardware architecture, there may be various combinations of ESPs and SIPs that can create the potential for internal oversubscription, as illustrated in Figure 28-4, which features a Cisco ASR 1000 with two ESP-10s and three SIP-10s with four 10-GE SPAs each.


Forwarding Processor (Active)
Forwarding Engine Control Processor
Quantum Flow Processor Subsystem
Crypto Assist
Interconnect
Route Processor (Active)
Route Processor
Interconnect
Route Processor (Standby)
Route Processor
Interconnect
Forwarding Processor (Standby)
Forwarding Engine Control Processor
Crypto Assist
Quantum Flow Processor Subsystem
Interconnect
Midplane
Interconnect
SPA Aggregator
Input/ Output Control Processor
SPA
...
SPA
Interconnect
SPA Aggregator
Input/ Output Control Processor
SPA
...
SPA
Interconnect
SPA Aggregator
Input/ Output Control Processor
SPA
...
SPA
1
2
3
4
Enhanced Serdes Interface (ESI)—11.5Gbps
Hypertransport—10Gbps
SPA-SPI—11.2Gbps

ESP10 Egress QoS
Cisco QFP into SPI
(30 Gbps → 10 Gbps)

Interconnect QoS
3x SIP-10s into ESP10
(30 Gbps → 10 Gbps)

SIP Egress QoS
10 Gbps into SIP-10
(IP Multicast replicated
10 Gbps+)

SIP10 Ingress QoS
4x 10GE SPAs into
Hypertransport
(40 Gbps → 10 Gbps)

Figure 28-4 *Cisco ASR 1000 Potential Internal Congestion Points*

Each of the callout boxes in Figure 28-4 represents a possible congestion point internal to the ASR 1000. Specifically, these potential congestion points are as follows:

1. **SIP-10 ingress classification and scheduling:** This point represents a possible 4:1 oversubscription point (up to 40 Gbps of SPA interfaces contending for a 10-Gbps interconnect).

2. **ESP-10 interconnect scheduling:** This point represents a possible 3:1 oversubscription point (three SIP-10s contending for a 10-Gbps interconnect).

3. **ESP-10 egress QoS:** This point represents a possible 3:1 oversubscription point (30 Gbps from the QFP contending for a 10-Gbps interconnect).

4. **SIP-10 buffering and egress QoS:** This point represents a possible 1:1+ oversubscription point (10 Gbps from the interconnect contending for 10 Gbps of SPA interfaces, and 1.2 Gbps to address the possible requirement of replicating IP multicast packets, for a maximum throughput of 11.2 Gbps).

To deal with these potential oversubscription scenarios, the Cisco ASR 1000 uses a two-queue internal bandwidth scheduler to prioritize packets: one queue operates as a strict-priority queue, and the other operates as a low-priority queue.

Figure 28-5 shows a basic overview of how this internal queuing system operates on ingress and highlights how the system is constantly aware of the queue status. Because of this internal queue status communication, high-priority packets can always be processed, even when the interconnect is overcommitted, because the interconnect uses backpressure from the Cisco QFP to delay low-priority packets so that high-priority packets continue to flow.

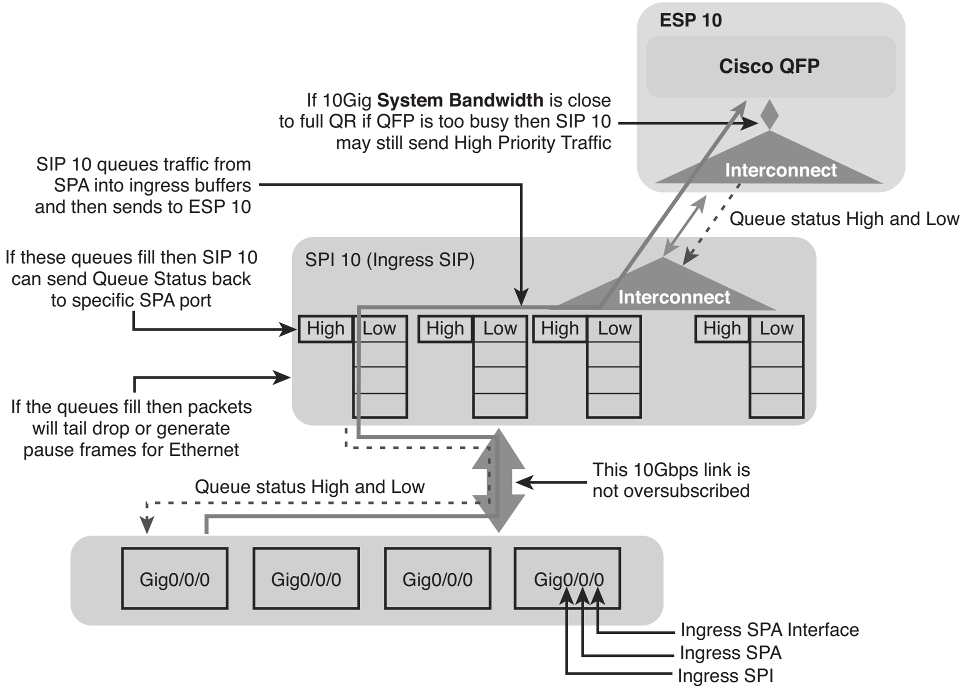

ESP 10
Cisco QFP
If 10Gig **System Bandwidth** is close to full QR if QFP is too busy then SIP 10 may still send High Priority Traffic
SIP 10 queues traffic from SPA into ingress buffers and then sends to ESP 10
Interconnect
Queue status High and Low
If these queues fill then SIP 10 can send Queue Status back to specific SPA port
SPI 10 (Ingress SIP)
Interconnect
High
Low
High
Low
High
Low
High
Low
If the queues fill then packets will tail drop or generate pause frames for Ethernet
This 10Gbps link is not oversubscribed
Queue status High and Low
Gig0/0/0
Gig0/0/0
Gig0/0/0
Gig0/0/0
Ingress SPA Interface
Ingress SPA
Ingress SPI

Figure 28-5 *Cisco ASR 1000 Internal Ingress Scheduling Operation*

In the opposite direction, Figure 28-6 illustrates how scheduling takes place on egress. Within the Cisco QFP, the QFP Traffic Manager Scheduler performs packet-scheduling decisions based on MQC policies. Then, the internal bandwidth scheduler allocates the egress packets (from the QFP) among the SIPs, selecting first the internally designated high-priority packets. Excess bandwidth—bandwidth remaining after the internally designated high-priority packets have been serviced—is evenly shared among the SIPs (that is, unless the administrator has tuned internal scheduling weights or set internal scheduling minimum bandwidth guarantees per SIP). Finally, shallow buffers on the SIP and SPAs are used to allow simultaneous packet transfer out of multiple ports.

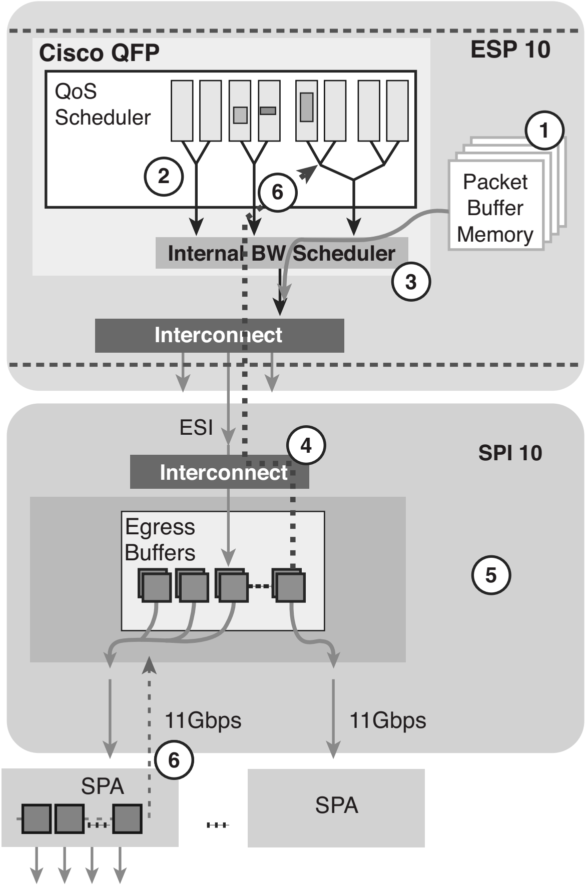


Figure 28-6 *Cisco ASR 1000 Internal Egress Scheduling Operation*

Note that not every combination of SIPs and ESP poses a potential oversubscription scenario. For example, Table 28-1 details some oversubscription scenarios for SIP-10s.

Even though oversubscription scenarios are possible on the ASR 1000 platforms, internal schedulers can be configured to prioritize application classes.

Although the ASR 1000 internal QoS architecture may seem complex, the tasks required to configure internal hardware queuing is relatively simple: The main requirement is to identify (at either the SPA or the SIP level) the packets that are to be assigned to the strict-priority internal queue.

Table 28-1 *Cisco ASR 1000 SIP-10 Oversubscription Scenarios*

ASR 1000 Chassis Type	ESP Type	Oversubscription State
■ ASR1000-SIP10G ■ Incoming rate from SPAs. Maximum 4 multiplied by 11.2 Gbps ■ Outgoing rate toward ESP 11.2 GbpsConclusion: SIP10G is always oversubscribed.	ESP-2.5 G and ESP-5 G (Supported only on ASR-1002)	**No ESP oversubscription.** ESPs interconnect device handles the entire 10G incoming traffic.
	ESP-10G	**ESP is oversubscribed.** Example: Input rate from 3 carrier cards is (3 × 11.2 Gbps) 33.6 Gbps. Output rate toward QFP is 12.8 Gbps. Therefore the QFP is oversubscribed (by 20.8 Gbps) in this example.
	ESP-20G	**ESP is oversubscribed.** Example: Input rate from 3 carrier cards is (3 × 11.2 Gbps) 33.6 Gbps. Output rate toward QFP is 25.6 Gbps. Therefore the QFP is oversubscribed (by 8 Gbps) in this example.
	ESP-40G	**No ESP oversubscription.** Example: Input rate from 3 carrier cards is (3 × 11.2 Gbps) 33.6 Gbps. Output rate toward QFP is (2 × 25.6 Gbps) 51.2 Gbps. Therefore the QFP is not oversubscribed.
	ESP-100G	**No ESP oversubscription.** Example: Input rate from 3 carrier cards is (3 × 11.2Gbps) 33.6 Gbps. Output rate toward QFP is (4 × 25.6 Gbps) 102.4 Gbps. Therefore the QFP is not oversubscribed.

Note Optionally, the scheduling of low-priority packets may also be weighted/guaranteed on a per SIP or SPA basis. However, the design principles in this chapter are based on application priority (as indicated by differentiated services code point [DSCP] values) rather than by interface priority. Therefore, this type of low-priority scheduling tuning is not covered in this design chapter, but can be referenced in the documentation at http://www.cisco.com/en/US/partner/docs/interfaces_modules/shared_port_adapters/configuration/ASR1000/ASRimpqos.html#wp1292478.

Ethernet SPAs and ATM SPAs require internal priority classification at the SPA level. However, POS SPAs, channelized SPAs, and clear-channel SPAs require internal priority classification at the SIP level. Table 28-3 provides a matrix of SPAs by internal classification schemes.

Table 28-3 *Cisco ASR SPA-Based Matrix of Ingress Classification by SIP or SPA Level*

Classification at the SPA Level	Classification at the SIP Level
Ethernet SPAs	Serial and channelized SPA
■ SPA-4X1FE-TX-V2	■ SPA-2XCT3/DS0
■ SPA-8X1FE-TX-V2	■ SPA-4XCT3/DS0
■ SPA-2X1GE-V2	■ SPA-8XCHT1/E1
■ SPA-5X1GE-V2	■ SPA-1XCHSTM1/OC3
■ SPA-8X1GE-V2	■ SPA-1XCHOC12/DS0
■ SPA-10X1GE-V2	■ SPA-2xT3/E3
■ SPA-1X10GE-L-V2	■ SPA-4xT3/E3
	■ SPA-4xT-SERIAL
ATM SPAs	■ POS SPAs
■ SPA-1XOC3-ATM-V2	■ SPA-2XOC3-POS
■ SPA-3XOC3-ATM-V2	■ SPA-4XOC3-POS
■ SPA-1XOC12-ATM-V2	■ SPA-1XOC12-POS
	■ SPA-2XOC12-POS
	■ SPA-4XOC12-POS
	■ SPA-8XOC12-POS
	■ SPA-8XOC3-POS
	■ SPA-1XOC48POS/RPR
	■ SPA-2XOC48POS/RPR
	■ SPA-4XOC48POS/RPR
	■ SPA-OC192POS-XFP

Both SPA-level and SIP-level internal scheduling classification takes place at the Physical Layer Interface Module (PLIM). These are discussed in turn.

SPA-Based PLIM

Ethernet and ATM SPAs perform classification at the SPA level. In the SPA-based classification model, the SPA performs both Layer 2 and Layer 3 classification and decides on the internal priority of the packet. After classifying the packets into high priority and low priority, the SPA has unique channels per priority: All high-priority packets are sent on separate channels than the low-priority packets. In such a scenario, the SPA queues the packets on high channels to high-priority buffers and low-channels to low-priority buffers. After the packets are classified into high priority and low priority, the packets are then sent to the ESP for further processing.

Internal classification can be based on DSCP, IPv6 traffic class, MPLS EXP or 802.1Q/p class of service (CoS) values. DSCP-based classification is used in the following examples (for policy consistency).

The ASR-specific CLI for enabling SPA-based internal scheduling classification is the **plim qos input map** interface configuration command, as shown in Example 28-1. By default, Expedited Forwarding (EF) is mapped to the strict-priority internal queue, and all other DSCP values are mapped to the low-priority internal queue. Therefore, the only amendment needed from these default settings is to map CS4 (real-time interactive) and CS5 (broadcast video) to the strict-priority internal queue.

Example 28-1 *Cisco ASR 1000 SPA-Based Internal Scheduling Classification Example*

```
ASR(config)# interface GigabitEthernet0/0/0
ASR(config-if)# plim qos input map ip dscp-based
 ! Designates that internal scheduling is to be DSCP-based
ASR(config-if)# plim qos input map ip dscp  cs4 cs5 ef queue strict-priority
 ! Maps CS4 (real-time interactive) and
 ! CS5 (broadcast video) to the internal PQ
 ! (EF is already mapped to the internal PQ by default)
```

Note CS4 and CS5 are converted to DSCP 32 and 40 (respectively) in the ASR configuration. In addition, neither EF nor 46 will appear in the configuration because DSCP EF/46 is mapped to the internal priority queue by default. Including it in Example 28-1 has been done simply for the sake of consistency with previous design examples that map all three values to the strict-priority queue.

You can verify the configuration in Example 28-1 with the **show platform hardware interface** *type sip/spa/interface* **plim qos input map** command, as demonstrated in Example 28-2.

Example 28-2 *Verifying ASR SPA-Based Ingress Scheduling Classification:* ***show platform hardware interface plim qos input map***

```
ASR# show platform hardware interface Gig 0/0/0 plim qos input map
Interface GigabitEthernet0/0/0
    Low Latency Queue(High Priority):
      IP DSCP, 32, 40, 46
      IPv6 TC, 46
      MPLS EXP, 6, 7
```

As shown in Example 28-2, DSCP values 32 (CS4), 40 (CS5), and EF (46) are mapped to the internal priority queue.

SIP-Based PLIM

Packet over SONET (POS), channelized, and clear-channel SPAs support packet classification at SIP level. In SIP-based classification, the SIP does the classification for SPAs and classifies the packet as either high-priority or low-priority.

High-priority packets are classified via a template defined with the **ingress-class map** index command. The classification details are defined inside the template, and then the template is attached to an interface using the **plim qos class-map** index interface configuration command.

Similar to the SPA-based internal scheduling classification example, DSCP-based classification is used in this SIP-based example. By default, EF is mapped to the strict-priority internal queue, and all other DSCP values are mapped to the low-priority internal queue. Therefore, the only amendment needed for these default settings is to map CS4 (real-time interactive) and CS5 (broadcast video) to the strict-priority internal queue via the ingress-class map index, as shown in Example 28-3.

Example 28-3 *Cisco ASR 1000 SIP-Based Internal Scheduling Classification Example*

```
 ! This section defines the ingress scheduling classification template
ASR(config)# ingress-class-map 1
ASR(config-ing-class-map)# map ip dscp-based
 ! Designates that internal scheduling is to be DSCP-based
ASR(config-ing-class-map)# map ip dscp 32 40 queue strict-priority
 ! Maps DSCP 32 (CS4 / real-time interactive) and
 ! DSCP 40 (CS5 / broadcast video) to the internal PQ
 ! (EF is already mapped to the internal PQ by default)
! This section attaches the ingress scheduling classification template ! to the
interface(s)
ASR(config)# interface POS0/1/0
ASR(config-if)# plim qos input class-map 1
 ! Attaches the ingress scheduling classification template
 ! to the interface
```

You can verify the configuration in Example 28-3 with the **show platform hardware interface** *type sip/spa/interface* **plim qos input map** command, as shown in Example 28-4.

Example 28-4 *Verifying ASR SIP-Based Ingress Scheduling Classification:* ***show platform hardware interface plim qos input map***

```
ASR# show platform hardware interface pos 0/1/0 plim qos input map
Interface POS0/1/0 uses
Ingress-class-map 1
High-priority queue settings:
  IPv4 DSCP: 32 40 46
  IPv6 TC: 46
  MPLS EXP: 6 7
```

As shown in Example 28-4, interface POS 0/1/0 references the ingress class map 1 index (associated with its SIP), which maps DSCP values 32 (CS4), 40 (CS5), and EF (46) to the internal priority queue.

Ingress QoS Models

As previously mentioned, classification and marking should be performed as close to the edge as possible. However, Catalyst switches have limited capability (if any) to perform deep packet inspection (DPI), such as or Medianet metadata or Network Based Application Recognition 2 (NBAR2) flow identification. Therefore, there may be a need to configure ingress DPI classification policies on the LAN edge of the WAN aggregation router.

However, such ingress DPI policies would be identical to—and more commonly deployed—on branch router LAN edges. Therefore, to minimize redundancy, ingress DPI QoS policies (although potentially applicable to WAN aggregators) are detailed in Chapter 29, “Branch Router (Cisco ISR G2) QoS Design.”

Egress QoS Models

Although the Cisco ASR 1000 performs queuing in hardware, it is not bound by a four or eight physical queue structure, like Cisco Catalyst switches. Rather, the Cisco ASR 1000 supports up to 16,000 queues per interface with three levels of hierarchy. Therefore, it can handily support the 4-class, 8-class, and even the 12-class strategic QoS models with dedicated hardware queues per class, as discussed next.

Four-Class Model

In the four-class model (illustrated in Figures 11-3 and 11-4), the application class to queue mappings are as follows:

- **Real-time** (marked EF): Assigned to a LLQ with 33 percent of strict-priority bandwidth
- **Control** (marked CS3): Assigned to a CBWFQ with 7 percent guaranteed-bandwidth allocation
- **Transactional data** (marked AF2): Assigned to a CBWFQ with 35 percent guaranteed-bandwidth allocation, with a fair-queuing presorter enabled and with DSCP-based WRED enabled
- **Best effort** (marked DF): Assigned to the default CBWFQ with 25 percent guaranteed-bandwidth allocation, with a fair-queuing presorter enabled and with WRED enabled

Example 28-5 shows the configuration of a four-class egress queuing policy applied to a T3 serial interface.

Example 28-5 *Cisco ASR 1000 Four-Class Egress Queuing Model Example*

```
! This section configures the class maps for the
! Four-class egress queuing policy
ASR(config-cmap)# class-map match-all REALTIME
ASR(config-cmap)#  match dscp ef
 ! Matches Realtime traffic on DSCP EF (IPv4/IPv6)
ASR(config-cmap)# class-map match-all CONTROL
ASR(config-cmap)#  match dscp cs3
 ! Matches Control traffic on DSCP CS3 (IPv4/IPv6)
ASR(config-cmap)# class-map match-any TRANSACTIONAL-DATA
ASR(config-cmap)#  match dscp af21
ASR(config-cmap)#  match dscp af22
ASR(config-cmap)#  match dscp af23
 ! Matches Transactional Data traffic on AF2 (IPv4/IPv6)
 ! This section defines the four-class egress queuing policy
ASR(config)# policy-map FOUR-CLASS-WAN-EDGE
ASR(config-pmap)# class REALTIME
ASR(config-pmap-c)# priority percent 33
 ! Provisions the REALTIME class with 33% LLQ
ASR(config-pmap)# class CONTROL
ASR(config-pmap-c)# bandwidth percent 7
 ! Provisions the CONTROL class with 7% CBWFQ
ASR(config-pmap)# class TRANSACTIONAL-DATA
ASR(config-pmap-c)# bandwidth percent 35
 ! Provisions the TRANSACTIONAL-DATA class with 35% CBWFQ
ASR(config-pmap-c)# fair-queue
 ! Enables a fair-queuing presorter
ASR(config-pmap-c)# random-detect dscp-based
 ! Enables DSCP-based WRED
ASR(config-pmap)# class class-default
ASR(config-pmap-c)# bandwidth percent 25
 ! Provisions the default/Best Effort class with 25% CBWFQ
ASR(config-pmap-c)# fair-queue
 ! Enables a fair-queuing presorter
ASR(config-pmap-c)# random-detect
 ! Enables WRED


 ! This section applies the queuing policy to a serial (T3) interface
 ! And tunes the Tx-Ring
ASR(config)# interface Serial2/1/0
ASR(config-if)# tx-ring-limit 10
  ! Optimizes the Tx-Ring for the serial (T3) interface
ASR(config-if)# service-policy output FOUR-CLASS-WAN-EDGE
  ! Attaches egress queuing service policy to serial T3 interface
```

Note The **match dscp** class map command is used in all class map examples in this design chapter, rather than **match ip dscp**. This is because **match ip dscp** is an older syntax that matches the DSCP values of only IPv4 packets, whereas **match dscp** will match the DSCP values of IPv4/IPv6 packets.

Note Multiple DSCP values can be matched on a single line (for example, **match af21 af22 af23**) or can be matched on three separate lines (as shown in Example 28-5). The approach used is largely a matter of administrative preference. The only advantage of one approach over the other is that the latter will provide more visibility via the **show policy-map interface** verification command and the corresponding class-based QoS MIB statistics.

Note Class maps that match on a single code point may use either the (default) **match-all** logical operator or the **match-any** logical operator. However, class maps that match on DSCP values using the multiple line approach *must* use the **match-any** logical operator; otherwise, no matches will ever register (as no packet can have a DSCP value of *x* and a DSCP value of *y*).

Note WRED is enabled *only* on the transactional data queue and the default queue (as real-time traffic and control traffic should never be early dropped). Furthermore, only WRED is needed (as opposed to DSCP-based WRED) on the default class, because all traffic in this class should be marked Default Forwarding (DF).

Note One consideration specific to T3 serial interfaces is that the Tx-Ring may need to be tuned. Cisco Technical Marketing has found that tuning the Tx-Ring to a value of 10 on T3 (and similarly on E3) serial interfaces is optimal for supporting HD-based rich-media applications, such as Cisco TelePresence.

You can verify this configuration with the following commands:

- **show class-map** (as shown in Example 28-6)
- **show policy-map** (as shown in Example 28-7)
- **show policy-map interface**
- **show controllers interface** *type sip/spa/interface* | **include tx_limited** (as shown in Example 27-1 in Chapter 27, “WAN and Branch QoS Design Considerations and Recommendations”)

Example 28-6 *Verifying Class Maps:* ***show class-map***

```
ASR# show class-map
 Class Map match-all REALTIME (id 17)


   Match   dscp ef (46)


 Class Map match-all CONTROL (id 18)


   Match   dscp cs3 (24)


 Class Map match-any TRANSACTIONAL-DATA (id 19)
```

```
   Match   dscp af21 (18)
   Match   dscp af22 (20)
   Match   dscp af23 (22)

ASR#
```

Example 28-7 *Verifying Policy Maps:* ***show policy-map***

```
ASR# show policy-map FOUR-CLASS-WAN-EDGE
  Policy Map FOUR-CLASS-WAN-EDGE
    Class REALTIME
      priority 33 (%)
    Class CONTROL
      bandwidth 7 (%)
    Class TRANSACTIONAL-DATA
      bandwidth 35 (%)
      fair-queue
       wred, exponential weight 4

      dscp    min-threshold    max-threshold    mark-probability
      ----------------------------------------------------------
      default (0)   -                -                1/10
    Class class-default
      bandwidth 25 (%)
      fair-queue
       wred, exponential weight 4

      class    min-threshold    max-threshold    mark-probability
      ----------------------------------------------------------
      0       -                -                1/10
      1       -                -                1/10
      2       -                -                1/10
      3       -                -                1/10
      4       -                -                1/10
      5       -                -                1/10
      6       -                -                1/10
      7       -                -                1/10
ASR#
```

Eight-Class Model

In the eight-class model (illustrated in Figures 11-5 and 11-6) the application class to queue mappings are as follows:

- **Voice (marked EF):** Assigned to a LLQ with 10 percent of strict-priority bandwidth
- **Multimedia conferencing (marked AF4):** Assigned to a second LLQ with 23 percent of strict-priority bandwidth
- **Network control (marked CS6):** Assigned to a CBWFQ with 5 percent guaranteed-bandwidth allocation
- **Signaling (marked CS3):** Assigned to a CBWFQ with 2 percent guaranteed-bandwidth allocation
- **Multimedia streaming (marked AF3):** Assigned to a CBWFQ with 10 percent guaranteed-bandwidth allocation, with a fair-queuing presorter enabled and with DSCP-based WRED enabled
- **Transactional data (marked AF2):** Assigned to a CBWFQ with 24 percent guaranteed-bandwidth allocation, with a fair-queuing presorter enabled and with DSCP-based WRED enabled
- **Scavenger (marked CS1):** Assigned to a bandwidth-constrained CBWFQ provisioned at 1 percent
- **Best effort (marked DF):** Assigned to the default CBWFQ with 25 percent guaranteed-bandwidth allocation, with a fair-queuing presorter enabled and with WRED enabled

Example 28-8 shows the configuration of an eight-class egress queuing policy applied to an OC-3 ATM PVC.

Example 28-8 *Cisco ASR 1000 Eight-Class Egress Queuing Model Example*

```
! This section configures the class maps for the
! eight-class egress queuing policy
ASR(config)# class-map match-all VOICE
ASR(config-cmap)# match dscp ef
 ! Matches Voice traffic on EF (IPv4/IPv6)
ASR(config-cmap)# class-map match-any MULTIMEDIA-CONFERENCING
ASR(config-cmap)#  match dscp af41
ASR(config-cmap)#  match dscp af42
ASR(config-cmap)#  match dscp af43
 ! Matches MM-Conferencing traffic on AF4 (IPv4/IPv6)
ASR(config-cmap)# class-map match-all NETWORK-CONTROL
ASR(config-cmap)#  match dscp cs6
 ! Matches Network Control traffic on CS6 (IPv4/IPv6)
ASR(config-cmap)# class-map match-all SIGNALING
ASR(config-cmap)#  match dscp cs3
 ! Matches Signaling traffic on CS3 (IPv4/IPv6)
ASR(config-cmap)# class-map match-any MULTIMEDIA-STREAMING
ASR(config-cmap)#  match dscp af31
ASR(config-cmap)#  match dscp af32
ASR(config-cmap)#  match dscp af33
 ! Matches MM-streaming traffic on AF3 (IPv4/IPv6)
ASR(config-cmap)# class-map match-any TRANSACTIONAL-DATA
ASR(config-cmap)#  match dscp af21
ASR(config-cmap)#  match dscp af22
ASR(config-cmap)#  match dscp af23
 ! Matches transactional data on AF2 (IPv4/IPv6)
ASR(config-cmap)# class-map match-all SCAVENGER
ASR(config-cmap)#  match dscp cs1
 ! Matches scavenger traffic on CS1 (IPv4/IPv6)


 ! This section defines the eight-class egress queuing policy
ASR(config-cmap)# policy-map EIGHT-CLASS-WAN-EDGE
ASR(config-pmap)#  class VOICE
ASR(config-pmap-c)# priority percent 10
 ! Provisions the Voice class with 10% LLQ
ASR(config-pmap-c)# class MULTIMEDIA-CONFERENCING
ASR(config-pmap-c)#  priority percent 23
 ! Provisions the Multimedia Conferencing class with 23% LLQ
ASR(config-pmap-c)# class NETWORK-CONTROL
ASR(config-pmap-c)#  bandwidth percent 5
 ! Provisions the Network Control class with 5% CBWFQ
ASR(config-pmap-c)# class SIGNALING
ASR(config-pmap-c)#  bandwidth percent 2
 ! Provisions the Signaling class with 2% CBWFQ
ASR(config-pmap-c)# class MULTIMEDIA-STREAMING
ASR(config-pmap-c)#  bandwidth percent 10
 ! Provisions the Multimedia Streaming class with 10% CBWFQ
ASR(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing presorter
ASR(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ASR(config-pmap-c)# class TRANSACTIONAL-DATA
ASR(config-pmap-c)#  bandwidth percent 24
 ! Provisions the Transactional Data class with 24% CBWFQ
ASR(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing presorter
ASR(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ASR(config-pmap-c)# class SCAVENGER
ASR(config-pmap-c)#  bandwidth percent 1
 ! Constrains the Scavenger class to 1% CBWFQ
ASR(config-pmap-c)# class class-default
ASR(config-pmap-c)#  bandwidth percent 25
 ! Provisions the default/Best Effort class with 25% CBWFQ
ASR(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing presorter
ASR(config-pmap-c)#  random-detect
 ! Enables WRED



 ! This section applies the queuing policy to an OC-3 ATM PVC
ASR(config)# interface ATM4/1/0.1 point-to-point
ASR(config-if-atm)# pvc 0/112
ASR(config-if-atm-pvc)# vbr-rt 149760 149760
 ! Defines the ATM traffic contract:
 ! Variable Bit Rate - Realtime (OC-3)
ASR(config-if-atm-pvc)# service-policy output EIGHT-CLASS-WAN-EDGE
   ! Attaches egress queuing service policy to the ATM PVC
```

Note This eight-class model features a dual-LLQ design (one for voice and another for multimedia conferencing). This dual-LLQ design prevents bursty video flows from interfering with voice traffic. However, the sum of both LLQs is within the 33 percent best-practice recommendation.

You can verify the configuration in Example 28-8 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Twelve-Class Model

In the 12-class model (illustrated in Figures 11-7 and 11-8), the application class to queue mappings are as follows:

- **Voice (marked EF):** Assigned to a LLQ with 10 percent of strict-priority bandwidth
- **Broadcast video (marked CS5):** Assigned to a LLQ with 10 percent of strict-priority bandwidth
- **Real-time interactive (marked CS4):** Assigned to a LLQ with 13 percent of strict-priority bandwidth
- **Network control (marked CS6):** Assigned to a CBWFQ with 2 percent guaranteed-bandwidth allocation

- **Signaling (marked CS3):** Assigned to a CBWFQ with 2 percent guaranteed-bandwidth allocation
- **Network management (marked CS2):** Assigned to a CBWFQ with 3 percent guaranteed-bandwidth allocation
- **Multimedia conferencing (marked AF4):** Assigned to a CBWFQ with 10 percent guaranteed-bandwidth allocation, with a fair-queuing presorter enabled and with DSCP-based WRED enabled
- **Multimedia streaming (marked AF3):** Assigned to a CBWFQ with 10 percent guaranteed-bandwidth allocation, with a fair-queuing presorter enabled and with DSCP-based WRED enabled
- **Transactional data (marked AF2):** Assigned to a CBWFQ with 10 percent guaranteed-bandwidth allocation, with a fair-queuing presorter enabled and with DSCP-based WRED enabled
- **Bulk data (marked AF1):** Assigned to a CBWFQ with 4 percent guaranteed-bandwidth allocation, with a fair-queuing presorter enabled and with DSCP-based WRED enabled
- **Scavenger (marked CS1):** Assigned to a bandwidth-constrained CBWFQ provisioned at 1 percent
- **Best Effort (marked DF):** Assigned to the default CBWFQ with 25 percent guaranteed-bandwidth allocation, with a fair-queuing presorter enabled and with WRED enabled

Example 28-9 shows the configuration of a 12-class egress queuing policy—with (optional) tuned WRED thresholds—applied to a (OC-48) POS interface.

Example 28-9 *Cisco ASR 1000 12-Class Egress Queuing Model Example*

```
! This section configures the class maps for the
! 12-class egress queuing policy
ASR(config-cmap)# class-map match-all VOICE
ASR(config-cmap)#  match dscp ef
 ! Matches voice traffic on EF (IPv4/IPv6)
ASR(config-cmap)# class-map match-all BROADCAST-VIDEO
ASR(config-cmap)#  match dscp cs5
 ! Matches broadcast video traffic on CS5 (IPv4/IPv6)
ASR(config-cmap)# class-map match-all REALTIME-INTERACTIVE
ASR(config-cmap)#  match dscp cs4
 ! Matches real-time interactive traffic on CS4 (IPv4/IPv6)
ASR(config-cmap)# class-map match-all NETWORK-CONTROL
ASR(config-cmap)#  match dscp cs6
 ! Matches network control traffic on CS6 (IPv4/IPv6)
ASR(config-cmap)# class-map match-all SIGNALING
ASR(config-cmap)#  match dscp cs3
 ! Matches signaling traffic on CS3 (IPv4/IPv6)
ASR(config-cmap)# class-map match-all NETWORK-MANAGEMENT
ASR(config-cmap)#  match dscp cs2
 ! Matches network management traffic on CS2 (IPv4/IPv6)
ASR(config-cmap)# class-map match-any MULTIMEDIA-CONFERENCING
ASR(config-cmap)#  match dscp af41
ASR(config-cmap)#  match dscp af42
ASR(config-cmap)#  match dscp af43
 ! Matches multimedia conferencing traffic on AF4 (IPv4/IPv6)
ASR(config-cmap)# class-map match-any MULTIMEDIA-STREAMING
ASR(config-cmap)#  match dscp af31
ASR(config-cmap)#  match dscp af32
ASR(config-cmap)#  match dscp af33
 ! Matches multimedia streaming traffic on AF3 (IPv4/IPv6)
ASR(config-cmap)# class-map match-any TRANSACTIONAL-DATA
ASR(config-cmap)#  match dscp af21
ASR(config-cmap)#  match dscp af22
ASR(config-cmap)#  match dscp af23
 ! Matches transactional data traffic on AF2 (IPv4/IPv6)
ASR(config-cmap)# class-map match-any BULK-DATA
ASR(config-cmap)#  match dscp af11
ASR(config-cmap)#  match dscp af12
ASR(config-cmap)#  match dscp af13
 ! Matches bulk data traffic on AF1 (IPv4/IPv6)
ASR(config-cmap)# class-map match-all SCAVENGER
ASR(config-cmap)#  match dscp cs1
 ! Matches scavenger traffic on CS1 (IPv4/IPv6)


 ! This section defines the 12-class egress queuing policy
 ! with tuned WRED thresholds
ASR(config-cmap)# policy-map TWELVE-CLASS-WAN-EDGE
ASR(config-pmap)# class VOICE
ASR(config-pmap-c)#  priority percent 10
 ! Provisions the Voice class with 10% LLQ
ASR(config-pmap-c)# class BROADCAST-VIDEO
ASR(config-pmap-c)#  priority percent 10
 ! Provisions the Broadcast Video class with 10% LLQ
ASR(config-pmap-c)# class REALTIME-INTERACTIVE
ASR(config-pmap-c)#  priority percent 13
 ! Provisions the Real-Time Interactive class with 13% LLQ
ASR(config-pmap-c)# class NETWORK-CONTROL
ASR(config-pmap-c)#  bandwidth percent 2
 ! Provisions the Network Control class with 2% CBWFQ
```

```
ASR(config-pmap-c)# class SIGNALING
ASR(config-pmap-c)#  bandwidth percent 2
 ! Provisions the Signaling class with 2% CBWFQ
ASR(config-pmap-c)# class NETWORK-MANAGEMENT
ASR(config-pmap-c)#  bandwidth percent 3
 ! Provisions the Network Management class with 3% CBWFQ
ASR(config-pmap-c)# class MULTIMEDIA-CONFERENCING
ASR(config-pmap-c)#  bandwidth percent 10
 ! Provisions the Multimedia Conferencing class with 10% CBWFQ
ASR(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing presorter
ASR(config-pmap-c)#  queue-limit 350
 ! Sets the maximum queue depth to 350 packets
ASR(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ASR(config-pmap-c)#  random-detect dscp af43 210 350
 ! Tunes minimum DSCP-WRED threshold for AF43 to 60% queue depth
ASR(config-pmap-c)#  random-detect dscp af42 245 350
 ! Tunes minimum DSCP-WRED threshold for AF42 to 70% queue depth
ASR(config-pmap-c)#  random-detect dscp af41 280 350
 ! Tunes minimum DSCP-WRED threshold for AF41 to 80% queue depth
ASR(config-pmap-c)# class MULTIMEDIA-STREAMING
ASR(config-pmap-c)#  bandwidth percent 10
 ! Provisions the Multimedia Streaming class with 10% CBWFQ
ASR(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing presorter
ASR(config-pmap-c)#  queue-limit 350
 ! Sets the maximum queue depth to 350 packets
ASR(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ASR(config-pmap-c)#  random-detect dscp af33 210 350
 ! Tunes minimum DSCP-WRED threshold for AF33 to 60% queue depth
ASR(config-pmap-c)#  random-detect dscp af32 245 350
 ! Tunes minimum DSCP-WRED threshold for AF32 to 70% queue depth
ASR(config-pmap-c)#  random-detect dscp af31 280 350
 ! Tunes minimum DSCP-WRED threshold for AF31 to 80% queue depth
ASR(config-pmap-c)# class TRANSACTIONAL-DATA
ASR(config-pmap-c)#  bandwidth percent 10
 ! Provisions the Transactional Data class with 10% CBWFQ
ASR(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing presorter
ASR(config-pmap-c)#  queue-limit 350
 ! Sets the maximum queue depth to 350 packets
ASR(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
```

```
ASR(config-pmap-c)#  random-detect dscp af23 210 350
 ! Tunes minimum DSCP-WRED threshold for AF23 to 60% queue depth
ASR(config-pmap-c)#  random-detect dscp af22 245 350
 ! Tunes minimum DSCP-WRED threshold for AF22 to 70% queue depth
ASR(config-pmap-c)#  random-detect dscp af21 280 350
 ! Tunes minimum DSCP-WRED threshold for AF21 to 80% queue depth
ASR(config-pmap-c)# class BULK-DATA
 ! Provisions the Bulk Data class with 10% CBWFQ
ASR(config-pmap-c)#  bandwidth percent 4
ASR(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing presorter
ASR(config-pmap-c)#  queue-limit 150
 ! Sets the maximum queue depth to 150 packets
ASR(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ASR(config-pmap-c)#  random-detect dscp af13 90 150
 ! Tunes minimum DSCP-WRED threshold for AF13 to 60% queue depth
ASR(config-pmap-c)#  random-detect dscp af12 105 150
 ! Tunes minimum DSCP-WRED threshold for AF12 to 70% queue depth
ASR(config-pmap-c)#  random-detect dscp af11 120 150
 ! Tunes minimum DSCP-WRED threshold for AF11 to 80% queue depth
ASR(config-pmap-c)# class SCAVENGER
ASR(config-pmap-c)#  bandwidth percent 1
 ! Constrains the Scavenger class to 1% CBWFQ
ASR(config-pmap-c)# class class-default
ASR(config-pmap-c)# bandwidth percent 25
 ! Provisions the default/Best Effort class with 25% CBWFQ
ASR(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing presorter
ASR(config-pmap-c)#  queue-limit 3500
 ! Sets the maximum queue depth to 3500 packets
ASR(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ASR(config-pmap-c)#  random-detect dscp default 2800 3500
 ! Tunes minimum DSCP-WRED threshold for DF to 80% queue depth


 ! This section applies the queuing policy to a (OC-48) POS interface
ASR(config)#interface pos 1/1/1
ASR(config-if)# service-policy output TWELVE-CLASS-WAN-EDGE
 ! Attaches egress queuing service policy to the POS interface


 ! This section applies the queuing policy to the LAN GE interface
ASR(config)#interface Gigabit 0/0/1
ASR(config-if)# service-policy output TWELVE-CLASS-WAN-EDGE
 ! Attaches egress queuing service policy to the GE interface
```

Note The queue limits for the multimedia-conferencing, multimedia-streaming, and transactional data queues were set by the IOS Software (by default on this OC-48 POS interface) at 347 packets; these have been rounded to 350 packets to make the threshold tuning numbers likewise more rounded. Similarly, the bulk data and best effort queues were set by default to queue-limits of 139 packets and 3478 packets (respectively) and have been likewise rounded to 150 packets and 3500 packets.

Note DSCP-based WRED tuning has been done such that AFx3 packets begin randomly dropping at 60 percent of the queue depth, AFx2 packets at 70 percent, and AFx1 packets at 80 percent. In the default queue, packets are all marked DF/DSCP 0 and begin to be dropped only at 80 percent.

Note In this particular design case example, the (OC-48) WAN-edge bandwidth exceeds the (GE) LAN-edge bandwidth; therefore, the same 12-class queuing policy has been applied to the LAN-edge GE interface to prevent drops due to oversubscription in the WAN-to-LAN direction. Admittedly, this is an unusual (and perhaps even extreme) case. However, it serves to highlight an additional design consideration for the administrator to keep in mind.

You can verify the configuration in Example 28-9 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface** (as shown in Example 28-10)

Example 28-10 *Verifying Service Policies: **show policy-map interface***

```
ASR# show policy-map interface pos 1/1/1
 POS1/1/1

  Service-policy output: TWELVE-CLASS-WAN-EDGE

    queue stats for all priority classes:
      Queueing
      queue limit 512 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 4627164/2984695008

    Class-map: VOICE (match-all)
      2955876 packets, 567528192 bytes
      30 second offered rate 14147000 bps, drop rate 0 bps
      Match:  dscp ef (46)
      Priority: 10% (248800 kbps), burst bytes 6220000, b/w exceed drops: 0


    Class-map: BROADCAST-VIDEO (match-all)
      738967 packets, 1058200744 bytes
      30 second offered rate 11457000 bps, drop rate 0 bps
      Match:  dscp cs5 (40)
      Priority: 10% (248800 kbps), burst bytes 6220000, b/w exceed drops: 0

    Class-map: REALTIME-INTERACTIVE (match-all)
      923708 packets, 1322749856 bytes
      30 second offered rate 14322000 bps, drop rate 0 bps
      Match:  dscp cs4 (32)
      Priority: 13% (323440 kbps), burst bytes 8086000, b/w exceed drops: 0


    Class-map: NETWORK-CONTROL (match-all)
      796 packets, 49092 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match:  dscp cs6 (48)
      Queueing
      queue limit 69 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0
      bandwidth 2% (49760 kbps)

    Class-map: SIGNALING (match-all)
      73894 packets, 105816208 bytes
      30 second offered rate 1143000 bps, drop rate 0 bps
      Match:  dscp cs3 (24)
      Queueing
      queue limit 69 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0
      bandwidth 2% (49760 kbps)

    Class-map: NETWORK-MANAGEMENT (match-all)
      147792 packets, 211638144 bytes
      30 second offered rate 2290000 bps, drop rate 0 bps
      Match:  dscp cs2 (16)
      Queueing
      queue limit 104 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0
      bandwidth 3% (74640 kbps)

    Class-map: MULTIMEDIA-CONFERENCING (match-any)
      1477932 packets, 2116398624 bytes
      30 second offered rate 22914000 bps, drop rate 3862000 bps
      Match:  dscp af41 (34)
        640540 packets, 917253280 bytes
        30 second rate 9932000 bps
      Match:  dscp af42 (36)
        320270 packets, 458626640 bytes
        30 second rate 4967000 bps
      Match:  dscp af43 (38)
        517122 packets, 740518704 bytes
        30 second rate 8014000 bps
      Queueing
      queue limit 350 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 67/249090/0/0
      (pkts output/bytes output) 1228842/1759701744
      bandwidth 10% (248800 kbps)
      Fair-queue: per-flow queue limit 87 packets

        Exp-weight-constant: 4 (1/16)
        Mean queue depth: 64 packets
dscp   Transmitted       Random drop      Tail/Flow drop MinimumMaximum Mark
       pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob
af41   533437/763881784  53344/76388608   60142/86123344   280     350   1/10
af42   277738/397720816  27774/39772368   17949/25702968   245     350   1/10
af43   429912/615633984  42992/61564544   49375/70705000   210     350   1/10


    Class-map: MULTIMEDIA-STREAMING (match-any)
      1477935 packets, 2116402920 bytes
      30 second offered rate 22915000 bps, drop rate 3861000 bps
      Match:  dscp af31 (26)
        640539 packets, 917251848 bytes
        30 second rate 9931000 bps
      Match:  dscp af32 (28)
        320271 packets, 458628072 bytes
        30 second rate 4967000 bps
      Match:  dscp af33 (30)
        517125 packets, 740523000 bytes
        30 second rate 8016000 bps
      Queueing
      queue limit 350 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 71/249093/0/0
      (pkts output/bytes output) 1228842/1759701744
      bandwidth 10% (248800 kbps)
      Fair-queue: per-flow queue limit 87 packets

        Exp-weight-constant: 4 (1/16)
        Mean queue depth: 78 packets
dscp   Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
       pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob
af31   534072/764791104  53407/76478824   59443/85122376   280     350   1/10
af32   278117/398263544  27812/39826784   17532/25105824   245     350   1/10
af33   428903/614189096  42890/61418480   50486/72295952   210     350   1/10

    Class-map: TRANSACTIONAL-DATA (match-any)
      1477932 packets, 2116398624 bytes
      30 second offered rate 22915000 bps, drop rate 3856000 bps
      Match:  dscp af21 (18)
        640540 packets, 917253280 bytes
        30 second rate 9932000 bps
      Match:  dscp af22 (20)
        320269 packets, 458625208 bytes
        30 second rate 4967000 bps
      Match:  dscp af23 (22)
        517123 packets, 740520136 bytes
        30 second rate 8016000 bps
      Queueing
      queue limit 350 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 65/249097/0/0
      (pkts output/bytes output) 1228835/1759691720
      bandwidth 10% (248800 kbps)
      Fair-queue: per-flow queue limit 87 packets

        Exp-weight-constant: 4 (1/16)
        Mean queue depth: 65 packets
dscp   Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
       pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob
af21   533571/764073672  53357/76407224   59995/85912840   280     350    1/10
af22   277641/397581912  27764/39758048   18056/25856192   245     350    1/10
af23   429877/615583864  42988/61558816   49409/70753688   210     350    1/10
```

```
    Class-map: BULK-DATA (match-any)
      2216901 packets, 3174602232 bytes
      30 second offered rate 34373000 bps, drop rate 26751000 bps
      Match:  dscp af11 (10)
        640541 packets, 917254712 bytes
        30 second rate 9931000 bps
      Match:  dscp af12 (12)
        320270 packets, 458626640 bytes
        30 second rate 4966000 bps
      Match:  dscp af13 (14)
        1256090 packets, 1798720880 bytes
        30 second rate 19475000 bps
      Queueing
      queue limit 150 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 69/1725401/0/0
      (pkts output/bytes output) 491500/703828000
      bandwidth 4% (99520 kbps)
      Fair-queue: per-flow queue limit 37 packets

        Exp-weight-constant: 84 (1/16)
        Mean queue depth: 0 packets
dscp   Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob
af11   139388/199603616  13938/19959216   493599/706833768  280    350    1/10
af12    90683/129858056   9068/12985376   223710/320352720  245    350    1/10
af13   266331/381385992  26633/38138456   975647/1397126504 210    350    1/10


    Class-map: SCAVENGER (match-all)
      738968 packets, 1058202176 bytes
      30 second offered rate 11459000 bps, drop rate 9552000 bps
      Match:  dscp cs1 (8)
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 64/616084/0
      (pkts output/bytes output) 122884/175969888
      bandwidth 1% (24880 kbps)

    Class-map: class-default (match-any)
      3325441 packets, 4761914000 bytes
      30 second offered rate 51560000 bps, drop rate 3921000 bps
      Match: any
      Queueing
      queue limit 3500 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 0/0/0/0
      (pkts output/bytes output) 0/0
      bandwidth 25% (622000 kbps)
      Fair-queue: per-flow queue limit 875 packets

        Exp-weight-constant: 4 (1/16)
        Mean queue depth: 1005 packets
dscp   Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
       pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob
default 3102802/4443096062 255783/366280146  0/0          2800    3500   1/10

ASR#
```

As shown in Example 28-10, the **show policy-map interface** command verifies that there are no drops occurring on any priority (LLQ) class, which includes the Voice class, the Broadcast Video class, and the Real-Time Interactive class. Neither are there any drops on the Network Control class and the Signaling class. In addition, the command shows that DSCP-WRED is managing congestion in the Multimedia Conferencing, Multimedia Streaming, Transactional Data, and Bulk Data classes. Furthermore, the output displays that the Scavenger class is dropping aggressively, while DSCP-based WRED is controlling congestion in the default class, causing minimal tail-drops within it.

Additional Platform-Specific QoS Design Options

These designs represent the basic elements of WAN aggregator QoS design, but they are by no means the only design options available to you. Additional options and considerations include the following:

- RSVP
- AutoQoS
- Control plane policing

Each of these additional QoS design options and considerations is briefly discussed in turn.

RSVP

Before Resource Reservation Protocol (RSVP) can be effectively enabled on a per-interface basis; the IOS RSVP Agent functionality must be enabled on a router. The Cisco RSVP Agent is a Cisco IOS feature that enables Cisco Unified Communications CallManager (CUCM) to perform RSVP-based call admission control. The Cisco RSVP Agent feature was introduced in Cisco IOS Release 12.4(6)T.

Note Detailed configuration guidance and best practices for the IOS RSVP Agent is beyond the scope of this chapter, because it is very CUCM oriented. Design guidance for IOS RSVP Agent is detailed in the *Cisco Unified Communications System 8.x SRND* in Chapter 11 at http://www.cisco.com/en/US/docs/voice_ip_comm/cucm/srnd/8x/cac.html.

After the IOS RSVP Agent has been properly configured to communicate with the CUCM to enable admission control for voice/video policies, the administrator can focus on interface-specific RSVP design. Two RSVP models will now be presented: a basic RSVP model, and a more advanced model that features application ID functionality.

Basic RSVP Model

After the IOS RSVP Agent has been configured, basic RSVP functionality is enabled by the **ip rsvp bandwidth** interface configuration command, which specifies how much bandwidth may be explicitly reserved by RSVP requests. (The default is 75 percent of a link's bandwidth.)

If all the PQ traffic is RSVP-enabled, the value specified in the **ip rsvp bandwidth** command and the priority LLQ command should match once Layer 2 overhead of the priority queue bandwidth has been taken into account. But if some PQ traffic is not RSVP enabled, ensure that the sum of the values specified in the **ip rsvp bandwidth** command and in the out-of-band call admission control mechanism do not exceed the bandwidth value specified in the **priority** command.

Also, if RSVP is enabled on one or more interfaces of a router, all interfaces through which you expect RSVP signaling to transit should also be enabled for RSVP to ensure that RSVP messages do not get dropped.

RSVP should be enabled at the edge of the network, including the router WAN interfaces on both sides of the WAN link, at all possible WAN congestion points, including redundant links of different speeds.

When deploying RSVP in the IP WAN, it is recommended to use the integrated/differentiated services (IntServ/DiffServ) model, such that RSVP's only application is to perform admission control, while DiffServ policies will perform classification, marking, policing, and scheduling. This model scales much better (because RSVP does not have to maintain state of all flows) and also better integrates with the DiffServ policies presented in this design chapter. Configuring the IntServ/DiffServ model requires two additional interface configuration commands: **ip rsvp resource provider none** and **ip rsvp data-packet classification none**. These two additional commands instruct RSVP not to perform QoS operations that are handled within the data plane by DiffServ policies.

Also it is recommended to use the **ip rsvp signaling dscp** interface configuration command to ensure that RSVP packets are marked to CS3/24, inline with other signaling flows (as by default these packets will be marked to 63).

Example 28-11 shows a basic IntServ/DiffServ RSVP model.

Example 28-11 *Cisco ASR 1000 Basic RSVP Model Example*

```
 ! This section configures basic IntServ/DiffServ RSVP
 ! on a WAN interface
ASR(config)# interface Serial2/1/0
ASR(config-if)# description CAMPUS-TO-BRANCH-SERIAL-T3-WITH-RSVP
ASR(config-if)# bandwidth 44210
ASR(config-if)# ip address 10.0.12.5 255.255.255.252
ASR(config-if)# load-interval 30
 ! Minimizes the interface-statistics sampling-period (optional)
ASR(config-if)# tx-ring-limit 10
 ! Optimizes the T3 Tx-Ring for rich media applications
ASR(config-if)# service-policy output WAN-EDGE-DIFFSERV-POLICY
 ! Attaches the DiffServ MQC policy to the interface
ASR(config-if)# ip rsvp bandwidth 15000
 ! Specifies the amount of reservable BW (should match LLQ BW)
ASR(config-if)# ip rsvp signalling dscp 24
 ! Marks RSVP signaling traffic to CS3
ASR(config-if)# ip rsvp data-packet classification none
 ! Enables the IntServ/DiffServ model by disabling RSVP for classification
ASR(config-if)# ip rsvp resource-provider none
 ! Enables the IntServ/DiffServ model by disabling RSVP for scheduling
```

This configuration can be verified with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**
- **show interface**
- **show ip rsvp interface** (as shown in Example 28-12)
- **show ip rsvp interface detail** (as shown in Example 28-13)
- **show ip rsvp installed** (as shown in Example 28-14)

Example 28-12 *Verifying RSVP Policies:* ***show ip rsvp interface***

```
ASR# show ip rsvp interface serial 2/1/0
interface    rsvp  allocated  i/f max  flow max sub max  VRF
Se2/1/0       ena   9996k         15M       15M       0
```

As shown in Example 28-12, this interface has been allocated 15 Mbps of bandwidth that may be reserved via RSVP, of which currently 9996 Kbps is being allocated.

Example 28-13 *Verifying RSVP Policies:* ***show ip rsvp interface detail***

```
ASR# show ip rsvp interface detail serial 2/1/0
 Se2/1/0:
   RSVP: Enabled
   Interface State: Up
   Bandwidth:
     Curr allocated: 9996k bits/sec
     Max. allowed (total): 15M bits/sec
     Max. allowed (per flow): 15M bits/sec
     Max. allowed for LSP tunnels using sub-pools: 0 bits/sec
     Set aside by policy (total): 0 bits/sec
   Admission Control:
     Header Compression methods supported:
       rtp (36 bytes-saved), udp (20 bytes-saved)
   Traffic Control:
     RSVP Data Packet Classification is OFF
     RSVP resource provider is: none
   Signalling:
     DSCP value used in RSVP msgs: 0x18
     Number of refresh intervals to enforce blockade state: 4
   Authentication: disabled
     Key chain:   <none>
     Type:        md5
     Window size: 1
     Challenge:   disabled
   Hello Extension:
     State: Disabled
```

As implied by the additional keyword—and as shown in Example 28-13—the **show ip rsvp interface detail** provides more detail about the basic RSVP policy. For example, it verifies that 9996 Kbps of a maximum of 15 Mbps is currently being reserved and allocated to media flows. Also it confirms that there are no data classification nor resource provider functions being performed by RSVP (because RSVP is operating in the IntServ/DiffServ model). And finally, that RSVP packets are being marked to CS3/24 (which is expressed in hexadecimal as 0x18).

Example 28-14 *Verifying RSVP Policies:* ***show ip rsvp installed***

```
ASR# show ip rsvp installed
80K    10.17.100.101    10.16.255.101   UDP    31257  26645
80K    10.17.100.102    10.16.255.102   UDP    16773  21718
80K    10.17.100.103    10.16.255.103   UDP    26470  16878
80K    10.17.100.104    10.16.255.104   UDP    18115  17845
```

```
80K    10.17.100.105    10.16.255.105   UDP    25294  22658
80K    10.17.100.106    10.16.255.106   UDP    24613  26904
80K    10.17.100.107    10.16.255.107   UDP    23175  26733
80K    10.17.100.108    10.16.255.108   UDP    22139  23492
80K    10.17.100.109    10.16.255.109   UDP    25590  29755
80K    10.17.100.110    10.16.255.110   UDP    27854  32460
80K    10.17.100.111    10.16.255.111   UDP    24189  29516
80K    10.17.100.112    10.16.255.112   UDP    22395  17433
753K   10.17.100.101    10.16.255.101   UDP    21502  28165
753K   10.17.100.102    10.16.255.102   UDP    17781  30069
753K   10.17.100.103    10.16.255.103   UDP    29918  25300
753K   10.17.100.104    10.16.255.104   UDP    29707  28284
753K   10.17.100.105    10.16.255.105   UDP    17780  19823
753K   10.17.100.106    10.16.255.106   UDP    28301  20751
753K   10.17.100.107    10.16.255.107   UDP    31230  19925
753K   10.17.100.108    10.16.255.108   UDP    20067  32170
753K   10.17.100.109    10.16.255.109   UDP    30677  17868
753K   10.17.100.110    10.16.255.110   UDP    22281  20874
753K   10.17.100.111    10.16.255.111   UDP    24056  16900
753K   10.17.100.112    10.16.255.112   UDP    22203  32146
```

Example 28-14 breaks down the composition of the 9996 Kbps of bandwidth that is currently being reserved and allocated to media flows, showing that there are a dozen voice calls (at 80 Kbps each) and a dozen video calls (at 753 Kbps each) that are currently active across the interface.

Advanced RSVP Model with Application ID

The RSVP Local Policy feature provides the mechanism for controlling a reservation based on an application ID. Application IDs are mapped to RSVP local policies through the **ip rsvp policy identity** command. RSVP local policy identities are defined globally and are available to each interface for policy enforcement. Each identity can have one policy locator defined to match an application ID.

To give the user as much flexibility as possible in matching application policy locators to local policies, the RSVP local policy CLI accepts application ID match criteria in the form of UNIX-style regular expressions for the policy locator. Regular expressions are already used in the CLI for existing Cisco IOS components such as Border Gateway Protocol (BGP).

Note For more information on how regular expressions can be used in Cisco IOS, see http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094a92.shtml.

As previously mentioned, CUCM can set RFC 2872 application IDs for both voice and video flows as cluster-wide service parameters. By default, the application IDs used by CUCM are AudioStream for voice flows and VideoStream for video flows (for both audio and video components).

Therefore, the globally defined IOS RSVP policy identities that match these CUCM application IDs for voice and video are (respectively) are as follows:

- **ip rsvp policy identity rsvp-voice policy-locator .*AudioStream.***
- **ip rsvp policy identity rsvp-video policy-locator .*VideoStream.***

In turn, local policies based on application IDs are applied to an interface using the **ip rsvp policy local identity** interface command. For reservations that match its policy locator value, a local policy has the ability to perform the following functions:

- Define the maximum amount of bandwidth the reservations can reserve as a group or as a single sender
- Forward or not forward RSVP messages
- Accept or not accept RSVP messages
- Define the maximum bandwidth the group or sender can reserve

There is also a catchall local policy called the default local policy. This local policy will match any RSVP reservation that did not match the other RSVP local policies configured on the link. The default local policy can be used to match reservations that are not tagged with an application ID or reservations that are tagged with an application ID that is to be treated as untagged traffic.

Example 28-15 shows an advanced RSVP model featuring local policies for voice and video application IDs.

Example 28-15 *Cisco ASR 1000 Advanced (Application ID) RSVP Model Example*

```
 ! This section defines the regular expressions to match RSVP
 ! application IDs
ASR(config)# ip rsvp policy identity RSVP-VIDEO policy-locator .*VideoStream.*
 ! RSVP AppIDs with the string "VideoStream" will be
 ! Associated with the RSVP-VIDEO local RSVP policy
ASR(config)# ip rsvp policy identity RSVP-VOICE policy-locator .*AudioStream.*
 ! RSVP AppIDs with the string "AudioStream" will be
 ! Associated with the RSVP-VIDEO local RSVP policy


 ! This section configures advanced AppID RSVP on a WAN interface
ASR(config)# interface Serial2/1/0
ASR(config-if)# description CAMPUS-TO-BRANCH-SERIAL-T3-WITH-RSVP
ASR(config-if)# bandwidth 44210
ASR(config-if)# load-interval 30
 ! Minimizes the interface-statistics sampling-period (optional)
ASR(config-if)# tx-ring-limit 10
 ! Optimizes the T3 Tx-Ring for rich media applications
ASR(config-if)# service-policy output WAN-EDGE-DIFFSERV-POLICY
 ! Attaches the DiffServ MQC policy to the interface
ASR(config-if)# ip rsvp policy local identity RSVP-VIDEO
ASR(config-rsvp-local-if-policy)#  maximum bandwidth group 12500
ASR(config-rsvp-local-if-policy)#  forward all
 ! Defines a local RSVP policy to admit up to 12.5 Mbps of video flows
ASR(config-if)# ip rsvp policy local identity RSVP-VOICE
ASR(config-rsvp-local-if-policy)#  maximum bandwidth group 2500
ASR(config-rsvp-local-if-policy)#  forward all
 ! Defines a local RSVP policy to admit up to 2.5 Mbps of voice flows
ASR(config-if)# ip rsvp bandwidth 15000
 ! Specifies the amount of reservable BW (should match LLQ BW)
ASR(config-if)# ip rsvp signalling dscp 24
 ! Marks RSVP signaling traffic to CS3
ASR(config-if)# ip rsvp data-packet classification none
 ! Enables the IntServ/DiffServ model by disabling RSVP for
 ! classification
ASR(config-if)# ip rsvp resource-provider none
 ! Enables the IntServ/DiffServ model by disabling RSVP for scheduling
```

You can verify the configuration in Example 28-15 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**
- **show interface**
- **show ip rsvp interface**
- **show ip rsvp interface detail**
- **show ip rsvp installed**
- **show ip rsvp policy local** (as shown in Example 28-16)
- **show ip rsvp policy local detail** (as shown in Example 28-17)

Example 28-16 *Verifying RSVP Policies:* ***show ip rsvp policy local***

```
ASR# show ip rsvp policy local
    A=Accept    F=Forward
    Serial2/1/0:
      Path:AF Resv:AF PathErr:AF ResvErr:AF ID(s): RSVP-VIDEO
      Path:AF Resv:AF PathErr:AF ResvErr:AF ID(s): RSVP-VOICE
Generic policy settings:
    Default policy: Accept all
    Preemption:     Disabled
```

Example 28-17 verifies that the local RSVP policy will accept and forward flows on Serial 2/1/0 that match the application IDs defined in the RSVP-VIDEO and RSVP-VOICE policy identities. In addition, it confirms that the default policy is to accept all other RSVP requests and that preemption is disabled.

Example 28-17 *Verifying RSVP Policies:* ***show ip rsvp policy local detail***

```
ASR# show ip rsvp policy local detail
Serial2/1/0:
  Policy for ID(s): RSVP-VIDEO
    Preemption Scope: Unrestricted.
    Local Override:   Disabled.
    Fast ReRoute:     Accept.
    Handle:           01000406.
                           Accept               Forward
    Path:                  Yes                  Yes
    Resv:                  Yes                  Yes
    PathError:             Yes                  Yes
    ResvError:             Yes                  Yes
                           Setup Priority       Hold Priority
    TE:                    N/A                  N/A
    Non-TE:                N/A                  N/A
                           Current              Limit
    Senders:               12    N/A
    Receivers:             12    N/A
    Conversations:         12    N/A
    Group bandwidth (bps): 9036K                12500K
    Per-flow b/w (bps):    753K                 N/A

  Policy for ID(s): RSVP-VOICE
    Preemption Scope: Unrestricted.
    Local Override:   Disabled.
    Fast ReRoute:     Accept.
    Handle:          02000403.
                          Accept             Forward
    Path:                 Yes                Yes
    Resv:                 Yes                Yes
    PathError:            Yes                Yes
    ResvError:            Yes                Yes
                          Setup Priority     Hold Priority
    TE:                   N/A                N/A
    Non-TE:               N/A                N/A
                          Current            Limit
    Senders:              12                N/A
    Receivers:            12                N/A
    Conversations:        12                N/A
    Group bandwidth (bps): 960K             2500K
    Per-flow b/w (bps):   80K               N/A
Generic policy settings:
    Default policy: Accept all
    Preemption:     Disabled
```

Example 28-17 provides additional detail into each local policy highlighting that there are 12 video flows (at 753 Kbps each) and 12 voice flows (at 80 Kbps each) that are currently allocated to the video and voice pools, respectively.

AutoQoS SRND4

As noted in the previous chapter, AutoQoS SRND4 is not supported in Cisco IOS for ASR 1000 routers (at the time of this writing). However, AutoQoS-VoIP and AutoQoS-Enterprise are supported on this platform.

Control Plane Policing

Control plane policing is supported in Cisco IOS, and design recommendations for this feature are detailed in Appendix B, “Control Plane Policing.”

Summary

This chapter outlined the QoS roles of a WAN aggregator router and also identified the Cisco ASR 1000 as being well suited to this role.

Then the Cisco ASR 1000 hardware architecture was overviewed, specifically as it relates to QoS—highlighting scenarios where the internal system may be oversubscribed. Internal (PLIM QoS) mechanisms were presented to show how potential internal oversubscription could be managed on this platform.

Ingress QoS designs were mentioned in passing (because these are more typically deployed in the branch and are discussed in depth in the following chapter). However, these could likewise be deployed at the WAN aggregator LAN edges.

Next, egress QoS design was presented, focusing on 4-, 8- and 12-class queuing models. These designs took advantage of many Cisco IOS QoS features, such as LLQ, CBWFQ, DSCP-based WRED, and fair-queue presorting.

Additional QoS design options were also discussed, such as RSVP, where basic and advanced RSVP designs were presented as additional design options.

Further Reading

Medianet WAN Aggregation QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html

ASR 1000 QoS Architecture: http://www.cisco.com/en/US/prod/collateral/routers/ps9343/solution_overview_c22-449961_ps9343_Product_Solution_Overview.html

Classifying and Scheduling Packets for ASR 1000 Series: http://www.cisco.com/en/US/partner/docs/interfaces_modules/shared_port_adapters/configuration/ASR1000/ASRimpqos.html (requires a Cisco.com account)

Cisco IOS Quality of Service Solutions Configuration Guide Library: Cisco IOS Release 15M&T: http://www.cisco.com/en/US/docs/ios-xml/ios/qos/config_library/15-mt/qos-15-mt-library.html

Cisco Unified Communications System 8.x SRND: http://www.cisco.com/en/US/docs/voice_ip_comm/cucm/srnd/8x/cac.html.

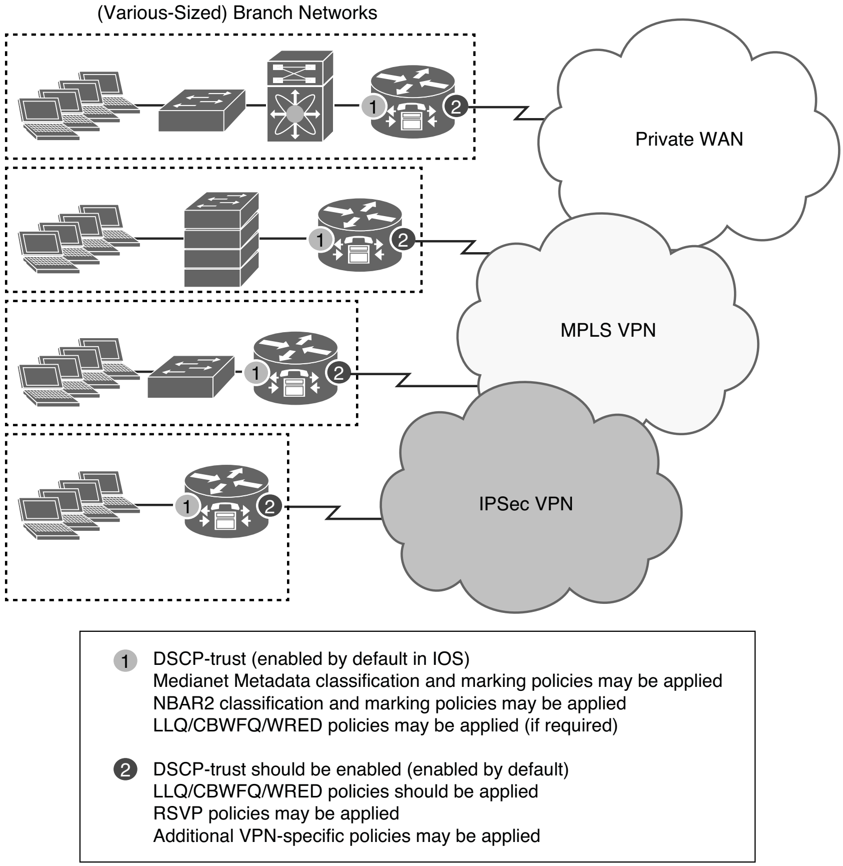
Chapter 29

Branch Router (Cisco ISR G2) QoS Design

Like the WAN aggregator, the primary quality of service (QoS) role of the branch router is to manage the inevitable congestion caused by the major reduction in link speeds from the LAN to the WAN, while minimizing packet loss and jitter.

A secondary role of these routers may be to perform deep packet inspection (DPI) for ingress application classification. DPI is especially relevant on today's networks because of the thousands of applications that run over HTTP, which, from a Layer 4 perspective all look the same (as TCP traffic over port 80). It is only by looking deeper into the packet and identifying Layer 7 parameters that distinctions can be made for these flows. There are limits to DPI classification that administrators should be aware of, however, such as the inability to discern traffic that is encrypted via HTTPS or tunneled traffic (such as Control And Provisioning of Wireless Access Points [CAPWAP]). Branch routers connect not only to private WAN networks but may also connect to MPLS or IPsec-based VPN networks. Figure 29-1 illustrates the interface-specific QoS roles of branch routers.

The Cisco ISR G2 series routers are tailor designed to the role of branch routers. Therefore, to begin, we will briefly review this platform's architecture.



1 DSCP-trust (enabled by default in IOS)
Medianet Metadata classification and marking policies may be applied
NBAR2 classification and marking policies may be applied
LLQ/CBWFQ/WRED policies may be applied (if required)

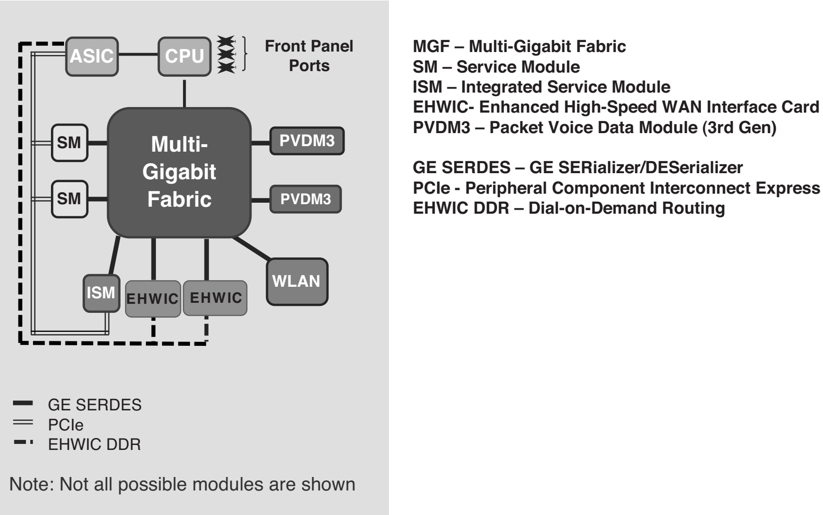
2 DSCP-trust should be enabled (enabled by default)
LLQ/CBWFQ/WRED policies should be applied
RSVP policies may be applied
Additional VPN-specific policies may be applied

Figure 29-1 *Branch Router Interface QoS Roles*

Cisco ISR G2 QoS Architecture

The Cisco ISR G2 series routers deliver a significant performance capacity over the first-generation Integrated Services Routers (ISR) (up to seven times in some models as compared to their first-generation counterparts). This is not only due to faster processors and more memory, but also due to a switch from a single-processor design to multicore processor designs. In addition, the system architecture has been redesigned around a Multi-Gigabit Fabric (MGF), as illustrated in Figure 29-2.

Although the ISR G2 does have a powerful array of hardware within its architecture, it bears repeating that it performs QoS operations in software (and not in dedicated hardware application-specific integrated circuit [ASICs]). Therefore, the performance chart shown in Table 27-1—or better yet Cisco.com—should be consulted when selecting the model of ISR G2 router to deploy at a given branch.



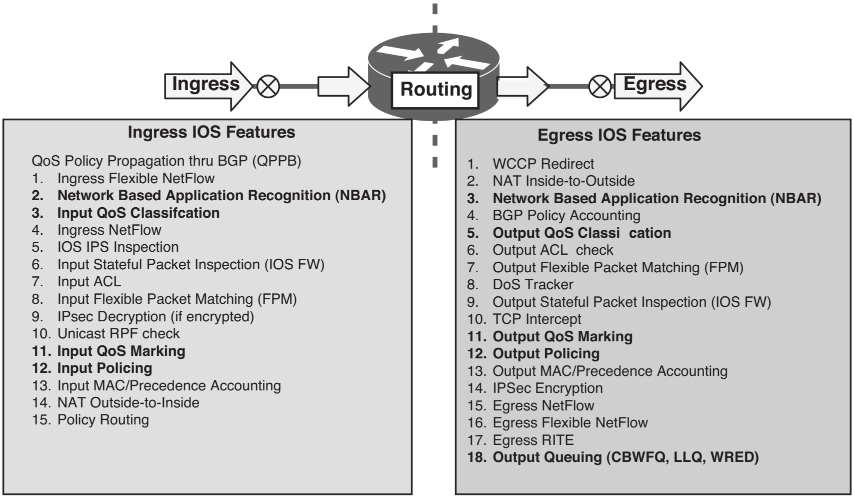
MGF – Multi-Gigabit Fabric
SM – Service Module
ISM – Integrated Service Module
EHWIC- Enhanced High-Speed WAN Interface Card
PVDM3 – Packet Voice Data Module (3rd Gen)

GE SERDES – GE SERializer/DESerializer
PCIe - Peripheral Component Interconnect Express
EHWIC DDR – Dial-on-Demand Routing

Figure 29-2 *Cisco ISR G2 Hardware Architecture*

Because all QoS operations are performed in IOS Software, it may be helpful to understand the order that these operations are performed in. Figure 29-3 shows the order-of-operations, not only of QoS features but also of other general IOS features to show their interrelationship.

Note Not all IOS features are shown in Figure 29-3.



Ingress IOS Features

QoS Policy Propagation thru BGP (QPPB)

1. Ingress Flexible NetFlow
2. **Network Based Application Recognition (NBAR)**
3. **Input QoS Classifcation**
4. Ingress NetFlow
5. IOS IPS Inspection
6. Input Stateful Packet Inspection (IOS FW)
7. Input ACL
8. Input Flexible Packet Matching (FPM)
9. IPsec Decryption (if encrypted)
10. Unicast RPF check
11. **Input QoS Marking**
12. **Input Policing**
13. Input MAC/Precedence Accounting
14. NAT Outside-to-Inside
15. Policy Routing

Egress IOS Features

1. WCCP Redirect
2. NAT Inside-to-Outside
3. **Network Based Application Recognition (NBAR)**
4. BGP Policy Accounting
5. **Output QoS Classification**
6. Output ACL check
7. Output Flexible Packet Matching (FPM)
8. DoS Tracker
9. Output Stateful Packet Inspection (IOS FW)
10. TCP Intercept
11. **Output QoS Marking**
12. **Output Policing**
13. Output MAC/Precedence Accounting
14. IPSec Encryption
15. Egress NetFlow
16. Egress Flexible NetFlow
17. Egress RITE
18. **Output Queuing (CBWFQ, LLQ, WRED)**

Figure 29-3 *Cisco IOS Order of Operations*

QoS Design Steps

There are two main steps to configure QoS on a Cisco ISR G2 router performing the role of a branch router:

1. Configure ingress QoS models, including the following:
   - Trust DSCP (enabled by default in IOS)
   - Medianet metadata classification and marking
   - NBAR2 classification and marking
2. Configure egress QoS policies.

Each of these design steps is covered in turn.

Ingress QoS Models

As previously mentioned, classification and marking should be performed as close to the edge as possible. However, Catalyst switches have limited ability (if any) to perform DPI, such as Medianet metadata flow identification or NBAR2. Therefore, there may be a need to configure ingress DPI classification policies on the LAN edge of the branch router.

Note As noted in the previous chapter, these ingress DPI policies may also be deployed on the LAN edges of the WAN aggregation routers (with identical configurations).

Medianet and Network Based Application Recognition 2 (NBAR2) classification models are discussed in turn.

Medianet Classification Models

Cisco IOS can perform DPI via Medianet flow metadata or Medianet NBAR2 integration (as discussed in Chapter 8, “Medianet”).

Medianet metadata classification can be performed using three different approaches:

- **Application-based classification:** Each **match application** command references a specific application that is to be matched on.
- **Application-group-based classification:** The **match application** command is combined with the **application-group** keyword, which allows subcomponents of an application to be represented with a single **match** statement.
- **Attribute-based classification:** The **match application** command is combined with the **attribute** keyword, which allows for several applications with similar characteristics (attributes) to be represented with a single **match** statement.

Each Medianet metadata classification option will be now be presented in detail.

Medianet Application-Based Classification and Marking Model

In the application-based classification model, individual application names must be specified for each **match application** statement (which effectively invokes NBAR2 for classification). The application-based classification model is more granular and specific, but can lead to longer and bulkier policies (as compared to application-group or attribute-based classification models, which are presented in turn).

Example 29-1 shows an example Medianet metadata application-based classification and marking policy that may be deployed on the ingress LAN edge of the branch router.

Example 29-1 *Cisco ISR G2 Medianet Metadata (Application-Based) Ingress Classification and Marking Model*

```
 ! This section globally enables Medianet metadata
ISR-G2(config)# metadata flow
 ! Globally enables Medianet metadata


 ! This section configures the metadata application-based class maps
ISR-G2(config-cmap)# class-map match-any MEDIANET-VOICE
ISR-G2(config-cmap)#  match application cisco-phone
 ! Identifies Cisco IP and Softphone voice flows via metadata

ISR-G2(config-cmap)# class-map match-all MEDIANET-REALTIME-INTERACTIVE
ISR-G2(config-cmap)#  match application telepresence-media
 ! Identifies TelePresence media flows via metadata

ISR-G2(config-cmap)# class-map match-any MEDIANET-MULTIMEDIA-CONFERENCING
ISR-G2(config-cmap)#  match application webex-voice
 ! Identifies WebEx voice flows via application metadata
ISR-G2(config-cmap)#  match application webex-video
 ! Identifies WebEx video flows via application metadata
ISR-G2(config-cmap)#  match application webex-meeting
 ! Identifies WebEx data flows via application metadata

ISR-G2(config-cmap)# class-map match-any MEDIANET-SIGNALING
ISR-G2(config-cmap)#  match application sip
 ! Identifies SIP signaling flows via application metadata
ISR-G2(config-cmap)#  match application telepresence-control
 ! Identifies TelePresence signaling flows via application metadata
ISR-G2(config-cmap)#  match application h323
 ! Identifies H.323 signaling flows via application metadata
ISR-G2(config-cmap)#  match application rtsp
```

```
 ! Identifies RTSP signaling flows via application metadata

ISR-G2(config-cmap)# class-map match-any MEDIANET-TRANSACTIONAL-DATA
ISR-G2(config-cmap)#  match application citrix
 ! Identifies Citrix flows via application metadata
ISR-G2(config-cmap)#  match application vmware-view
 ! Identifies VMware View flows via application metadata
ISR-G2(config-cmap)#  match application wyze-zero-client
 ! Identifies WYZE Zero Client flows via application metadata



 ! This section configures the ingress marking policy map
ISR-G2(config-cmap)# policy-map MEDIANET-MARKING
ISR-G2(config-pmap)#  class MEDIANET-VOICE
ISR-G2(config-pmap-c)#   set dscp ef
 ! Marks IPv4/IPv6 DSCP values to EF for voice
ISR-G2(config-pmap-c)#  class MEDIANET-REALTIME-INTERACTIVE
ISR-G2(config-pmap-c)#   set dscp cs4
 ! Marks IPv4/IPv6 DSCP values to CS4 for TelePresence
ISR-G2(config-pmap-c)#  class MEDIANET-MULTIMEDIA-CONFERENCING
ISR-G2(config-pmap-c)#   set dscp af41
 ! Marks IPv4/IPv6 DSCP values to AF41 for multimedia conferencing
ISR-G2(config-pmap-c)#  class MEDIANET-SIGNALING
ISR-G2(config-pmap-c)#   set dscp cs3
 ! Marks IPv4/IPv6 DSCP values to CS3 for Signaling class
ISR-G2(config-pmap-c)#  class MEDIANET-TRANSACTIONAL-DATA
ISR-G2(config-pmap-c)#   set dscp af21
 ! Marks IPv4/IPv6 DSCP values to AF21 for transactional data
ISR-G2(config-pmap-c)#  class class-default
ISR-G2(config-pmap-c)#   set dscp default
 ! Marks IPv4/IPv6 DSCP values to DF/DSCP0 for everything else



 ! This section applies the ingress (application-based)
 ! Medianet metadata marking policy to the interface(s)
ISR-G2(config)# interface GigabitEthernet0/1
ISR-G2(config-if)# service-policy input MEDIANET-MARKING
```

Note It is necessary to provide unique names for these class maps that are matching these application classes by Medianet application metadata (as opposed to WAN-edge class maps—presented in the previous chapter and also later in this chapter—that match these application classes by differentiated services code point [DSCP] markings);

otherwise, one or the other set of class maps will be overwritten at some point. To this end, the MEDIANET- prefix is used in these examples.

You can verify the configuration in Example 29-1 with the following commands:

- **show metadata application table** (as shown in Example 29-2)
- **show metadata flow classification table** (as shown in Example 29-3)
- **show metadata flow statistics** (as shown in Example 29-4)
- **show metadata flow table** (as shown in Example 29-5)
- **show class-map**
- **show policy-map**
- **show policy-map interface**

Example 29-2 *Verifying Metadata Applications:* ***show metadata application table***

```
ISR-G2# show metadata application table
ID     Name                  Vendor      Version
--------------------------------------------------------------------------------
218103921  telepresence-media    -           -
218103922  telepresence-contr$   -           -
218104286  telepresence-data     -           -
218104220  webex-voice           -           -
218104221  webex-video           -           -
218104222  webex-meeting         -           -
218103864  citrix                -           -
218103889  cisco-phone           -           -
218104284  vmware-view           -           -
218104281  wyze-zero-client      -           -
218103869  rtp                   -           -
218103872  h323                  -           -
218108868  sip                   -           -
218104362  rtsp                  -           -
```

Example 29-2 shows the various applications supported by the current version of IOS Software for Medianet metadata classification.

Example 29-3 *Verifying Metadata Flow Classifications:* ***show metadata flow classification table***

```
ISR-G2# show metadata flow classification table


Target          Flow ID    Dir   Policy     Filter(s)
                                 Type
---------------+----------+-----+----------+---------------------------

Gig0/0           5          OUT   PM         application webex-meeting vendor
Cisco Systems, Inc. version 1.4.5
                                  QOS        application webex-meeting vendor
Cisco Systems, Inc. version 1.4.5
Gig0/1           3          OUT
Gig0/2           5          IN
```

Example 29-3 shows various dynamic flows that have been classified via Medianet metadata classification, which in this case includes some Cisco WebEx flows.

Example 29-4 *Verifying Metadata Flow Statistics:* ***show metadata flow statistics***

```
ISR-G2# show metadata flow statistics

Interface specific report :

GigabitEthernet0/0: Ingress flows 2, Egress flows 2
GigabitEthernet0/1: Ingress flows 0, Egress flows 1
GigabitEthernet0/2: Ingress flows 1, Egress flows 0

Chunk statistics:

Type                Allocated       Returned        Failed

IPV4 Flow            3                0                0
Flow Key             3                1                0
Source List          1                0                0
Flow Info            3                3                0
Attribute Data       3                3                0
Feature Object       0                0                0

Event Statistics:

Add Flow                 : 3         Delete Flow              : 0
```

```
Received                    : 3        Rejected                  : 0
Transient                   : 0        Posted                    : 3
Ingress Change              : 0        Egress Change             : 0
Unknown                     : 0        Source Limit Exceeded     : 0
```

Example 29-4 shows various dynamic flow and event counters relating to Medianet applications.

Example 29-5 *Verifying Metadata Flow Statistics:* ***show metadata flow statistics***

```
ISR-G2# show metadata flow table

Flow  To               From             Protocol DPort SPort Ingress I/F Egress I/F
SSRC


2     209.165.202.129  209.165.202.150  UDP      49152 5060  Gi0/0       0
1     209.165.202.150  209.165.202.129  UDP      5060  49152 Gi0/0       0
3     209.165.202.150  209.165.202.129  UDP      5062  49154 Gi0/1       0
4     209.165.202.129  209.165.202.150  UDP      49154 5062  Gi0/2       0
```

Example 29-5 provides detailed flow information about Medianet flows, including source and destination IP address pairings, Layer 4 protocols and ports, and also the ingress interfaces of the flows.

Medianet Application-Group-Based Classification Model

In the attribute-based classification model, the **match application** command is combined with the **application-group** keyword, which allows for application subcomponents to be grouped together via a single **match** statement.

Example 29-6 shows an example Medianet metadata application-group-based classification class map for the Multimedia Conferencing class. In the previous configuration example (Example 29-1), three separate **match** statements were required to match the voice, video, and data subcomponents of the Cisco WebEx application for this class. However, these can be simplified to a single **match application application-group** statement, as demonstrated in Example 29-6.

Note For the sake of minimizing redundancy, only a single class map is shown in the application-group-based and attribute-based classification model examples that follow.

Example 29-6 *Cisco ISR G2 Medianet Metadata (Application-Group-Based) Ingress Classification Model*

```
ISR-G2(config)# class-map match-all MEDIANET-MULTIMEDIA-CONFERENCING
ISR-G2(config-cmap)#  match application application-group webex-group
```

You can verify the configuration in Example 29-6 with the **show class-map** command.

Medianet Attribute-Based Classification Model

In the attribute-based classification model, the **match application** command is combined with the **attribute** keyword that allows for several applications with similar characteristics/attributes to be grouped together via a single **match** statement.

Example 29-7 shows an example Medianet metadata attribute-based classification class map for the Transactional Data class. In the original configuration example (Example 29-1), three separate **match** statements were required to match Citrix, VMware, and Wyse virtual desktop protocols. However, these can be simplified to a single **match application attribute** statement, as demonstrated here.

Example 29-7 *Cisco ISR G2 Medianet Metadata (Attribute-Based) Ingress Classification Model*

```
ISR-G2(config)# class-map match-all MEDIANET-TRANSACTIONAL-DATA
ISR-G2(config-cmap)# match application attribute category business-and-productivity-
tools
```

You can verify the configuration in Example 29-7 with the **show class-map** command.

NBAR2 Classification Models

As previously discussed, NBAR2 is a re-architecture of NBAR based on the Service Control Engine (SCE) but with advanced classification techniques, accuracy, and many more signatures. Currently, NBAR2 supports well over a thousand applications and subclassifications. In addition, Cisco provides new signatures (and signatures updates) through monthly released protocol packs.

Similar to Medianet metadata classification, NBAR2 classification can be performed using three different approaches:

- **Application-based classification:** Each **match protocol** command references a specific application that is to be matched on.
- **Application-group-based classification:** The **match protocol** command is combined with the **application-group** keyword, which allows groups of applications to be represented with a single **match** statement.

- **Attribute-based classification:** The **match protocol** command is combined with the **attribute** keyword, which allows for several applications with similar characteristics (attributes) to be represented with a single **match** statement.

The sections that follow cover each NBAR2 classification option in detail.

NBAR2 Application-Based Classification and Marking Model

In the application-based classification model, individual application names must be specified for each **match protocol** statement. The application-based classification model is more granular and specific, but can lead to longer and bulkier policies (as compared to application-group or attribute-based classification models, which are presented in turn).

Example 29-8 shows an example NBAR2 application-based classification and marking policy that may be deployed on the ingress LAN edge of the branch router.

Example 29-8 *Cisco ISR G2 Application-Based NBAR2 Classification and Marking Model*

```
 ! This section configures the NBAR2 application-based class maps
ISR-G2(config-cmap)# class-map match-all NBAR2-VOICE
ISR-G2(config-cmap)#  match protocol cisco-phone
 ! Identifies Cisco IP Phone VoIP traffic via NBAR2


ISR-G2(config-cmap)# class-map match-all NBAR2-REALTIME-INTERACTIVE
ISR-G2(config-cmap)#  match protocol telepresence-media
 ! Identifies TelePresence audio/video traffic via NBAR2


ISR-G2(config-cmap)# class-map match-any NBAR2-MULTIMEDIA-CONFERENCING
ISR-G2(config-cmap)#  match protocol webex-media
 ! Identifies Cisco WebEx audio/video traffic via NBAR2
ISR-G2(config-cmap)#  match protocol webex-meeting
 ! Identifies Cisco WebEx meeting traffic via NBAR2
ISR-G2(config-cmap)#  match protocol webex-app-sharing
 ! Identifies Cisco WebEx application-sharing traffic via NBAR2


ISR-G2(config-cmap)# class-map match-any NBAR2-SIGNALING
ISR-G2(config-cmap)#  match protocol skinny
 ! Identifies Skinny Call Control Protocol (SCCP) via NBAR2
ISR-G2(config-cmap)#  match protocol telepresence-control
 ! Identifies TelePresence control protocols via NBAR2
ISR-G2(config-cmap)#  match protocol sip
 ! Identifies Session Initiation Protocol (SIP) via NBAR2
```

```
ISR-G2(config-cmap)#  match protocol h323
 ! Identifies H.323 protocol via NBAR2


ISR-G2(config-cmap)# class-map match-any NBAR2-TRANSACTIONAL-DATA
ISR-G2(config-cmap)#  match protocol salesforce
 ! Identifies Salesforce.com traffic via NBAR2
ISR-G2(config-cmap)#  match protocol citrix
 ! Identifies Citrix traffic via NBAR2
ISR-G2(config-cmap)#  match protocol sap
 ! Identifies SAP traffic via NBAR2
ISR-G2(config-cmap)#  match protocol oracle-bi
 ! Identifies Oracle traffic via NBAR2


ISR-G2(config-cmap)# class-map match-any NBAR2-BULK-DATA
ISR-G2(config-cmap)#  match protocol ftp
ISR-G2(config-cmap)#  match protocol ftp-data
ISR-G2(config-cmap)#  match protocol ftps-data
  ! Identifies FTP/SFTP traffic via NBAR2
ISR-G2(config-cmap)#  match protocol cifs
 ! Identifies Common Internet File System traffic via NBAR2
ISR-G2(config-cmap)#  match protocol exchange
 ! Identifies Microsoft Exchange traffic via NBAR2
ISR-G2(config-cmap)#  match protocol netapp-snapmirror
 ! Identifies NetApp Backup traffic via NBAR2


ISR-G2(config-cmap)# class-map match-any NBAR2-SCAVENGER
ISR-G2(config-cmap)#  match protocol netflix
 ! Identifies Netflix traffic via NBAR2
ISR-G2(config-cmap)#  match protocol facebook
 ! Identifies Facebook traffic via NBAR2
ISR-G2(config-cmap)#  match protocol hulu
 ! Identifies Hulu traffic via NBAR2
ISR-G2(config-cmap)#  match protocol itunes
 ! Identifies Apple iTunes traffic via NBAR2
ISR-G2(config-cmap)#  match protocol bittorrent
ISR-G2(config-cmap)#  match protocol encrypted-bittorrent
 ! Identifies (regular and encrypted) BitTorrent traffic via NBAR2


 ! This section configures the ingress NBAR2 marking policy map
ISR-G2(config-cmap)# policy-map NBAR2-MARKING
ISR-G2(config-pmap)#  class NBAR2-VOICE
```

```
ISR-G2(config-pmap-c)#   set dscp ef
 ! Marks IPv4/IPv6 DSCP values to EF for voice
ISR-G2(config-pmap-c)#  class NBAR2-REALTIME-INTERACTIVE
ISR-G2(config-pmap-c)#   set dscp cs4
 ! Marks IPv4/IPv6 DSCP values to CS4 for TelePresence
ISR-G2(config-pmap-c)#  class NBAR2-MULTIMEDIA-CONFERENCING
ISR-G2(config-pmap-c)#   set dscp af41
 ! Marks IPv4/IPv6 DSCP values to AF41 for multimedia conferencing
ISR-G2(config-pmap-c)#  class NBAR2-SIGNALING
ISR-G2(config-pmap-c)#   set dscp cs3
 ! Marks IPv4/IPv6 DSCP values to CS3 for signaling
ISR-G2(config-pmap-c)#  class NBAR2-TRANSACTIONAL-DATA
ISR-G2(config-pmap-c)#   set dscp af21
 ! Marks IPv4/IPv6 DSCP values to AF21 for transactional data
ISR-G2(config-pmap-c)#  class NBAR2-BULK-DATA
ISR-G2(config-pmap-c)#   set dscp af11
 ! Marks IPv4/IPv6 DSCP values to AF11 for bulk data
ISR-G2(config-pmap-c)#  class NBAR2-SCAVENGER
ISR-G2(config-pmap-c)#   set dscp cs1
 ! Marks IPv4/IPv6 DSCP values to CS1 for scavenger
ISR-G2(config-pmap-c)#  class class-default
ISR-G2(config-pmap-c)#   set dscp default
 ! Marks IPv4/IPv6 DSCP values to DF/DSCP0 for everything else


 ! This section applies the ingress (application-based)
 ! NBAR2 marking policy to the interface(s)
ISR-G2(config)# interface GigabitEthernet0/1
ISR-G2(config-if)# service-policy input NBAR2-MARKING
```

Note It is necessary to provide unique names for these class maps that are matching these application classes by NBAR2 (as opposed to WAN-edge class maps—presented in the previous chapter and also later in this chapter—that match these application classes by DSCP markings); otherwise one or the other set of class maps will be overwritten at some point. To this end, the NBAR2- prefix is used in these examples.

You can verify the configuration in Example 29-8 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

NBAR2 Application-Group-Based Classification Model

In the attribute-based classification model, the **match protocol** command is combined with the **application-group** keyword, which allows for application subcomponents to be grouped together via a single **match** statement.

Note You can list the application subcomponents that share a given application group by using the **show ip nbar attribute application-group** command.

Example 29-9 shows an example NBAR2 application-group-based classification class map for the Multimedia Conferencing class. In the previous configuration example (Example 29-8), three separate **match** statements were required to match the media, meeting and application-sharing traffic of the Cisco WebEx application for this class. However, these can be simplified to a single **match protocol application-group** statement, as shown here.

Example 29-9 *Cisco ISR G2 NBAR2 (Application-Group-Based) Ingress Classification Model*

```
ISR-G2(config)# class-map match-all NBAR2-MULTIMEDIA-CONFERENCING
ISR-G2(config-cmap)# match protocol application-group webex-group
```

You can verify this configuration with the **show class-map** command.

NBAR2 Attribute-Based Classification Model

In the attribute-based classification model, the **match protocol** command is combined with the **attribute** keyword, which allows for several applications with similar characteristics/attributes to be grouped together via a single **match** statement. As previously mentioned, because NBAR2 supports more than a thousand application signatures, policies can quickly become large and unwieldy if each specific application is explicitly configured with a separate **match** statement.

Note You can list the applications that share a given attribute category or subcategory by using the **show ip nbar attribute category** or **show ip nbar attribute subcategory** commands, respectively.

Example 29-10 shows an extended example NBAR2 application- and attribute-based classification to highlight the many classification options available through IOS Modular QoS command-line interface (MQC) and NBAR2. Some class maps use the **match-all** (logical AND) operator, whereas others use the **match-any** (logical OR) operator, and still others make use of the **match-not** (logical NOT) operator. Attributes are matched both by categories/subcategories, as required. Furthermore, a nested class map is also used to combine logical OR and logical AND classification.

Example 29-10 *Cisco ISR G2 NBAR2 (Application- and Attribute-Based) Ingress Classification and Marking Model*

```
 ! This section configures the NBAR2 class maps
ISR-G2(config-cmap)# class-map match-all NBAR2-VOICE
ISR-G2(config-cmap)#  match protocol cisco-phone
 ! Voice is matched by application-based NBAR2

ISR-G2(config-cmap)# class-map match-all NBAR2-REALTIME-INTERACTIVE
ISR-G2(config-cmap)#  match protocol telepresence-media
 ! TelePresence is matched by application-based NBAR2

ISR-G2(config-cmap)# class-map match-all NBAR2-MULTIMEDIA-CONFERENCING
ISR-G2(config-cmap)#  match protocol attribute category voice-and-video
ISR-G2(config-cmap)#  match protocol attribute sub-category voice-video-chat-collab-
oration
 ! Multimedia-Conferencing is matched by attribute-based NBAR2
 ! This class is matched by the voice-and-video category AND
 ! the voice-video-chat-collaboration subcategory attributes


ISR-G2(config-cmap)# class-map match-all NBAR2-MULTIMEDIA-STREAMING
ISR-G2(config-cmap)#  match protocol attribute category voice-and-video
ISR-G2(config-cmap)#  match protocol attribute sub-category streaming
ISR-G2(config-cmap)#  match not protocol youtube
ISR-G2(config-cmap)#  match not protocol netflix
 ! Multimedia-Streaming is matched by attribute-based NBAR2
 ! This class is matched by the voice-and-video category AND
 ! the streaming subcategory attributes AND
 ! both youtube AND netflix are excluded (via match-not)
 ! from this class (as these are later included to the Scavenger class)


ISR-G2(config-cmap)# class-map match-any NBAR2-TRANSACTIONAL-DATA
ISR-G2(config-cmap)#  match protocol attribute category business-and-productivity-
tools
ISR-G2(config-cmap)# match protocol attribute category instant-messaging
 ! Transactional-Data is matched by attribute-based NBAR2
 ! This class is matched by EITHER the business-and-productivity-tools
 ! OR the instant-messaging category
```

```
ISR-G2(config-cmap)# class-map match-all NBAR2-CLIENT-SERVER-FILE-SHARING
ISR-G2(config-cmap)#  match protocol attribute category file-sharing
ISR-G2(config-cmap)#  match protocol attribute sub-category client-server
 ! This special class map matches by attribute-based NBAR2
 ! This class is matched by the file-sharing category AND
 ! the client-server subcategory attributes


ISR-G2(config-cmap)# class-map match-any NBAR2-BULK-DATA
ISR-G2(config-cmap)#  match protocol attribute category email
ISR-G2(config-cmap)#  match class CLIENT-SERVER-FILE-SHARING
ISR-G2(config-cmap)#  match protocol attribute sub-category backup-systems
 ! Bulk Data is matched by attribute-based NBAR2
 ! This class is matched by EITHER the email category OR
 ! the (nested) class map that matches
 ! (file-sharing AND client-server applications) OR
 ! the backup-systems subcategory attributes

ISR-G2(config-cmap)# class-map match-all NBAR2-SIGNALING
ISR-G2(config-cmap)#  match protocol attribute category voice-and-video
ISR-G2(config-cmap)#  match protocol attribute sub-category control-and-signaling
 ! Signaling is matched by attribute-based NBAR2
 ! This class is matched by the voice-and-video category AND
 ! the control-and-signaling subcategory attributes


ISR-G2(config-cmap)# class-map match-all NBAR2-MANAGEMENT
ISR-G2(config-cmap)#  match protocol attribute category net-admin
ISR-G2(config-cmap)#  match protocol attribute sub-category network-management
 ! Management is matched by attribute-based NBAR2
 ! This class is matched by the net-admin category AND
 ! the network-management subcategory attributes



ISR-G2(config-cmap)# class-map match-any NBAR2-SCAVENGER
ISR-G2(config-cmap)#  match protocol youtube
ISR-G2(config-cmap)#  match protocol netflix
ISR-G2(config-cmap)#  match protocol hulu
ISR-G2(config-cmap)#  match protocol attribute category social-networking
ISR-G2(config-cmap)#  match protocol attribute category gaming
ISR-G2(config-cmap)#  match protocol attribute sub-category p2p-file-transfer
ISR-G2(config-cmap)#  match protocol attribute sub-category p2p-networking
 ! The Scavenger class is matched by both application-
 ! and attribute-based NBAR2
 ! This class matches youtube OR netflix OR hulu OR
 ! the social-networking category OR the gaming category OR
 ! the p2p-file-transfer subcategory OR the p2p-networking
 ! subcategory attributes



 ! This section configures the ingress NBAR2 marking policy map
ISR-G2(config-cmap)# policy-map NBAR2-MARKING
ISR-G2(config-pmap)#   class NBAR2-VOICE
ISR-G2(config-pmap-c)#  set dscp ef
 ! Marks IPv4/IPv6 DSCP values to EF for voice
ISR-G2(config-pmap-c)# class NBAR2-REALTIME-INTERACTIVE
ISR-G2(config-pmap-c)#  set dscp cs4
 ! Marks IPv4/IPv6 DSCP values to CS4 for TelePresence
ISR-G2(config-pmap-c)# class NBAR2-MULTIMEDIA-CONFERENCING
ISR-G2(config-pmap-c)#  set dscp af41
 ! Marks IPv4/IPv6 DSCP values to AF41 for multimedia-conferencing
ISR-G2(config-pmap-c)# class NBAR2-MULTIMEDIA-STREAMING
ISR-G2(config-pmap-c)#  set dscp af31
 ! Marks IPv4/IPv6 DSCP values to AF31 for multimedia-streaming
ISR-G2(config-pmap-c)# class NBAR2-TRANSACTIONAL-DATA
ISR-G2(config-pmap-c)#  set dscp af21
 ! Marks IPv4/IPv6 DSCP values to AF21 for transactional-data
ISR-G2(config-pmap-c)# class NBAR2-BULK-DATA
ISR-G2(config-pmap-c)#  set dscp af11
 ! Marks IPv4/IPv6 DSCP values to AF11 for bulk-data
ISR-G2(config-pmap-c)# class NBAR2-SIGNALING
ISR-G2(config-pmap-c)#  set dscp cs3
 ! Marks IPv4/IPv6 DSCP values to CS3 for signaling
ISR-G2(config-pmap-c)# class NBAR2-MANAGEMENT
ISR-G2(config-pmap-c)#  set dscp cs2
 ! Marks IPv4/IPv6 DSCP values to CS2 for management
ISR-G2(config-pmap-c)# class NBAR2-SCAVENGER
ISR-G2(config-pmap-c)#  set dscp cs1
 ! Marks IPv4/IPv6 DSCP values to CS1 for scavenger
ISR-G2(config-pmap-c)# class class-default
ISR-G2(config-pmap-c)#  set dscp default
 ! Marks IPv4/IPv6 DSCP values to DF/DSCP0 for everything else


 ! This section applies the ingress (application- and
 ! attribute-based) NBAR2 marking policy to the interface(s)
ISR-G2(config)# interface GigabitEthernet0/2
ISR-G2(config-if)# service-policy input NBAR2-MARKING
```

You can verify the configuration in Example 29-10 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Custom-Protocol NBAR2 Classification

In addition to the thousand-plus protocols that NBAR2 already supports, you can define your own custom-tailored protocols to match applications on, as illustrated in the following examples.

In Example 29-11, the custom protocol SALES_APP will identify TCP packets that have a source port of 4567 and that contain the (ASCII) term SALES in the first payload packet.

Example 29-11 *Cisco ISR G2 NBAR2 Custom Protocol Example*

```
ISR-G2(config)# ip nbar custom SALES_APP 5 ascii SALES source tcp 4567
ISR-G2(config)# class-map match-all MATCH-SALES_APP
ISR-G2(config-cmap)# match protocol SALES_APP
```

You can verify the configuration in Example 29-11 with the following commands:

- **show ip nbar port-map** (as shown in Example 29-12)
- **show class-map**

Example 29-12 *Verifying Custom NBAR2 Protocols:* ***show ip nbar port-map***

```
ISR-G2# show ip nbar port-map SALES_APP
port-map SALES_APP                tcp 4567
ISR-G2#
```

In Example 29-13, the custom protocol NEW_VIRUS will identify UDP packets that have a destination port of 3000 and that contain (the hexadecimal value) 0x56 in the seventh byte of the first packet of the flow.

Example 29-13 *Cisco ISR G2 NBAR2 Custom Protocol Example*

```
ISR-G2(config)# ip nbar custom NEW_VIRUS 7 hex 56 destination udp 3000
ISR-G2(config)# class-map match-all MATCH-NEW_VIRUS
ISR-G2(config-cmap)# match protocol NEW_VIRUS
```

You can verify the configuration in Example 29-12 with the following commands:

- **show ip nbar port-map**
- **show class-map**

In Example 29-14, the custom protocol NEW_MEDIA will identify TCP packets that have a destination or source port range of 4500 through 4510 (inclusive) and that have a (decimal) value of 90 at the sixth byte of the payload. Only the first packet of the flow is checked for the value 90 at the offset 6.

Example 29-14 *Cisco ISR G2 NBAR2 Custom Protocol Example*

```
ISR-G2(config-cmap)# ip nbar custom NEW_MEDIA 6 decimal 90 tcp range 4500 4510
ISR-G2(config)# class-map match-all MATCH-NEW_MEDIA
ISR-G2(config-cmap)# match protocol NEW_MEDIA
```

You can verify the configuration in Example 29-14 with the following commands:

- **show ip nbar port-map**
- **show class-map**

Egress QoS Models

Typically, the WAN-edge egress QoS models will match those configured on the WAN aggregator WAN edges (as detailed in Chapter 28, “WAN Aggregator [Cisco ASR 1000] QoS Design”). However, minor exceptions/deviations from these policies may exist.

For example, some application classes are unidirectional, and therefore usually flow from the campus to the branch (such as the Broadcast Video or Multimedia Streaming application classes). These application classes may optionally be omitted from corresponding branch WAN-edge design. However, it is good to keep in mind that any unused bandwidth for a given class will automatically be redistributed by the IOS Software to other classes that may require it—and so there is not much advantage of removing such classes from the branch WAN edges. In addition, for the sake of policy simplicity, consistency, and troubleshooting, you may prefer to keep the same policies on the branch WAN edge as at the WAN aggregator WAN edge.

Another minor deviation may exist if weighted random early detection (WRED) is tuned. In such cases, the default queue depths from WAN aggregator platforms (such as the Cisco ASR 1000) will likely differ from the default queue depths of the branch router platforms (such as the Cisco ISR G2s). In such cases, the policy will vary slightly to reflect these differing queue depths and WRED thresholds.

Otherwise, it can generally be said that the branch WAN-edge egress QoS models will be nearly identical to the WAN aggregator WAN-edge egress QoS models.

Four-Class Model

The branch router four-class WAN-edge egress QoS model is identical to the WAN aggregator four-class WAN-edge egress QoS model, as detailed in Example 29-5.

Eight-Class Model

The branch router eight-class WAN-edge egress QoS model is identical to the WAN aggregator eight-Class WAN-edge egress QoS model, as detailed in Example 29-8.

Twelve-Class Model

The branch router 12-class WAN-edge egress QoS model is nearly identical to the WAN aggregator 12-class WAN-edge egress QoS model (as detailed in Example 29-9), with the exception that the WRED thresholds have been tuned based on the default Cisco ISR G2 queue depth of 64 packets, and shown in Example 29-15.

Note In Example 29-15, all application classes (including classes that may be unidirectional from the campus to branch) have been retained for policy consistency with the WAN aggregator WAN edge.

Note To minimize redundancy, the class maps are not repeated (but are detailed in Example 29-9).

Note DSCP-based WRED tuning has been done such that AFx3 packets begin randomly dropping at (approximately) 60 percent of the default queue depth (of 64 packets), AFx2 packets at 70 percent, and AFx1 packets at 80 percent. In the default queue, packets are all marked DF/DSCP 0 and begin to be dropped only at 80 percent.

Example 29-15 *Cisco ISR G2 12-Class Egress Queuing Model Example*

```
 ! This section defines the Twelve-Class egress queuing policy
 ! with tuned WRED thresholds (for the ISR G2)
ISR-G2(config-cmap)# policy-map ISR-TWELVE-CLASS-WAN-EDGE
ISR-G2(config-pmap)# class VOICE
ISR-G2(config-pmap-c)#  priority percent 10
 ! Provisions the VOICE class with 10% LLQ
ISR-G2(config-pmap-c)# class BROADCAST-VIDEO
ISR-G2(config-pmap-c)#  priority percent 10
```

```
 ! Provisions the BROADCAST-VIDEO class with 10% LLQ
ISR-G2(config-pmap-c)# class REALTIME-INTERACTIVE
ISR-G2(config-pmap-c)#  priority percent 13
 ! Provisions the REALTIME-INTERACTIVE class with 13% LLQ
ISR-G2(config-pmap-c)# class NETWORK-CONTROL
ISR-G2(config-pmap-c)#  bandwidth percent 2
 ! Provisions the NETWORK-CONTROL class with 2% CBWFQ
ISR-G2(config-pmap-c)# class SIGNALING
ISR-G2(config-pmap-c)#  bandwidth percent 2
 ! Provisions the SIGNALING class with 2% CBWFQ
ISR-G2(config-pmap-c)# class NETWORK-MANAGEMENT
ISR-G2(config-pmap-c)#  bandwidth percent 3
 ! Provisions the NETWORK-MANAGEMENT class with 3% CBWFQ
ISR-G2(config-pmap-c)# class MULTIMEDIA-CONFERENCING
ISR-G2(config-pmap-c)#  bandwidth percent 10
 ! Provisions the MULTIMEDIA-CONFERENCING class with 10% CBWFQ
ISR-G2(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ISR-G2(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ISR-G2(config-pmap-c)#  random-detect dscp af43 40 64
 ! Tunes minimum DSCP-WRED threshold for AF43 to 60% queue-depth
ISR-G2(config-pmap-c)#  random-detect dscp af42 45 64
 ! Tunes minimum DSCP-WRED threshold for AF42 to 70% queue-depth
ISR-G2(config-pmap-c)#  random-detect dscp af41 50 64
 ! Tunes minimum DSCP-WRED threshold for AF41 to 80% queue-depth
ISR-G2(config-pmap-c)# class MULTIMEDIA-STREAMING
ISR-G2(config-pmap-c)#  bandwidth percent 10
 ! Provisions the MULTIMEDIA-STREAMING class with 10% CBWFQ
ISR-G2(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ISR-G2(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ISR-G2(config-pmap-c)#  random-detect dscp af33 40 64
 ! Tunes minimum DSCP-WRED threshold for AF33 to 60% queue-depth
ISR-G2(config-pmap-c)#  random-detect dscp af32 45 64
 ! Tunes minimum DSCP-WRED threshold for AF32 to 70% queue-depth
ISR-G2(config-pmap-c)#  random-detect dscp af31 50 64
 ! Tunes minimum DSCP-WRED threshold for AF31 to 80% queue-depth
ISR-G2(config-pmap-c)# class TRANSACTIONAL-DATA
ISR-G2(config-pmap-c)#  bandwidth percent 10
 ! Provisions the TRANSACTIONAL-DATA class with 10% CBWFQ
ISR-G2(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ISR-G2(config-pmap-c)#  random-detect dscp-based
```

```
 ! Enables DSCP-based WRED
ISR-G2(config-pmap-c)#  random-detect dscp af23 40 64
 ! Tunes minimum DSCP-WRED threshold for AF23 to 60% queue-depth
ISR-G2(config-pmap-c)#  random-detect dscp af22 45 64
 ! Tunes minimum DSCP-WRED threshold for AF22 to 70% queue-depth
ISR-G2(config-pmap-c)#  random-detect dscp af21 50 64
 ! Tunes minimum DSCP-WRED threshold for AF21 to 80% queue-depth
ISR-G2(config-pmap-c)# class BULK-DATA
ISR-G2(config-pmap-c)#  bandwidth percent 4
 ! Provisions the BULK-DATA class with 4% CBWFQ
ISR-G2(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ISR-G2(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ISR-G2(config-pmap-c)#  random-detect dscp af13 40 64
 ! Tunes minimum DSCP-WRED threshold for AF13 to 60% queue-depth
ISR-G2(config-pmap-c)#  random-detect dscp af12 45 64
 ! Tunes minimum DSCP-WRED threshold for AF12 to 70% queue-depth
ISR-G2(config-pmap-c)#  random-detect dscp af11 50 64
 ! Tunes minimum DSCP-WRED threshold for AF11 to 80% queue-depth
ISR-G2(config-pmap-c)# class SCAVENGER
ISR-G2(config-pmap-c)#  bandwidth percent 1
 ! Constrains the SCAVENGER class to 1% CBWFQ
ISR-G2(config-pmap-c)# class class-default
ISR-G2(config-pmap-c)# bandwidth percent 25
 ! Provisions the default/Best-Effort class with 25% CBWFQ
ISR-G2(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ISR-G2(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ISR-G2(config-pmap-c)#  random-detect dscp default 50 64
 ! Tunes minimum DSCP-WRED threshold for DF to 80% queue-depth



 ! This section applies the queuing policy to a DS3 Serial interface
ISR-G2(config)#interface Serial1/0
ISR-G2(config-if)# service-policy output ISR-TWELVE-CLASS-WAN-EDGE
 ! Attaches egress queuing service policy to the interface
```

You can verify the configuration in Example 29-15 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Additional Platform-Specific QoS Design Options

These designs represent the basic elements of branch router QoS design, but they are by no means the only design options available to you. Additional options and considerations include the following:

- RSVP
- AutoQoS
- Control plane policing

Each of these additional QoS design options and considerations is briefly discussed in turn.

RSVP

Resource Reservation Protocol (RSVP) designs for the branch router will be identical to those on WAN aggregation routers. Therefore, refer to the previous chapter for details on these RSVP design options.

AutoQoS SRND4

As noted in Chapter 28, AutoQoS SRND4 is not supported in Cisco IOS for ISR G2 routers (at the time of this writing); however, AutoQoS-VoIP and AutoQoS-Enterprise are supported on this platform.

Control Plane Policing

Control plane policing (CPP) is supported in Cisco IOS, and design recommendations for this feature are detailed in Appendix B, "Control Plane Policing."

Summary

This chapter outlined the QoS roles of a branch router and also identified the Cisco ISR G2 family of routers as being well suited to this role.

Ingress QoS designs were detailed, focusing on the DPI capabilities of Cisco IOS, including both Medianet metadata and NBAR2 classification. In addition, it was shown how these classifiers can operate on an application basis, an application-group basis, and on an attribute basis. Also included in this discussion was how custom protocols could be mapped into the NBAR2 classification engine for enterprise-specific application classification.

Next, egress QoS design were overviewed in brief. Most of these are identical to the 4-, 8-, and 12-class queuing models detailed in the previous chapter. However, an example 12-class queuing model—specifically tuned to the ISR G2—was presented.

Additional QoS design options were briefly discussed, including RSVP, AutoQoS, and CPP.

Further Reading

Medianet WAN Aggregation QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html

Cisco IOS Release 15M&T—QoS Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos/config_library/15-mt/qos-15-mt-library.html

Cisco IOS Release 15M&T—Medianet Metadata Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/mdata/configuration/15-mt/metadata-framework.html

Cisco IOS Release 15M&T—Medianet NBAR Integration Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/mdata/configuration/15-mt/mdata-nbar-intgrtn.html

Cisco IOS Release 15M&T—NBAR Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_nbar/configuration/15-mt/qos-nbar-15-mt-book.html

Cisco NBAR2 Protocol Library: http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6558/ps6616/product_bulletin_c25-627831.html

Chapter 30

WAN and Branch QoS Design Case Study

With their (wired and wireless) campus networks in place, Tifosi's networking team is ready to extend their QoS deployment over the wide area. Their first phase of WAN QoS deployment will focus on their private WAN infrastructure.

Tifosi uses Cisco ASR 1000 series routers (each with ESP20s and SIP10s) for their WAN aggregation routers and either Cisco 3925E ISR G2s or Cisco 3945E ISR G2s for their branch routers (depending on the size of the branch).

Tifosi uses a combination of link types for WAN connectivity, including direct Serial and Packet over Sonet (POS) links, and ATM links. Link speeds range from T3/DS3 (45 Mbps) through OC3 (155 Mbps).

Tifosi's networking team has two main objectives for their WAN and branch QoS designs:

1. To deploy their strategic eight-class QoS model (detailed in Chapter 12, "Strategic QoS Design Case Study")
2. To increase the granularity of application classification by leveraging the deep packet inspection capabilities available in Cisco IOS for ASR and ISR routers

Figure 30-1 shows the QoS policies to be applied to Tifosi's (private) WAN and branch network.

Tifosi wants to simplify and modularize their QoS policies as much as possible. To this end, they will provision bandwidth-use percentages (so that a single queuing policy can be leveraged for all link speeds). In addition, they have decided to use identical LAN- and WAN-edge policies on both their WAN aggregators and their branch routers.

Thus, Tifosi's WAN and branch QoS policies are detailed as follows:

- **Policy 1: Internal (PLIM) QoS for ASR 1000**
  - Enable SIP-based PLIM
  - Enable SPA-based PLIM

- **Policy 2: LAN-Edge QoS Policies**
  - Trust DSCP on ingress (Cisco IOS default)
  - Perform NBAR2 protocol classification on ingress, per Table 30-1
  - Provision 8-class queuing model on egress (required only on Tifosi's WAN aggregation routers, due to potential WAN-to-LAN oversubscription, as discussed later)

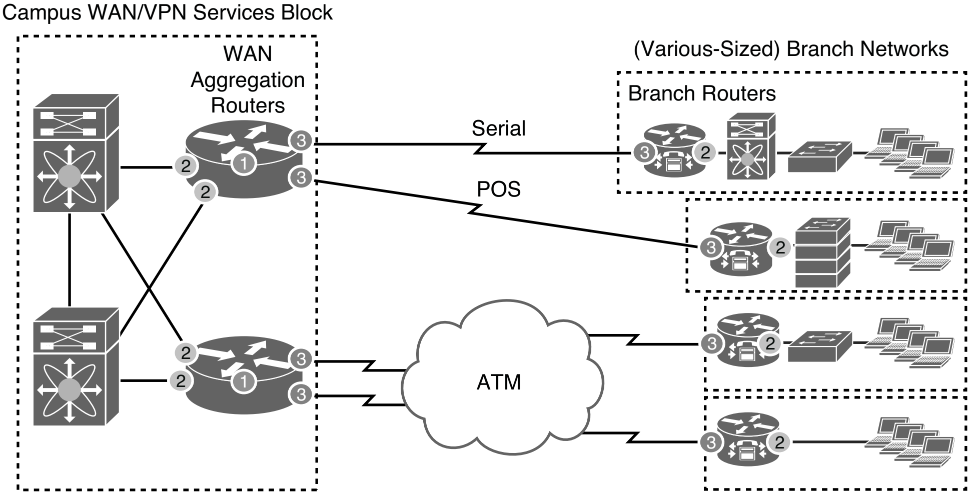

Campus WAN/VPN Services Block
WAN Aggregation Routers
(Various-Sized) Branch Networks
Branch Routers
Serial
POS
ATM
1
2
3

Figure 30-1 *Tifosi Software Case Study: WAN and Branch Network QoS Design*

Table 30-1 *Tifosi 8-Class Deep-Packet Inspection Classification and Marking Policy Table*

Application Class	Deep Packet Inspection Classification Criteria	Marking
Voice	**match protocol cisco-phone**	EF
Real-Time Interactive	**match protocol telepresence-media**	CS4
Signaling	**match protocol attribute category voice-and-video** AND **match protocol attribute sub-category control-and-signaling**	CS3
Multimedia Conferencing	**match protocol attribute category voice-and-video** AND **match protocol attribute sub-category voice-video-chat-collaboration**	AF4

Application Class	Deep Packet Inspection Classification Criteria	Marking
Transactional Data	**match protocol SALES_APP** (*custom protocol*) OR **match protocol attribute category business-and-productivity-tools** OR **match protocol attribute category instant-messaging**	AF2
Bulk Data	**match protocol attribute category email** OR **match protocol attribute sub-category backup-systems** OR **(match protocol attribute category file-sharing** AND **match protocol attribute sub-category client-server)**	AF1
Scavenger	**match protocol youtube** OR **match protocol netflix** OR **match protocol hulu** OR **match protocol attribute category social-networking** OR **match protocol attribute category gaming** OR **match protocol attribute sub-category p2p-file-transfer** OR **match protocol attribute sub-category p2p-networking**	CS1
Best Effort	Default	DF

- **Policy 3: WAN-Edge QoS Policies**
  - Trust DSCP (Cisco IOS default)
  - Provision eight-class queuing model on egress

Policy 1: Internal (PLIM) QoS for ASR 1000

Tifosi's WAN aggregation routers have the potential for internal congestion due to the hardware combination of ESP20s with SIP10Gs (as pointed out in Table 28-1). Therefore, these routers will require Physical Layer Interface Module (PLIM) QoS enabled to protect real-time voice and TelePresence traffic from internal oversubscription.

The Serial and POS interfaces will require SIP-based PLIM, and the GE and ATM interfaces will require SPA-based PLIM. Each option will be detailed in turn.

Policy 1a: SIP-Based PLIM QoS

The Serial and POS interfaces will require SIP-based PLIM, as detailed in Example 30-1.

Example 30-1 *Tifosi WAN Aggregator Design: Cisco ASR 1000 SIP-Based Internal Scheduling Classification Example*

```
 ! This section defines the ingress scheduling classification template
ASR(config)# ingress-class-map 1
ASR(config-ing-class-map)# map ip dscp-based
ASR(config-ing-class-map)# map ip dscp 32 queue strict-priority
 ! Maps DSCP 32/CS4 (Realtime Interactive) to the internal PQ
 ! (EF is already mapped to the internal PQ by default)
...

ASR(config)# interface Serial0/1/0
ASR(config-if)# plim qos input class-map 1
 ! Attaches the SIP-based PLIM ingress scheduling template
 ! to the Serial interface
...

ASR(config)# interface POS0/2/0
ASR(config-if)# plim qos input class-map 1
 ! Attaches the SIP-based PLIM ingress scheduling template
 ! to the POS interface
```

Policy 1b: SPA-Based PLIM QoS

The GE and ATM interfaces will require SPA-based PLIM, as detailed in Example 30-2.

Example 30-2 *Tifosi WAN Aggregator Design: Cisco ASR 1000 SPA-Based Internal Scheduling Classification Example*

```
ASR(config)# interface GigabitEthernet0/0/0
ASR(config-if)# plim qos input map ip dscp-based
ASR(config-if)# plim qos input map ip dscp cs4 queue strict-priority
 ! Maps CS4 (Realtime Interactive) to the internal PQ for the GE int
 ! (EF is already mapped to the internal PQ by default)
```

```
...

ASR(config)# interface ATM4/1/0
ASR(config-if)# plim qos input map ip dscp-based
ASR(config-if)# plim qos input map ip dscp  cs4 queue strict-priority
 ! Maps CS4 (Realtime Interactive) & to the internal PQ for the ATM int
 ! (EF is already mapped to the internal PQ by default)
```

Policy 2: LAN-Edge QoS Policies

The WAN and branch LAN-edge QoS policies will trust DSCP on ingress (which is a Cisco IOS default), and perform detailed deep packet inspection via NBAR2. This inspection which will include the identification of a custom (Transactional Data) protocol called SALES_APP.

Example 30-3 shows the (ingress) QoS policy to be applied to Tifosi's WAN aggregator and branch router LAN edges.

Example 30-3 *Tifosi Case-Study: LAN-Edge (Ingress) QoS Policy*

```
 ! This section configures the custom NBAR2 protocol SALES_APP
ASR/ISR(config)# ip nbar custom SALES_APP 5 ascii SALES source tcp 4567


 ! This section configures the NBAR2 class-maps
ASR/ISR(config-cmap)# class-map match-all NBAR2-VOICE
ASR/ISR(config-cmap)#  match protocol cisco-phone
 ! Voice is matched by the cisco-phone NBAR2 protocol


ASR/ISR(config-cmap)# class-map match-all NBAR2-REALTIME-INTERACTIVE
ASR/ISR(config-cmap)#  match protocol telepresence-media
 ! TelePresence is matched by the telepresence-media NBAR2 protocol


ASR/ISR(config-cmap)# class-map match-all NBAR2-SIGNALING
ASR/ISR(config-cmap)#  match protocol attribute category voice-and-video
ASR/ISR(config-cmap)#  match protocol attribute sub-category control-and-signaling
 ! Signaling is matched the voice-and-video category AND
 ! the control-and-signaling sub-category NBAR2 attributes


ASR/ISR(config-cmap)# class-map match-all NBAR2-MULTIMEDIA-CONFERENCING
ASR/ISR(config-cmap)#  match protocol attribute category voice-and-video
```

```
ASR/ISR(config-cmap)#  match protocol attribute sub-category voice-video-chat-collaboration
 ! Multimedia-Conferencing is matched by the voice-and-video category
 ! AND the voice-video-chat-collaboration sub-category NBAR2 attributes



ASR/ISR(config-cmap)# class-map match-any NBAR2-TRANSACTIONAL-DATA
ASR/ISR(config-cmap)#  match protocol SALES_APP
ASR/ISR(config-cmap)#  match protocol attribute category business-and-productivity-tools
ASR/ISR(config-cmap)# match protocol attribute category instant-messaging
 ! Transactional-Data is matched by the custom SALES_APP OR
 ! the business-and-productivity-tools category OR
 ! the instant-messaging category NBAR2 attributes


ASR/ISR(config-cmap)# class-map match-all NBAR2-CLIENT-SERVER-FILE-SHARING
ASR/ISR(config-cmap)#  match protocol attribute category file-sharing
ASR/ISR(config-cmap)#  match protocol attribute sub-category client-server
 ! This special class-map matches the file-sharing category AND
 ! the client-server sub-category NBAR2 attributes


ASR/ISR(config-cmap)# class-map match-any NBAR2-BULK-DATA
ASR/ISR(config-cmap)#  match protocol attribute category email
ASR/ISR(config-cmap)#  match protocol attribute sub-category backup-systems
ASR/ISR(config-cmap)#  match class CLIENT-SERVER-FILE-SHARING
 ! Bulk Data is matched by the email category OR
 ! the backup-systems sub-category NBAR2 attributes OR
 ! the (nested) class-map that matches the
 ! (file-sharing AND client-server NBAR2 attributes)


ASR/ISR(config-cmap)# class-map match-any NBAR2-SCAVENGER
ASR/ISR(config-cmap)#  match protocol youtube
ASR/ISR(config-cmap)#  match protocol netflix
ASR/ISR(config-cmap)#  match protocol hulu
ASR/ISR(config-cmap)#  match protocol attribute category social-networking
ASR/ISR(config-cmap)#  match protocol attribute category gaming
ASR/ISR(config-cmap)#  match protocol attribute sub-category p2p-file-transfer
ASR/ISR(config-cmap)#  match protocol attribute sub-category p2p-networking
 ! The Scavenger class is matched by youtube OR netflix OR hulu
 ! NBAR2 protocols OR the social-networking category (which includes
 ! facebook) OR the gaming category OR the p2p-file-transfer
 ! sub-category OR the p2p-networking sub-category NBAR2 attributes


 ! This section configures the ingress NBAR2 marking policy map
ASR/ISR(config-cmap)# policy-map LAN-EDGE
ASR/ISR(config-pmap)#   class NBAR2-VOICE
ASR/ISR(config-pmap-c)#  set dscp ef
ASR/ISR(config-pmap-c)# class NBAR2-REALTIME-INTERACTIVE
ASR/ISR(config-pmap-c)#  set dscp cs4
ASR/ISR(config-pmap-c)# class NBAR2-SIGNALING
ASR/ISR(config-pmap-c)#  set dscp cs3
ASR/ISR(config-pmap-c)# class NBAR2-MULTIMEDIA-CONFERENCING
ASR/ISR(config-pmap-c)#  set dscp af41
ASR/ISR(config-pmap-c)# class NBAR2-TRANSACTIONAL-DATA
ASR/ISR(config-pmap-c)#  set dscp af21
ASR/ISR(config-pmap-c)# class NBAR2-BULK-DATA
ASR/ISR(config-pmap-c)#  set dscp af11
ASR/ISR(config-pmap-c)# class NBAR2-SCAVENGER
ASR/ISR(config-pmap-c)#  set dscp cs1
ASR/ISR(config-pmap-c)# class class-default
ASR/ISR(config-pmap-c)#  set dscp default


 ! This section applies the LAN edge policy to an ASR GE int
ASR(config)# interface GigabitEthernet0/0/0
ASR(config-if)# service-policy input LAN-EDGE
```

Note The LAN-edge policy would correspondingly be applied to the GE LAN interfaces of the ISR G2 branch routers.

Note Again, it is necessary to provide unique names for these class maps, which are matching these application classes by NBAR2 (as opposed to WAN-edge class maps—presented later in this chapter—that match these application classes by DSCP markings); otherwise, one or the other set of class maps will be overwritten at some point. To this end, the NBAR2- prefix is used in these examples.

Policy 3: WAN Edge QoS Policies

Figure 30-2 illustrates Tifosi's eight-class WAN-edge queuing model (detailed in Chapter 12, "Strategic QoS Design Case Study").

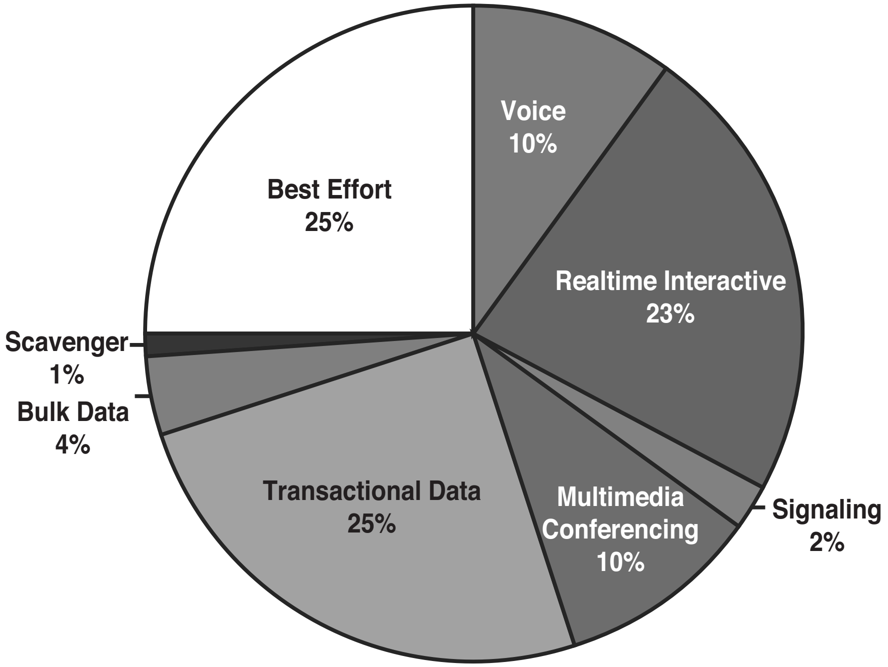


Figure 30-2 *Tifosi Software Case Study: Eight-Class Queuing Model*

Example 30-4 shows the corresponding WAN queuing policy.

Example 30-4 *Tifosi Case-Study: WAN-Edge QoS Policy*

```
 ! This section configures the DSCP-matching class maps
 ! for the Tifosi eight-class egress queuing policy
ASR/ISR(config-cmap)# class-map match-all VOICE
ASR/ISR(config-cmap)#  match dscp ef
ASR/ISR(config-cmap)# class-map match-all REALTIME-INTERACTIVE
ASR/ISR(config-cmap)#  match dscp cs4
ASR/ISR(config-cmap)# class-map match-all SIGNALING
ASR/ISR(config-cmap)#  match dscp cs3
ASR/ISR(config-cmap)# class-map match-all MULTIMEDIA-CONFERENCING
ASR/ISR(config-cmap)#  match dscp af41
ASR/ISR(config-cmap)# class-map match-all TRANSACTIONAL-DATA
ASR/ISR(config-cmap)#  match dscp af21
ASR/ISR(config-cmap)# class-map match-all BULK-DATA
ASR/ISR(config-cmap)#  match dscp af11
ASR/ISR(config-cmap)# class-map match-all SCAVENGER
ASR/ISR(config-cmap)#  match dscp cs1
```

```
 ! This section defines the eight-class WAN-edge egress queuing policy
ASR/ISR(config-cmap)# policy-map WAN-EDGE
ASR/ISR(config-pmap)#  class VOICE
ASR/ISR(config-pmap-c)# priority percent 10
ASR/ISR(config-pmap-c)# class REALTIME-INTERACTIVE
ASR/ISR(config-pmap-c)#  priority percent 23
ASR/ISR(config-pmap-c)# class SIGNALING
ASR/ISR(config-pmap-c)#  bandwidth percent 2
ASR/ISR(config-pmap-c)# class MULTIMEDIA-CONFERENCING
ASR/ISR(config-pmap-c)#  bandwidth percent 10
ASR/ISR(config-pmap-c)#  fair-queue
ASR/ISR(config-pmap-c)#  random-detect dscp-based
ASR/ISR(config-pmap-c)# class TRANSACTIONAL-DATA
ASR/ISR(config-pmap-c)#  bandwidth percent 25
ASR/ISR(config-pmap-c)#  fair-queue
ASR/ISR(config-pmap-c)#  random-detect dscp-based
ASR/ISR(config-pmap-c)# class BULK-DATA
ASR/ISR(config-pmap-c)#  bandwidth percent 4
ASR/ISR(config-pmap-c)#  fair-queue
ASR/ISR(config-pmap-c)#  random-detect dscp-based
ASR/ISR(config-pmap-c)# class SCAVENGER
ASR/ISR(config-pmap-c)#  bandwidth percent 1
ASR/ISR(config-pmap-c)# class class-default
ASR/ISR(config-pmap-c)#  bandwidth percent 25
ASR/ISR(config-pmap-c)#  fair-queue
ASR/ISR(config-pmap-c)#  random-detect


 ! This section tunes the Serial interface Tx-Ring and
 ! attaches the 8-class WAN-edge queuing policy
ASR(config)# interface Serial0/1/0
ASR(config-if)# tx-ring-limit 10
 ! Optimizes the T3 Tx-Ring for rich media applications
ASR(config-if)# service-policy output WAN-EDGE
 ! Attaches the WAN-edge queuing policy to the Serial interface


 ! This section attaches the 8-Class WAN-edge queuing policy
 ! to a POS interface
ASR(config)# interface POS0/2/0
ASR(config-if)# service-policy output WAN-EDGE
 ! Attaches the WAN-edge queuing policy to the POS interface
 ! This section attaches the 8-Class WAN-edge queuing policy
 ! to an ATM OC3 PVC
ASR(config)# interface ATM4/1/0.1 point-to-point
ASR(config-if-atm)# pvc 0/112
ASR(config-if-atm-pvc)# vbr-rt 149760 149760
 ! Defines the ATM traffic contract:
 ! Variable Bit Rate - Realtime (OC-3)
ASR(config-if-atm-pvc)# service-policy output WAN-EDGE
 ! Attaches the WAN-edge queuing policy to the (OC3) ATM PVC
```

Note The WAN-edge policy would correspondingly be applied to the WAN interfaces of the ISR G2 branch routers.

Tifosi's network administrators have added up the aggregate bandwidth of all the individual WAN links on the WAN aggregation routers and found these to be in excess of 1 Gbps (which is the speed of the LAN-edge links). Therefore, to protect traffic in the WAN-to-LAN direction (in the case of oversubscription), they have decided to apply the same eight-class WAN-edge policy to the GE LAN-edge link, as shown in Example 30-5.

Example 30-5 *Tifosi Case-Study: WAN-Edge QoS Policy Applied to LAN-Edge Link (WAN Aggregator Only)*

```
 ! This section applies the 8-class WAN-edge queuing policy
 ! to the LAN-Edge GE interface (WAN Aggregator only)
ASR(config)#interface Gigabit 0/0/0
ASR(config-if)# service-policy output WAN-EDGE
 ! Attaches the WAN-edge queuing policy to the LAN-edge
 ! GE interface (on the WAN aggregator)
```

Note Because no oversubscription exists in the WAN-to-LAN direction on any of their branch networks, the eight-class WAN-edge policy is not required to be applied to the LAN-edge links on the branch routers.

Summary

This chapter continued the case study example of Tifosi Software and applied their strategic eight-class end-to-end QoS model to their WAN aggregation and branch routers for their private WAN network.

LAN- and WAN-edge policies were simplified and modularized so that the same policies could be used on both the WAN aggregation Cisco ASR 1000 routers and the Cisco ISR G2 branch routers for all link speeds.

The LAN-edge policies utilized NBAR2 application recognition to provide deep packet inspection of traffic-classes (which is beyond the ability of campus access switches to do). In addition, these classification policies were simplified by leveraging NBAR2 category and subcategory attributes, allowing for a large number of application protocols to be utilized without having an extensive and complex policy configuration.

The WAN-edge policies utilized Cisco IOS LLQ/CBWFQ and WRED to optimize their eight-class strategic QoS model over the WAN. The same eight-class WAN-edge model was also applied to the LAN edge of the WAN aggregation routers to manage the potential for oversubscription in the WAN-to-LAN direction.

Further Reading

Medianet WAN Aggregation QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html

ASR 1000 QoS Architecture: http://www.cisco.com/en/US/prod/collateral/routers/ps9343/solution_overview_c22-449961_ps9343_Product_Solution_Overview.html

Cisco IOS Quality of Service Solutions Configuration Guide Library: Cisco IOS Release 15M&T http://www.cisco.com/en/US/docs/ios-xml/ios/qos/config_library/15-mt/qos-15-mt-library.html

Cisco IOS Release 15M&T—NBAR Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_nbar/configuration/15-mt/qos-nbar-15-mt-book.html

Cisco NBAR2 Protocol Library: http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6558/ps6616/product_bulletin_c25-627831.html

This page intentionally left blank

Chapter 31

MPLS VPN QoS Design Considerations and Recommendations

In addition to the dual role of quality of service (QoS) in the private WAN and branch networks (namely that of managing packet loss and jitter by queuing policies and of enhancing classification granularity by leveraging deep packet inspection engines), the role of QoS over the Multiprotocol Label Switching (MPLS) virtual private network (VPN) may be expanded to include the following:

- Shaping traffic to contracted service rates
- Performing hierarchical queuing and dropping within these shaped rates
- Mapping enterprise-to-service provider class of service markings
- Policing traffic classes according to contracted rates
- Restoring packet markings

Therefore, several of the strategic QoS design principles (discussed in Chapter 11, "QoS Design Principles and Strategies") apply to MPLS VPN QoS designs, including the following:

- **Classify and mark applications as close to their sources as technically and administratively feasible:** Some classification policies may require Layer 7 awareness and, therefore, may not be possible to perform on campus and branch Catalyst switches. Therefore, the ingress edge of the customer-edge (CE) router may be the closest technically feasible point to perform such detailed classification.
- **Police unwanted traffic flows as close to their sources as possible:** Service providers will police traffic at their provider-edge (PE) ingress edges to meter flows according to contracted rates. Traffic in excess of these rates may be subject to re-marking, dropping, or additional charges to the subscriber.
- **Enable queuing policies at every node where the potential for congestion exists:** Enable queuing policies on all CE and PE edges; also enable queuing policies in the provider core (if potentially oversubscribed).

- **(Optional) Protect the control plane and data plane by enabling control plane policing:** To harden the network infrastructure to mitigate and constrain network attacks.

However, before these strategic QoS design principles can be translated into platform-specific configuration recommendations, you need to account for a few additional MPLS VPN-specific considerations, including the following:

- MPLS VPN architectures
- Carrier Ethernet services
- Sub-line rate Ethernet design implications
- QoS paradigm shift
- Service provider class of service models
- MPLS DiffServ tunneling modes
- Enterprise-to-service provider mapping
- MPLS VPN interface QoS roles

Each of these MPLS VPN specific considerations is discussed in turn.

MPLS VPN Architectures

MPLS VPN architectures include the following:

- CE routers
- PE routers
- Provider (core) routers

Figure 31-1 shows these routers in their respective roles within the MPLS VPN architecture.

MPLS VPNs provide fully meshed Layer 3 virtual WAN services to all interconnected CE routers, as defined in RFC 2547, *BGP MPLS VPNs*. It is this full mesh that provides additional QoS complexity, as discussed shortly. But first, let's consider the access links to the MPLS VPN.

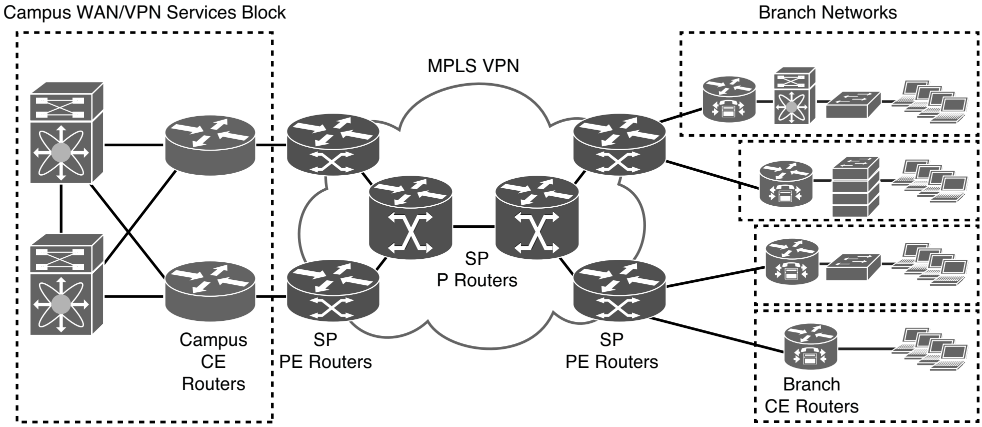


Figure 31-1 *MPLS VPN Architectures and Router Roles*

MAN and WAN Ethernet Service Evolution

Recommendation:

- Realize that Ethernet—already the de facto protocol in the LAN—is rapidly dominating the MAN/WAN as well

Due to the popularity—and indeed the ultimate dominance—of Ethernet as the de facto LAN protocol, customers began demanding their service providers (SPs) to extend Ethernet capabilities to seamlessly connect their multisite LANs—at first, via metropolitan-area networks (MANs) and then over wide-area networks (WANs).

To this end, the Metro Ethernet Forum (MEF) was formed in 2001 to develop ubiquitous business services for enterprise users who connected their LANs over optical metropolitan networks. The principal concept was to bring the simplicity and cost model of Ethernet to the WAN.

To create a market in wide-area Ethernet services, it was first necessary to clarify and standardize the services to be provided. Recognizing this, the MEF defined the following key Ethernet connectivity services:

- **E-Line:** A service connecting two customer Ethernet ports over a WAN. E-Line is based on a point-to-point Ethernet Virtual Connection (EVC). Two E-Line services are defined:

- **Ethernet Private Line (EPL):** A simple and basic point-to-point service characterized by low frame delay, frame delay variation, and frame loss ratio. No service multiplexing is allowed, and—other than a Committed Information Rate (CIR)—no class of service (CoS) bandwidth profiling is allowed.

  - **Ethernet Virtual Private Line (EVPL):** A point-to-point service wherein service multiplexing (of more than one EVC) is allowed. The individual EVCs can be defined with a rich set of bandwidth profiles and Layer 2 control protocol processing methods.

- **E-LAN:** A multipoint service connecting a set of customer endpoints, giving the appearance to the customer of a bridged Ethernet network connecting the sites. E-LAN is based on a multipoint-to-multipoint EVC. Service multiplexing (of more than one EVC) is permitted, as is the rich set of performance assurances, such as CIR with an associated committed burst size (CBS) and excess information rate (EIR).

- **E-Tree:** A point-to-multipoint ELAN service in which the spoke "leaves" can communicate with the hub or "root" location but not with each other. Typical application for E-Tree is in franchise operations.

As MPLS VPNs evolved, so did their capacity to extend the reach of Ethernet. For example, RFC 3985 defined the pseudo wire (PW) standard, which used MPLS label-switched paths (LSPs) inside MPLS tunnels to transparently connect Ethernet LANs. PW strategy supports both point-to-point Virtual Private Wire Service (VPWS) and multipoint Virtual Private LAN Service (VPLS) services. Following this, RFC 4448 defined Ethernet over MPLS (EoMPLS), which expanded PW functionality to further emulate Ethernet LAN capabilities.

Cisco IP Next-Generation Network (NGN) Carrier Ethernet architecture goes one step further: Recognizing that nearly all traffic originates as Ethernet, the design goal of the architecture is for Ethernet to be pervasive throughout the SP network, either in native form or as an virtualized/emulated service. The underpinning technologies of IP NGN include MPLS VPNs, EoMPLS, Hierarchal Virtual Private LAN Service (H-VPLS), IP over dense wavelength-division multiplexing (IPoDWDM), IEEE 802.1ad, and emerging Ethernet, IP, and MPLS technologies. Figure 31-2 illustrates the Cisco IP NGN architecture.

The key takeaway here is that Ethernet is already emerging as the dominant protocol for MAN and WAN services, which can present some unique implications from a QoS design perspective, as discussed next.

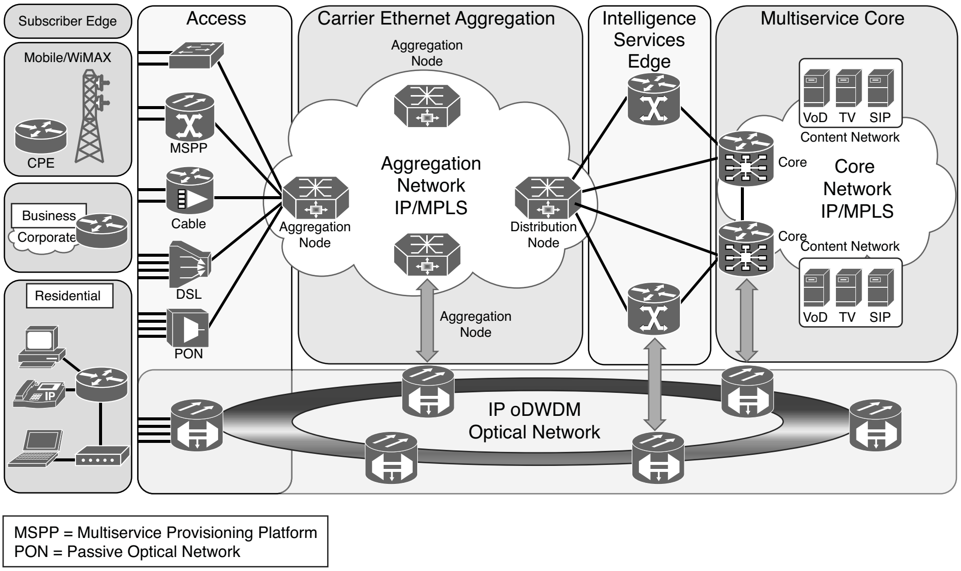


Figure 31-2 *Cisco IP NGN Carrier Ethernet Architectures*

Sub-Line-Rate Ethernet Design Implications

Recommendation:

- Understand the QoS implications of sub-line-rate Ethernet access (specifically the need for hierarchical shaping with nested queuing policies on CE router access links).
- Configure the CE shaper's committed burst (Bc) to be no more than half the value of the SP policer's Bc.

The ubiquity and flexibility of Ethernet make it an attractive choice as a VPN access media for both enterprise customers and SPs alike. Enterprise subscribers no longer have to buy special linecards and modules to support WAN protocols, like ATM and POS. Similarly, SPs can simplify their hardware requirements as well. Furthermore, SPs can offer both flexibility and scalability in their service offerings by presenting subscribers with sub-line-rate Ethernet services.

Sub-line-rate Ethernet services (as the name implies) describes a service contract where the traffic rate offered is less than a GE link's capacity. It may be as low as 1 Mbps or as high as 999 Mbps. In addition, the contract is flexible; it can be increased without requiring a hardware upgrade, as is often the case with a fixed-speed WAN circuits. Consider, for example, a point-to-point WAN link running at T3/DS3 (45-Mbps) speeds. If more bandwidth would be required on this link, a hardware upgrade on the module would be needed, and the next generally supported speed for such a link would be OC3 (155 Mbps), which is a significant upgrade in speed (more than 300 percent). In contrast, if a subscriber was given a 45 Mbps sub-line Ethernet service, he could upgrade in nearly any increment of bandwidth the SP is prepared to accommodate *without* any corresponding hardware upgrade. For example, if the SP supports 5 Mbps increments, the subscriber could upgrade gradually—and cost-effectively—to meet his evolving business requirements, say to 50 Mbps, then 55 Mbps, and so on.

However, from a QoS perspective, sub-line rates require additional attention. As previously discussed, queuing policies only engage when the physical interface is congested (as is indicated to Cisco IOS Software by a full Tx-Ring). This means that queuing policies never engage on media that has a contracted sub-line rate of access, whether this media is Frame Relay, ATM, or Ethernet. In such a scenario, queuing can only be achieved at a sub-line rate by introducing a two-part policy, sometimes referred to a Hierarchical QoS (HQoS) policy or nested QoS policy, wherein

1. Traffic is shaped to the sub-line rate.
2. Traffic is queued according to the LLQ/CBWFQ policies within the sub-line rate.

With such an HQoS policy, it is not the Tx-Ring that signals IOS Software to engage LLQ/CBWFQ policies, but rather it is the class-based shaper that triggers software queuing when the shaped rate has been reached. Such an HQoS policy is graphically illustrated in Figure 31-3.

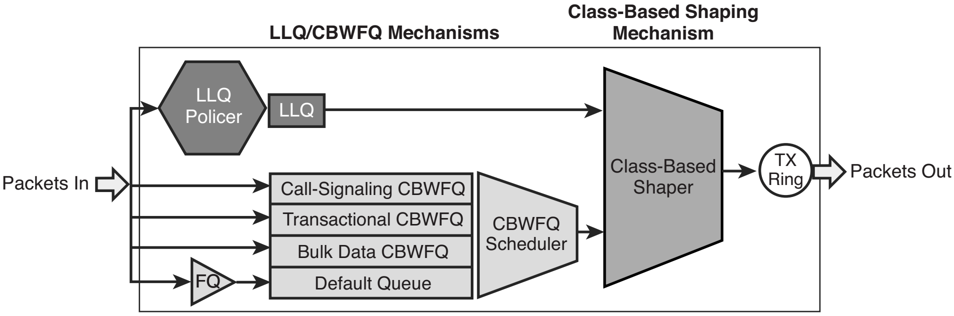


Figure 31-3 *Cisco IOS Hierarchical QoS (Shaping with Nested Queuing) Policy Operation*

In addition, a special relationship exists between how the shaper is to be configured relative to the SP's ingress policer. Both the shaper and the policer operate as token-bucket algorithms (as described in Chapter 4, "Policing, Shaping, and Markdown Tools"). However, it is critically important that the shaper's committed burst value be less than the ingress policer's committed burst; otherwise, traffic passed by the shaper may simply be dropped by the provider's ingress policer.

Shapers work by introducing delay. Ideally, the idle time is distributed between each packet. Only hardware-based shapers, such as those found in the Cisco Catalyst 3750-

Metro switch, can do this. Cisco IOS shapers use a software algorithm to enforce packets to delay. Cisco IOS-based shapers follow the formula:

Committed burst (Bc) = Committed information rate (CIR) * Time interval (Tc)

The target bandwidth (CIR) is divided into fixed time slices (Tc). Each Tc can send only Bc bytes worth of data. Additional traffic must wait for the next available time slice. In general, the smaller the time slice, the smoother the traffic is shaped. This algorithm is generally effective, but keep in mind some details. First, the minimum time slice that IOS shapers can meter traffic into is 4 ms. This means that idle time cannot be evenly distributed between all packets. Within a time slice, the interface still sends packets at line rate. If the queue of packets waiting is deeper than Bc bytes, all the packets are sent in sequence at the start of each Tc, followed by an idle period. In effect, if the offered rate exceeds the CIR rate for an extended period, the shaper introduces microbursts that are limited to Bc in size. Each time slice is independent of the previous time slice. A burst of packets may arrive at the shaper and completely fill a Bc at the very last moment, followed immediately by a new time slice with another Bc worth of available bandwidth. This means that although the interface routinely runs at line rate for each Bc worth of data, it is possible that it will run at line rate for 2 * Bc worth of bytes. When a shaper first becomes active, the traffic alignment in the previous Tc is not considered.

Partial packets are another feature of shapers to consider. Partial packets occur when a packet arrives whose length exceeds the remaining Bc bits available in the current time slice. There are two possible approaches to handle this. First, delay the packet until there are enough bits available in the bucket. The downside of this approach is twofold. First, the interface is not able to achieve CIR rate because time slices are expiring with bits still left in the Bc bucket. Second, although there might not be enough Bc bits for a large packet, there could be enough bits for a much smaller packet in queue behind the large packet. There are problems with trying to search the queue looking for the best use of the remaining Bc bits. Instead, the router allows the packet to transmit by borrowing some bits from the next time slice. Figure 31-4 shows the impact of using shapers.

In this MPLS VPN design context, the shaper is configured on customer equipment to ensure that traffic is not sent out of contract. In contrast, the SP uses a policer to enforce a contracted rate. The net effect is that shapers are often used to prevent upstream policers from dropping packets. Typically, the policer is set in place without regard to customer shapers. If the customer knows what the parameters of the policer are, this knowledge can be used to correctly configure a shaper.

Understanding the difference between policers and shapers helps in understanding the difference in implementation. First, a policer does not queue any packets. Any packets that do not conform are dropped. The shaper is the opposite. No packets are dropped until all queue memory is starved. Policers do not require the router to perform an action; instead, the router only reacts. Shaping is an active process. Queues must be managed. Events are triggered based on the fixed Tc timer. The algorithm for shaping is to maintain a token bucket. Each Tc seconds, Bc tokens are added to the bucket. When a packet arrives, the bucket is checked for available tokens. If there are enough tokens, the packet is allowed onto the Tx-Ring and the token bucket is debited by the size of the packet.

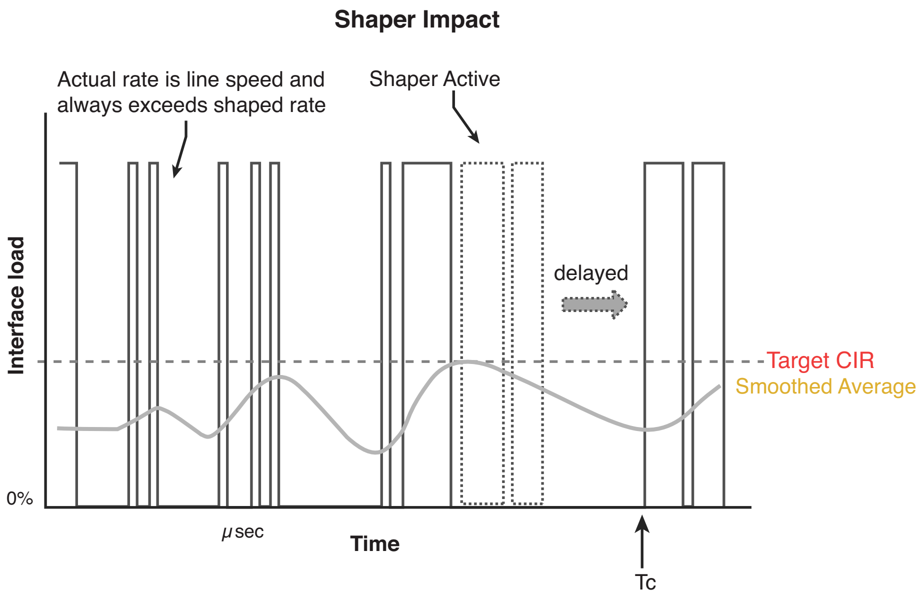


Figure 31-4 *Cisco IOS Shaper Operation and Impact*

If the bucket does not have enough tokens, the packet must wait in queue. At each Tc interval, Bc tokens are credited to the bucket. If there are packets waiting in queue, these packets can be processed until either the queue is empty or the bucket is again depleted of tokens.

By contrast, policing is a passive process. There is no time constant and no queue to manage. A simple decision is made to pass or drop a packet. With policing, the token bucket initially starts full with Bc tokens. When a packet arrives, the time interval since the last packet is calculated. The time elapsed is multiplied by the CIR to determine how many tokens should be added to the bucket. After these tokens have been credited, the size of the packet is compared with the token balance in the bucket. If there are available tokens, the packet is placed on the Tx-Ring and the size of the packet is subtracted from the token bucket. If the bucket does not have enough available tokens, the packet is dropped. As the policed rate approaches the interface line rate, the size of the bucket become less important. When CIR = line rate, the bucket refills at the same rate that it drains.

Because tokens are added based on packet arrival times, and not as periodic events as is done with shapers, there is no time constant (Tc) when discussing policers. The closest equivalent is the time required for an empty bucket to completely refill if no additional packets arrive. In an ideal case, a shaper sends Bc bytes at line rate, which completely drains the policer Bc bucket. The enforced idle time of the shaper for the remaining Tc time then allows the Bc bucket of the policer to completely refill. The enforced idle time of the shaper is Tc * (1 – CIR / Line rate). *In practice, it is best to set the shaper so that the policer Bc bucket does not go below half full. This is done by ensuring that when the shaped CIR equals the policed CIR, the shaper Bc should be half of the policer Bc.*

QoS Paradigm Shift

Recommendation:

- Recognize that enterprises and service providers must cooperate to jointly administer QoS over MPLS VPNs.

With the Layer 2 QoS considerations of MPLS VPN access in place, a network administrator can now proceed with Layer 3 QoS considerations.

As mentioned, MPLS VPNs offer a full mesh of connectivity between campus and branch networks. It is this fully meshed characteristic of MPLS VPNs that presents a significant QoS design implication, as compared to traditional Layer 2 private WAN QoS design: private WANs (which are usually deployed in either a point-to-point or a hub-and-spoke topology).

Because of cost, scalability, and manageability constraints, traditional private WAN designs rarely use full-mesh models. Instead, most Layer 2 WAN designs revolve around a hub-and-spoke model, implementing either a centralized hub design or the more efficient regional hub design. Under such hub-and-spoke designs, QoS primarily is administered at the hub router (a.k.a. the WAN aggregator) by the enterprise. As long as the SP meets the contracted service levels, the packets received at remote branches will reflect the scheduling policies of the hub router. The WAN aggregator controls not only campus-to-branch traffic, but also branch-to-branch traffic (which is homed through the hub). Under traditional hub-and-spoke models, QoS principally is administered by the enterprise customer, as shown in Figure 31-5.

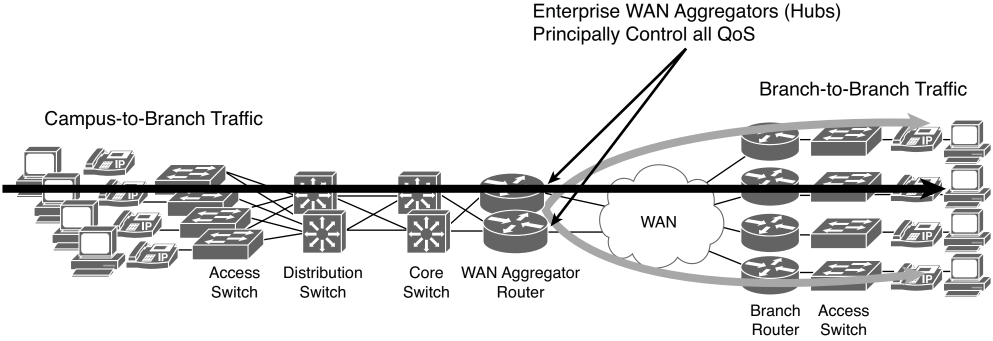


Figure 31-5 *QoS Administration in Traditional Hub-and-Spoke Layer 2 Private WAN Design*

However, with MPLS VPN service offerings that inherently offer full-mesh connectivity, the QoS administration paradigm shifts. Under a full-mesh design, the hub router still administers QoS for all campus-to-branch traffic, but it no longer fully controls the QoS for branch-to-branch traffic. Although it might appear that the only required workaround for this new scenario is to ensure that QoS is provisioned on all branch routers, this is insufficient because it addresses only part of the issue.

For example, consider the case of provisioning any-to-any multimedia-conferencing. As with a traditional Layer 2 WAN design, a scheduling policy to prioritize multimedia conferencing on the WAN aggregator is required. Then the enterprise must properly provision similar priority scheduling for multimedia conferencing on the branch routers also. In this manner, any multimedia-conferencing calls from the campus to the branch (and also from branch to branch) are protected against traffic of lesser importance flowing between the same sites. The complexity of the fully meshed model arises when considering that contending traffic might not always come for the same sites, but could come from *any* site. Furthermore, the enterprise no longer fully controls QoS for branch-to-branch traffic because this traffic no longer is homed through a hub. Continuing the example, if a multimedia-conferencing call is set up between two branches and a user from one of the branches also initiates a large FTP download from the central site, the potential for oversubscription of the PE-to-CE link from the fully meshed MPLS VPN cloud into one of the branches becomes very real, likely causing drops from the multimedia-conferencing call.

The only way to guarantee service levels in such a scenario is for the SP to provision QoS scheduling that is compatible with the enterprise's policies on all PE links to remote branches. This is what creates the paradigm shift in QoS administration for fully meshed topologies—namely, *enterprises and SPs must cooperate to jointly administer QoS over MPLS VPNs*, as shown in Figure 31-6.

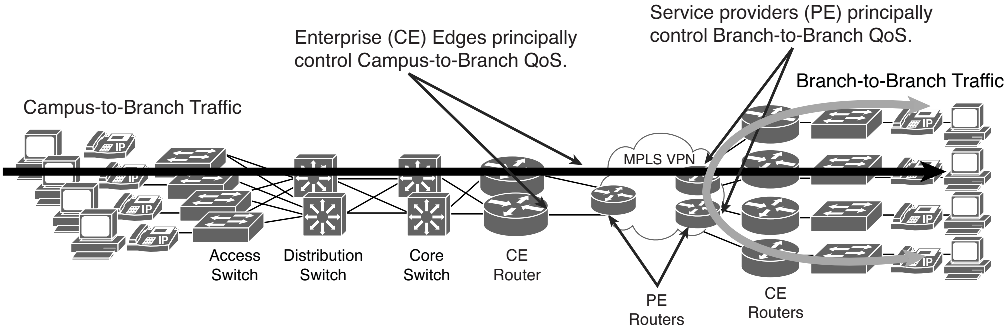


Figure 31-6 *QoS Administration in Fully Meshed MPLS VPN Design*

Therefore, queuing policies are mandatory on CE and PE router egress edges because of the full-mesh implications of MPLS VPNs. In addition, PE routers will have ingress policing policies to enforce service level agreements (SLAs).

QoS policies on P routers are optional. Such policies are optional because some SPs overprovision their MPLS core networks and, as such, do not require any additional QoS policies within their backbones; however, other providers might implement simplified differentiated services (DiffServ) policies within their cores or might even deploy MPLS Traffic Engineering (MPLS TE) to handle congestion scenarios within their backbones.

Service Provider Class of Service Models

Recommendations:

- Fully understand your SP's CoS models.
- If options exist, select the model that most closely matches your strategic end-to-end model.

It is up to the SP to define the class of service (CoS) models they will offer to their subscribers. There is no one-size-fits-all model, because these CoS models are often a key component of an SP's competitive differentiation strategy. Nonetheless, at the time of this writing, most SPs are offering four- and six-class QoS models, with a few offering eight (or more) classes of service.

Admission to a given CoS will depend on packet markings (usually differentiated services code point [DSCP]). However, the per-class specific markings will vary, as will the re-marking/dropping policies of a given provider and the bandwidth allocations per class.

Whenever multiple SP models are presented as options, it is recommended that the subscriber chooses the model that most closely aligns with his strategic end-to-end QoS model (and not vice versa). In other words, an administrator should not choose his strategic QoS model based on what CoS models his provider is offering.

MPLS DiffServ Tunneling Modes

Recommendations:

- Understand the MPLS DiffServ tunneling modes and how these affect subscriber DSCP markings.
- Short Pipe Mode offers enterprise customers the most transparency and control of their traffic classes as these traverse the MPLS VPN cloud.

Because MPLS labels include 3 experimental bits that commonly are used for QoS marking, it is possible to "tunnel DiffServ"—that is, preserve Layer 3 DiffServ markings through a SP's MPLS VPN cloud while still performing re-marking (via MPLS EXP bits) within the cloud to indicate in- or out-of-contract traffic.

RFC 3270 defines three distinct modes of MPLS DiffServ tunneling:

- Uniform Mode
- Short Pipe Mode
- Pipe Mode

Each of these MPLS DiffServ tunneling models is discussed in turn.

Uniform Mode

Recommendation:

- Be aware that your packets can have their DSCP values re-marked if your service provider is operating in Uniform Mode MPLS VPN DiffServ tunneling.

Uniform Mode is generally used when the customer and SP share the same DiffServ domain, as in the case of an enterprise deploying its own MPLS VPN core.

In Uniform Mode, which is the default mode, the first 3 bits of the IP ToS field (IP Precedence bits) automatically are mapped to the MPLS EXP bits on the ingress PE as labels are pushed onto the packets.

If policers or any other mechanisms re-mark the MPLS EXP values within the MPLS core, these marking changes are propagated to lower-level labels and eventually are propagated to the IP ToS field. (MPLS EXP bits are mapped to IP Precedence values on the egress PE.)

Figure 31-7 shows the behavior of Uniform Mode MPLS DiffServ tunneling.

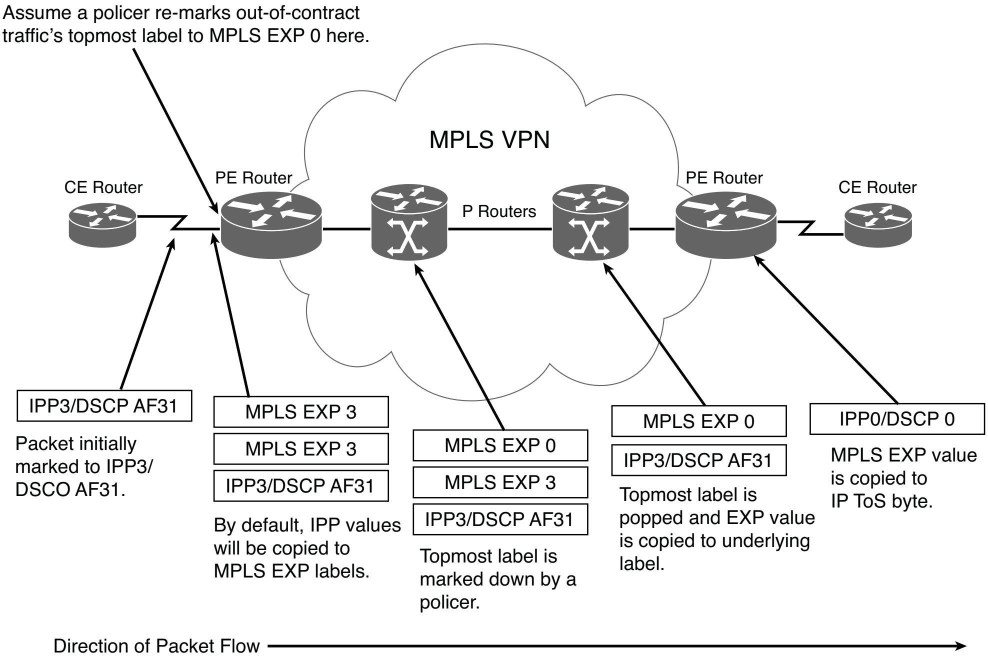

Assume a policer re-marks out-of-contract traffic's topmost label to MPLS EXP 0 here.
MPLS VPN
CE Router
PE Router
P Routers
PE Router
CE Router
IPP3/DSCP AF31
Packet initially marked to IPP3/ DSCO AF31.
MPLS EXP 3
MPLS EXP 3
IPP3/DSCP AF31
By default, IPP values will be copied to MPLS EXP labels.
MPLS EXP 0
MPLS EXP 3
IPP3/DSCP AF31
Topmost label is marked down by a policer.
MPLS EXP 0
IPP3/DSCP AF31
Topmost label is popped and EXP value is copied to underlying label.
IPP0/DSCP 0
MPLS EXP value is copied to IP ToS byte.
Direction of Packet Flow

Figure 31-7 *MPLS DiffServ Uniform Tunneling Mode Operation*

As shown in Figure 31-7, the enterprise customer's DSCP markings have been re-marked in transit by the SP in MPLS Uniform DiffServ Tunneling Model.

Short Pipe Mode

Recommendation:

- For enterprise customers who prefer transparency and maximum control over their traffic, choose Short Pipe Mode MPLS DiffServ tunneling.

Short Pipe Mode is used when the customer and SP are in different DiffServ domains. (The SP's DiffServ domain begins at the ingress PE's ingress interface and terminates on the egress PE's ingress interface.)

This mode is useful when the SP wants to enforce its own DiffServ policy and the customer requests that its DiffServ information be preserved through the MPLS VPN cloud. Short Pipe Mode provides DiffServ transparency through the SP network (as does Pipe Mode).

In Short Pipe Mode, the outmost label is used as the single most meaningful information source because it relates to the SP's QoS per-hop behavior (PHB). On MPLS label imposition, the IP classification is not copied into the outermost label's EXP. Instead, the value for the MPLS EXP is set explicitly on the ingress PE's ingress interface, according to the SP's administrative policies.

In the case of any re-marking occurrence within the SP's MPLS VPN cloud, changes are limited to MPLS EXP re-marking only and are not propagated down to the underlying IP packet's ToS byte. Figure 31-8 shows the operation of Short Pipe Mode MPLS DiffServ tunneling.

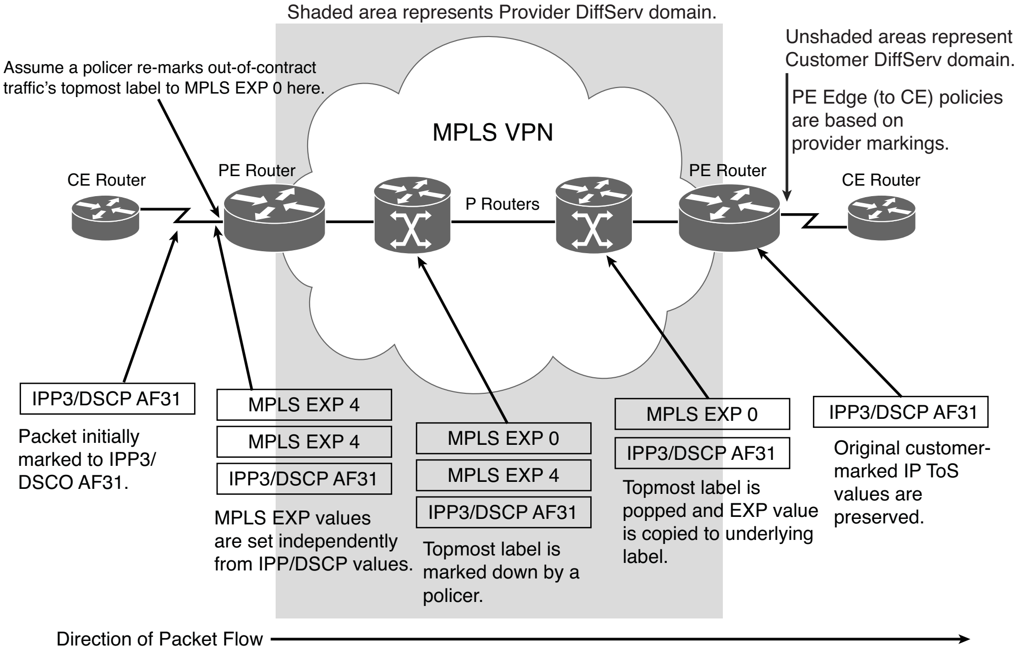

Shaded area represents Provider DiffServ domain.
Unshaded areas represent Customer DiffServ domain.
Assume a policer re-marks out-of-contract traffic's topmost label to MPLS EXP 0 here.
PE Edge (to CE) policies are based on provider markings.
MPLS VPN
CE Router
PE Router
P Routers
PE Router
CE Router
IPP3/DSCP AF31
Packet initially marked to IPP3/ DSCO AF31.
MPLS EXP 4
MPLS EXP 4
IPP3/DSCP AF31
MPLS EXP values are set independently from IPP/DSCP values.
MPLS EXP 0
MPLS EXP 4
IPP3/DSCP AF31
Topmost label is marked down by a policer.
MPLS EXP 0
IPP3/DSCP AF31
Topmost label is popped and EXP value is copied to underlying label.
IPP3/DSCP AF31
Original customer-marked IP ToS values are preserved.
Direction of Packet Flow

Figure 31-8 *MPLS DiffServ Short Pipe Mode Tunneling Operation*

As shown in Figure 31-8, MPLS EXP values can be marked in any way that the SP wants to convey local significance (and have no relation to the customer's packet-marking values). Therefore, the customer's markings are preserved in transit and are available to him as the packet exits the MPLS VPN. Furthermore, the final PE egress queuing policies are based on the customer's markings, granting him maximum control of the packet's QoS treatment through the MPLS VPN.

Pipe Mode

Recommendation:

- Pipe Mode will preserve customer packet markings, but these will not influence PE egress queuing policies.

The main difference between Short Pipe Mode and Pipe Mode MPLS DiffServ tunneling is that the PE egress policies (toward the customer CEs) are provisioned according to the SP's explicit markings and re-markings, not the enterprise customer's IP DiffServ markings (although these are preserved). As with Short Pipe Mode, any changes to label markings that occur within the SP's cloud do not get propagated to the IP ToS byte when the packet leaves the MPLS network. Figure 31-9 illustrates the Pipe Mode MPLS DiffServ tunneling operation.

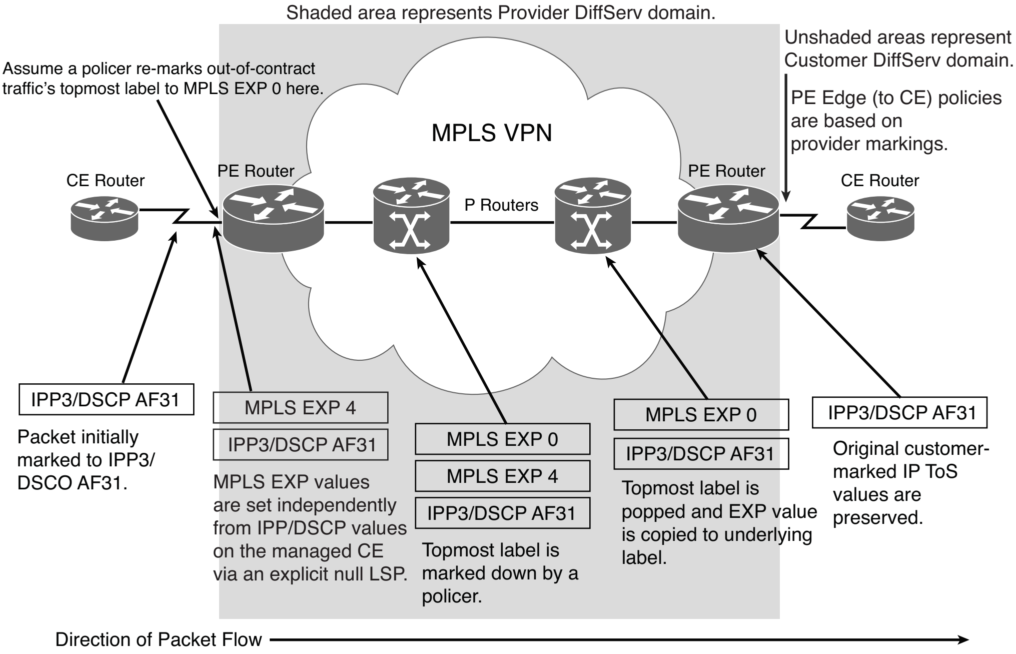

Shaded area represents Provider DiffServ domain.
Unshaded areas represent Customer DiffServ domain.
Assume a policer re-marks out-of-contract traffic's topmost label to MPLS EXP 0 here.
PE Edge (to CE) policies are based on provider markings.
MPLS VPN
CE Router
PE Router
P Routers
PE Router
CE Router
IPP3/DSCP AF31
Packet initially marked to IPP3/ DSCO AF31.
MPLS EXP 4
IPP3/DSCP AF31
MPLS EXP values are set independently from IPP/DSCP values on the managed CE via an explicit null LSP.
MPLS EXP 0
MPLS EXP 4
IPP3/DSCP AF31
Topmost label is marked down by a policer.
MPLS EXP 0
IPP3/DSCP AF31
Topmost label is popped and EXP value is copied to underlying label.
IPP3/DSCP AF31
Original customer-marked IP ToS values are preserved.
Direction of Packet Flow

Figure 31-9 *MPLS DiffServ Pipe Mode Tunneling Operation*

As shown in Figure 31-9, Pipe Mode operation is identical to Short Pipe, with the sole exception being that the final PE egress queuing policies are based on the SP's markings (and not the customer's).

Enterprise-to-Service Provider Mapping

Recommendation:

- Efficiently map enterprise application classes to SP CoS classes, as needed.

Sometimes the number of SP CoS classes will match (or exceed) the number of application classes that an enterprise has defined in their strategic end-to-end QoS policy; however, this might not always be the case.

If the number of enterprise application classes exceeds the number of SP CoS classes, the enterprise administrator will need to map into the SP's model, tactically and efficiently collapsing and combining application classes and performing any required re-marking in the process.

Enterprise-to-service provider mapping considerations include the following:

- Mapping real-time voice and video traffic
- Mapping signaling and control traffic
- Separating TCP-based applications from UDP-based applications (where possible)
- Re-marking and restoring packet markings (where required)

The sections that follow discuss each of these enterprise-to-service provider mapping considerations.

Mapping Real-Time Voice and Video

Recommendation:

- Balance service level requirements for real-time voice and video applications with SP premiums for real-time bandwidth
- In either scenario, use a dual-LLQ policy at the CE egress edge

SPs usually offer only a single real-time CoS. Therefore, if you—as an enterprise network administrator—are deploying both real-time voice and video (that ism, the Real-Time Interactive application class), you will have to choose whether to assign the video to the SP real-time class or not.

Choosing to map both real-time voice and video into the SP real-time CoS will result in the highest service levels for these real-time applications. The downside of this choice is that SPs typically sell their real-time bandwidth at a premium. Therefore, it might be expensive (even cost-prohibitive) to assign video to this class. If this option is chosen, however, then—as with private WAN designs—it is recommended to use a dual-LLQ (or multi-LLQ) policy to protect voice from video on the CE egress edge

However, choosing to relegate real-time video to a non-real-time SP CoS will usually be more cost-effective, often with only a minor downgrade in video quality. If this option is chosen, you should still use a dual-LLQ (or multi-LLQ) policy on the CE egress edge. (After all, that node is still under your administrative control.) In this manner, it is only at the PE egress edge where the video will receive a non-real-time service, minimizing the potential service level loss at what may be considerable cost-savings.

Mapping Control and Signaling Traffic

Recommendation:

- Avoid mixing control plane traffic with data plane traffic in a single SP CoS.

Whenever possible, control and signaling traffic should be assigned to a dedicated SP CoS. Otherwise, if such traffic were combined with data plane traffic, these could be dropped when the class is oversubscribed, resulting in network or voice/video-service instability. Sometimes SPs automatically provision network control traffic (marked DSCP CS6) into dedicated CoS.

If the number of CoS classes precludes the dedication of one class exclusively for control or signaling traffic, consider mapping these control plane flows to the SP's real-time CoS (because these flows are usually lightweight, yet critical).

Separating TCP from UDP

Recommendation:

- Separate TCP traffic from UDP traffic when mapping to SP CoS classes.

It is a general best practice to not mix TCP-based traffic with UDP-based traffic (especially if the UDP traffic is streaming video, such as the broadcast video or multimedia-streaming application classes) within a single SP CoS. This is because of the behaviors of these protocols during periods of congestion. Specifically, TCP transmitters throttle back flows when drops are detected. Although some UDP applications have application-level windowing, flow control, and retransmission capabilities, most UDP transmitters are completely oblivious to drops and, therefore, never lower transmission rates because of dropping.

When TCP flows are combined with UDP flows within a single SP CoS and the class experiences congestion, TCP flows continually lower their transmission rates, potentially giving up their bandwidth to UDP flows that are oblivious to drops. This effect is called TCP starvation/UDP dominance.

Even if weighted random early detection (WRED) is enabled on the SP class, the same behavior would be observed because WRED (primarily) manages congestion only on TCP-based flows.

Granted, it is not always possible to separate TCP-based flows from UDP-based flows, but it is beneficial to be aware of this behavior when making such application-mixing decisions within a single SP class.

Re-Marking and Restoring Markings

Recommendation:

- Re-mark application classes on the CE edge on *egress* (as required).
- Restore markings on the CE edge on *ingress* via deep packet inspection policies (as required).

Most SPs use the DSCP marking of packets offered to them to determine which SP CoS the packet should be assigned to. Therefore, enterprises must mark (or re-mark) their traffic consistent with their SP's admission criteria to gain the appropriate level of service.

A general DiffServ principle is to mark or trust traffic as close to the source as administratively and technically possible. However, certain traffic types might need to be re-marked before handoff to the SP to gain admission to the correct class. If such re-marking is required, it is recommended that the re-marking be performed at the CE's egress edge, not within the campus. This is because SP service offerings likely will evolve or expand over time, and adjusting to such changes will be easier to manage if re-marking is performed only at the CE egress edge.

In some cases, multiple types of traffic may be required to be re-marked to the same DSCP value to gain admission to the appropriate SP CoS. Alternatively, (as previously discussed) SPs operating Uniform Mode MPLS DiffServ tunneling might re-mark out-of-contract traffic at Layer 3 within their cloud, which might affect enterprises that require consistent end-to-end Layer 3 markings.

In either case, on exiting the MPLS VPN, these traffic classes will no longer be distinguishable one from another by DSCP markings alone. However, these DSCP markings can easily be restored using the same LAN-edge deep packet inspection policies (discussed in Chapter 29, "Branch Router (Cisco ISR G2) QoS Design") but applied on *ingress* on the CE router's VPN edge (that is, reclassifying traffic received from the MPLS VPN cloud that may have had its DSCP markings lost in transit).

MPLS VPN QoS Roles

Recommendation:

- Define consistent ingress and egress QoS policies for all router interface roles.

The QoS policy elements discussed so far can be grouped into roles that various router nodes serve within the MPLS VPN architecture, as illustrated in Figure 31-10.

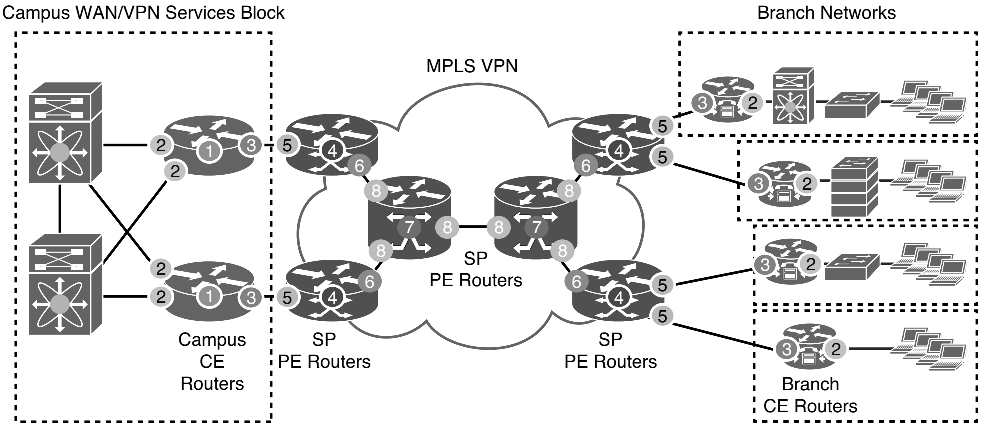


Figure 31-10 *MPLS VPN QoS Roles*

The specific QoS policies for these roles are as follows:

1. Campus CE ingress/internal QoS (ASR 1000): Ingress/internal QoS policies *may* be applied (if required).
2. CE LAN edge:
   - Ingress DSCP-trust *should* be enabled (enabled by default).
   - Ingress NBAR2 classification and marking policies *may* be applied.
   - Ingress Medianet metadata classification and marking policies *may* be applied.
   - Egress LLQ/CBWFQ/WRED policies *may* be applied (if required).
3. CE VPN edge:
   - Ingress DSCP-trust *should* be enabled (enabled by default).
   - Ingress NBAR2 classification and marking policies *may* be applied (to restore DSCP markings lost in transit).
   - Ingress Medianet metadata classification and marking policies *may* be applied (to restore DSCP markings lost in transit).
   - RSVP policies *may* be applied.
   - Egress LLQ/CBWFQ/WRED policies *should* be applied.
   - Egress hierarchical shaping with nested LLQ/CBWFQ/WRED policies *may* be applied.
   - Egress DSCP re-marking policies *may* be applied (to map application classes into specific SP classes of service).

4. PE ingress/internal QoS (ASR 9000): Ingress/internal QoS policies *may* be applied (if required).
5. PE customer-facing edge:
   - Ingress DSCP trust *should* be enabled (enabled by default).
   - Ingress policing policies to meter customer traffic *should* be applied.
   - Ingress MPLS tunneling mode policies *may* be applied.
   - Egress MPLS tunneling mode policies *may* be applied.
   - Egress LLQ/CBWFQ/WRED policies *should* be applied.
6. PE core-facing edge:
   - Ingress DSCP-trust *should* be enabled (enabled by default).
   - Ingress policing policies to meter customer traffic *should* be applied.
   - Egress MPLS EXP-based LLQ/CBWFQ policies *should* be applied.
   - Egress MPLS EXP-based WRED policies *may* be applied.
7. P (core router) ingress/internal QoS (CRS-3): Ingress/internal QoS policies *may* be applied (if required).
8. P edges:
   - Ingress DSCP trust *should* be enabled (enabled by default).
   - Egress MPLS EXP-based LLQ/CBWFQ policies *may* be applied (unless core is overprovisioned or has MPLS Traffic Engineering enabled).
   - Egress MPLS EXP-based WRED policies *may* be applied.

Summary

This chapter discussed the QoS design considerations and recommendations for enterprise MPLS VPNs. The role of QoS in the MPLS VPN includes the following:

- Managing packet loss and jitter
- Shaping traffic to contracted service rates
- Performing hierarchical queuing and dropping within these shaped rates
- Mapping enterprise-to-service provider CoS markings
- Policing traffic classes according to contracted rates
- Restoring packet markings

Strategic QoS design principles that apply in MPLS VPN networks include the following:

- Classify and mark applications as close to their sources as technically and administratively feasible.
- Police unwanted traffic flows as close to their sources as possible.
- Enable queuing policies at every node where the potential for congestion exists.

MPLS VPN architectures were overviewed, highlighting the increasingly relevant role of Ethernet over the MAN and WAN. This led to a discussion of the QoS design implications of sub-line-rate Ethernet circuits and their need for hierarchical (shaping with nested queuing) policies, with the shaper's committed burst set to half of the SP's ingress policer's committed burst.

The QoS paradigm shift of how enterprises and SPs must cooperate to jointly administer QoS over MPLS VPNs was examined, along with SP CoS models.

Next, MPLS DiffServ tunneling modes were presented, illustrating how the Uniform Mode may result in the overriding of the customer's DSCP markings, but how Pipe Mode and Short Pipe mode offer DSCP transparency to the customer. In addition, the advantage of Short Pipe Mode over Pipe Mode was highlighted.

Various enterprise-to-service mapping considerations were discussed, including how to map real-time voice and video, how to map control and signaling, and why it is good to separate TCP from UDP. Re-marking implications were examined, as was how to restore DSCP markings that may have been lost in transit.

Finally, all these recommendations were summarized and grouped into various specific MPLS VPN interface QoS roles, including the following:

- Campus CE ingress/internal QoS (ASR-only)
- CE LAN edge
- CE VPN edge
- PE customer-facing edge
- PE core-facing edge
- P edges

Further Reading

RFC 2547, BGP MPLS VPNs: http://www.ietf.org/rfc/rfc2547

RFC 3270, Multi-Protocol Label Switching (MPLS) Support of Differentiated Services: http://www.ietf.org/rfc/rfc3270

RFC 3985, Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture: http://www.ietf.org/rfc/rfc3985

RFC 4448, Encapsulation Methods for Transport of Ethernet over MPLS Networks: http://www.ietf.org/rfc/rfc4448

IP NGN Carrier Ethernet Design: Powering the Connected Life in the Zettabyte Era: http://www.cisco.com/en/US/solutions/collateral/ns341/ns524/ns562/ns577/net_implementation_white_paper0900aecd806a7df1_ns537_Networking_Solutions_White_Paper.html

This page intentionally left blank

Chapter 32

Enterprise Customer Edge (Cisco ASR 1000 and ISR G2) QoS Design

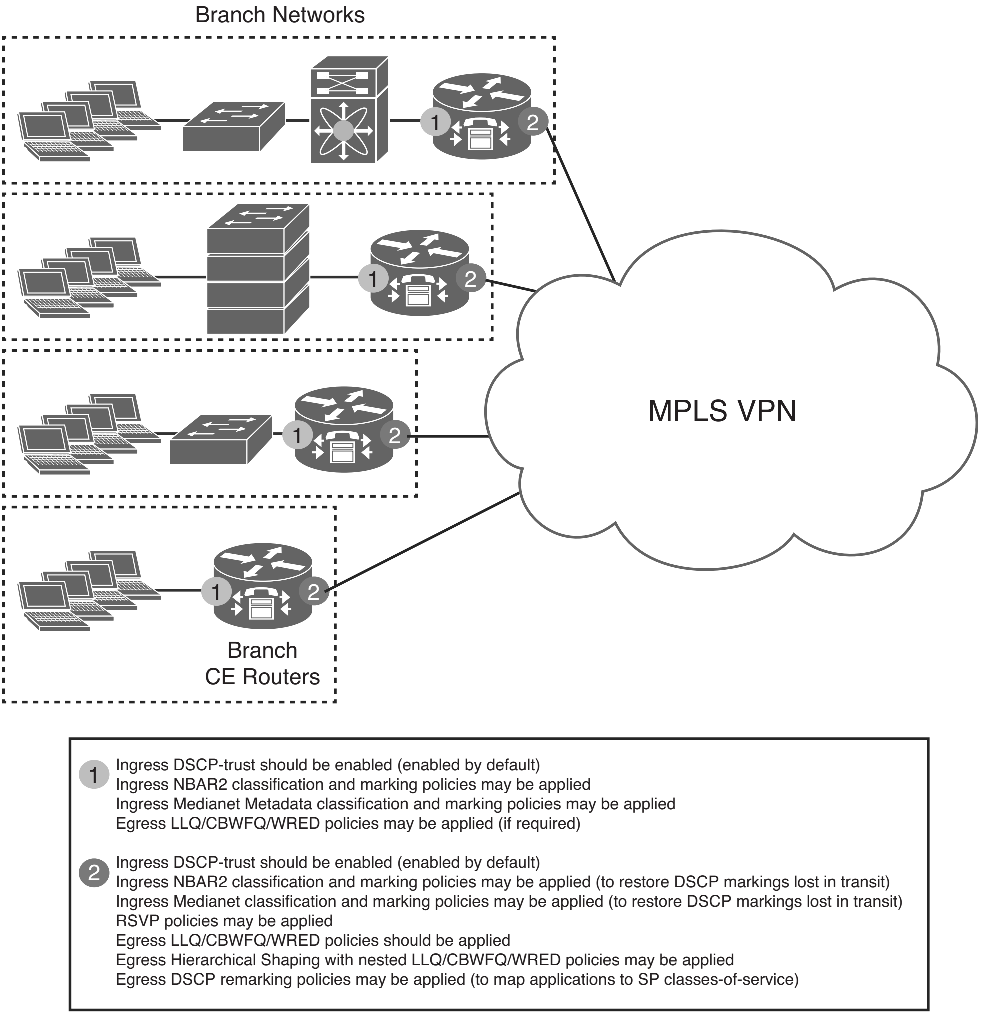
The enterprise customer-edge router has many quality of service (QoS) roles to play, including the following:

- Shaping traffic to contracted sub-line service rates
- Performing hierarchical queuing and dropping within these shaped rates to manage packet loss and jitter
- Mapping enterprise-to-service provider class of service markings
- Restoring packet markings that may have been lost by applying deep packet inspection classification on traffic received from the MPLS VPN cloud

Figure 32-1 illustrates the interface-specific QoS roles of enterprise customer-edge routers.

Both the ASR 1000 and ISR G2 routers may service in the role of a customer-edge router—the router used will depend on the size of the branch (or campus) connecting to the MPLS VPN and the speeds of the links.

Because these platform architectures have already been overviewed (in Chapter 28, "WAN Aggregation (Cisco ASR 1000) QoS Design," and Chapter 29, "Branch Router (Cisco ISR G2) QoS Design"), we will proceed directly to the QoS design steps for configuring this platform as an enterprise customer-edge router.



1 Ingress DSCP-trust should be enabled (enabled by default)
Ingress NBAR2 classification and marking policies may be applied
Ingress Medianet Metadata classification and marking policies may be applied
Egress LLQ/CBWFQ/WRED policies may be applied (if required)

2 Ingress DSCP-trust should be enabled (enabled by default)
Ingress NBAR2 classification and marking policies may be applied (to restore DSCP markings lost in transit)
Ingress Medianet classification and marking policies may be applied (to restore DSCP markings lost in transit)
RSVP policies may be applied
Egress LLQ/CBWFQ/WRED policies should be applied
Egress Hierarchical Shaping with nested LLQ/CBWFQ/WRED policies may be applied
Egress DSCP remarking policies may be applied (to map applications to SP classes-of-service)

Figure 32-1 *Enterprise Customer-Edge Router Interface QoS Roles*

QoS Design Steps

There are two main steps to configure QoS on a Cisco ISR G2 router performing the role of an enterprise customer-edge router:

1. Configure ingress QoS models, including the following:
   - Trust DSCP (enabled by default in IOS)
   - Medianet metadata classification and marking
   - NBAR2 classification and marking

2. Configure egress QoS policies, including the following:
   - Queuing and dropping policies
   - Hierarchical shaping (with nested queuing and dropping) policy (as required)

Each of these design steps is covered in turn.

Ingress QoS Models

Trust is enabled by default in Cisco IOS, so no explicit configuration is required to trust the DSCP of incoming packets.

As for classification and marking, any deep packet inspection policies (Medianet metadata application identification or NBAR2) are identical to the configurations detailed in Chapter 29.

Incidentally, such ingress LAN-edge policies may also be applied on the ingress VPN edge, to utilize DPI technologies to identify applications and restore markings that may have been lost in the MPLS VPN cloud (or in the process of enterprise-to-service provider mapping). From a configuration standpoint, this is a very simple addition: an extra **service-policy input** interface command on the VPN-edge interface is all that is required.

Egress QoS Models

The egress queuing models for enterprise customer-edge routers will be nearly identical to the WAN-edge queue models, although some bandwidth allocations may require adapting to the service provider class of service (CoS) models.

In addition to queuing and dropping policies, two additional policy elements may be required on enterprise customer-edge routers:

- Hierarchical shaping with nested queuing and dropping policies
- Enterprise-to-service provider mapping (re-marking) policies

Each of these is now discussed in turn.

Sub-Line-Rate Ethernet: Hierarchical Shaping and Queuing Models

Ethernet is rapidly becoming a major WAN/MAN access media, and often at sub-line rates. Because queuing and dropping policies will not engage in Cisco IOS at sub-line rates, an additional shaper is required to place backpressure on the queuing mechanisms to engage early (in other words, prior to physical interface congestion). Also, as covered in the previous chapter, it is recommended to set the shaper's committed burst to no more than half of the service provider's ingress policer's committed burst. This might not always be known, so two examples will be provided: one where the service provider's policing Bc is known and the other where it isn't.

Known SP Policing Bc

In this first example, the service provider is offering a four-class CoS model, with each class being policed to 10 Mbps with a Bc of 200 KB. Therefore, it is recommended that the shaper's Bc be set to no more than 800 Kb (200 KB * 8 bits per byte / 2). Incidentally this would yield a 20-ms shaping time interval (Tc = Bc / CIR). Example 32-1 shows the configuration for this hierarchical sub-line-rate shaping policy with nested queuing policy.

Example 32-1 *Cisco ISR G2 (CE Router) Hierarchical Shaping and Queuing Model Example*

```
 ! This section configures the hierarchical shaper
 ! with nested queuing policy
ISR-G2(config-pmap)# policy-map CE-EDGE-SHAPING
ISR-G2(config-pmap)# class class-default
ISR-G2(config-pmap-c)# shape average 40M 800K
 ! Shapes to 40 Mbps with 800 Kbit Bc
ISR-G2(config-pmap-c)# service-policy CE-EDGE-QUEUING
 ! Provides backpressure on the IOS queuing subsystem for the
 ! CE-EDGE-QUEUING queuing policy to engage
 ! when the shaping limit is reached



 ! This section applies the sub-line-rate shaper to the VPN edge int
ISR-G2(config)#interface GigabitEthernet0/2
ISR-G2(config-if)# service-policy output CE-EDGE-SHAPING
 ! Attaches the HQoS shaper to the CE Edge interface
```

Note It is true that the aggregate shaper in Example 32-1 could potentially permit any one of the traffic classes to oversubscribe their respective SP policed rates (with the extreme example being a single class offering a 40-Mbps flow with all other classes absent). However, this same caveat applies in exactly the same manner to line-rate handoff scenarios, because queuing policies do not engage until congestion is reached.

Nonetheless, it is worth noting that the tuned shaper in this example will operate at its maximum efficiency if the offered per-class traffic rates are in proportion to their provisioned rates.

An alternative option may be to introduce hierarchical shapers, with each per-class shapers in addition to an aggregate shaper. However, such a policy may be unnecessarily complex and CPU intensive. Furthermore, such a policy may also be wasteful of bandwidth (because SP policers more often re-mark—rather than drop—exceeding traffic).

You can verify the configuration in Example 32-1 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Unknown SP Policing Bc

In the case where the service provider's ingress policing Bc is not known, you may choose to set the hierarchical shaper's committed burst (Bc) value to correspond to a 10-ms interval, which has been Cisco's generic rich-media shaping recommendation for over a decade.

Note In the rare event that a 10-ms interval is still causing drops on the SP's PE-edge ingress policer, a lower shaping value can be used. The Cisco IOS minimum shaping committed burst corresponds to a minimum shaping interval of 4 ms. However, it is important to be aware that the smaller the value of the shaping interval, the more interrupts will be generated by the shaper to the QoS subsystem, and therefore the higher the marginal CPU impact of the QoS policy.

The tradeoff in this case is that extra shaping delay may be introduced, because packets may need to wait multiple (10-ms) shaping intervals before being transmitted. However, once sent out by the shaper, these are (generally) unlikely to be dropped by the ingress service provider policer (provided they do not exceed the per-class policing rate).

Continuing the previous example, Bc would work out to be 400 Kb (40-Mbps CIR * 10-ms Tc). Example 32-2 shows the configuration for this second hierarchical sub-line-rate shaping policy with nested queuing policy.

Example 32-2 *Cisco ISR G2 (CE Router) Hierarchical Shaping and Queuing Model Example*

```
 ! This section configures the hierarchical shaper
 ! with nested queuing policy
ISR-G2(config-pmap)# policy-map CE-EDGE-SHAPING
ISR-G2(config-pmap)# class class-default
ISR-G2(config-pmap-c)# shape average 40M 400K
ISR-G2(config-pmap-c)# service-policy CE-EDGE-QUEUING
 ! Provides backpressure on the IOS queuing subsystem for the
 ! CE-EDGE-QUEUING queuing policy to engage
 ! when the shaping limit is reached



 ! This section applies the sub-line-rate shaper to the VPN edge int
ISR-G2(config)#interface GigabitEthernet0/2
ISR-G2(config-if)# service-policy output CE-EDGE-SHAPING
 ! Attaches the HQoS shaper to the CE Edge interface
```

You can verify the configuration in Example 32-2 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Enterprise-to-Service Provider Mapping Models

As mentioned in the previous chapter, there is no one-size-fits-all service provider CoS model—as these vary and set providers apart from each other as competitive differentiators.

Therefore, to reflect real-world examples, three separate service provider models will be used for enterprise mapping examples, specifically a

- Four-CoS service provider model
- Six-CoS service provider model
- Eight-CoS service provider model

Similarly, as with previous design chapters, application-based strategic enterprise QoS models will include a 4-class model, an 8-class model, and a 12-class model. Each of these, in turn, will be mapped into a corresponding service provider class model, as will now be presented.

Four-Class Enterprise Model Mapped to a Four-CoS Service Provider Model

This four-to-four enterprise to service provider mapping is the simplest to configure, because no re-marking or class collapsing is required because a direct 1:1 relationship exists between the number of enterprise application classes and their corresponding service provider classes of service. The only configuration elements that may change are the bandwidth allocations per class, as determined by the service provider's policies.

Figure 32-2 shows a four-class enterprise model mapped to a four-class service provider model.

Example 32-3 shows the CE router configuration corresponding to Figure 32-2. To round out the example, this queuing policy is applied to the hierarchical (sub-line-rate) shaping policy presented in Example 32-1.

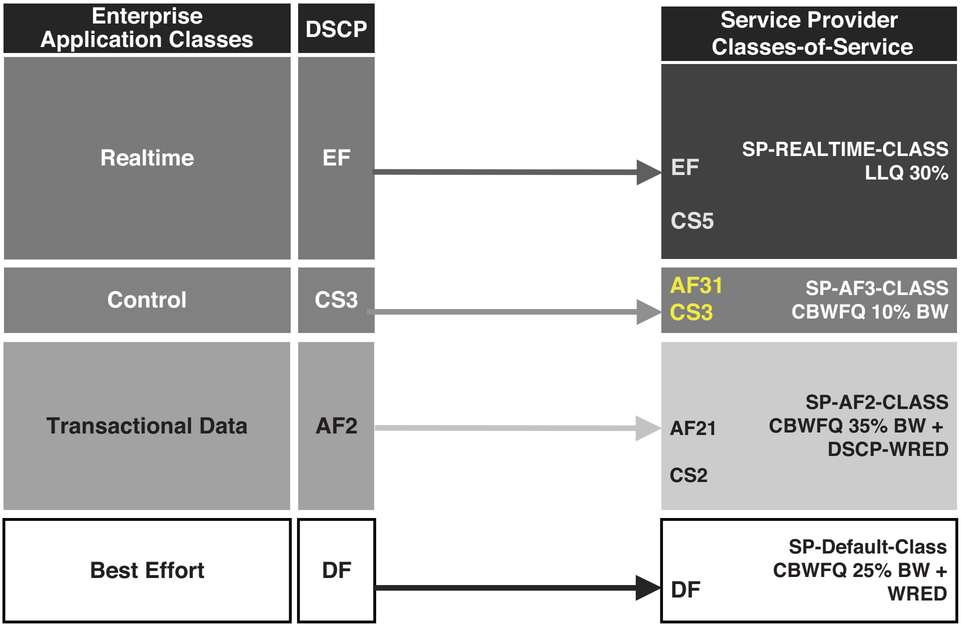

Enterprise Application Classes
DSCP
Service Provider Classes-of-Service
Realtime
EF
EF
CS5
SP-REALTIME-CLASS LLQ 30%
Control
CS3
AF31
CS3
SP-AF3-CLASS CBWFQ 10% BW
Transactional Data
AF2
AF21
CS2
SP-AF2-CLASS CBWFQ 35% BW + DSCP-WRED
Best Effort
DF
DF
SP-Default-Class CBWFQ 25% BW + WRED

Figure 32-2 *Four-Class Enterprise Model Mapped to a Four-Class Service Provider Model*

Example 32-3 *Cisco ISR G2 (CE Router) Four-Class Enterprise Model Mapped to a Four-Class Service Provider Model Configuration Example*

```
 ! The class maps for this policy are identical to the
 ! Four-class egress queuing model in Example 28-5
 ! and therefore will not be repeated here for the sake of brevity


 ! This section defines a four-class enterprise to
 ! Four-class service provider queuing policy map
ISR-G2(config)# policy-map FOUR-CLASS-ENT-TO-FOUR-CLASS-SP
ISR-G2(config-pmap)# class REALTIME
ISR-G2(config-pmap-c)# priority percent 30
 ! Provisions the REALTIME class with 30% LLQ
ISR-G2(config-pmap)# class CONTROL
ISR-G2(config-pmap-c)# bandwidth percent 10
 ! Provisions the CONTROL class with 10% CBWFQ
ISR-G2(config-pmap)# class TRANSACTIONAL-DATA
ISR-G2(config-pmap-c)# bandwidth percent 35
 ! Provisions the TRANSACTIONAL-DATA class with 35% CBWFQ
ISR-G2(config-pmap-c)# fair-queue
 ! Enables a fair-queuing pre-sorter
```

```
ISR-G2(config-pmap-c)# random-detect dscp-based
 ! Enables DSCP-based WRED
ISR-G2(config-pmap)# class class-default
ISR-G2(config-pmap-c)# bandwidth percent 25
 ! Provisions the default/Best Effort class with 25% CBWFQ
ISR-G2(config-pmap-c)# fair-queue
 ! Enables a fair-queuing pre-sorter
ISR-G2(config-pmap-c)# random-detect
 ! Enables WRED


! This section configures the hierarchical shaper
 ! with nested queuing policy
ISR-G2(config-pmap)# policy-map CE-EDGE-SHAPING
ISR-G2(config-pmap)# class class-default
ISR-G2(config-pmap-c)# shape average 40M 800K
 ! Shapes to 40 Mbps with 800 Kbit Bc
ISR-G2(config-pmap-c)# service-policy FOUR-CLASS-ENT-TO-FOUR-CLASS-SP
 ! Provides backpressure on the IOS queuing subsystem for the
 ! four-class queuing policy to engage when the shaping limit is reached


! This section applies the sub-line-rate shaper to the VPN edge int
ISR-G2(config-pmap-c)#interface GigabitEthernet0/2
ISR-G2(config-if)# service-policy output CE-EDGE-SHAPING
 ! Attaches the HQoS shaper to the CE-edge interface
```

You can verify the configuration in Example 32-3 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Eight-Class Enterprise Model Mapped to a Six-CoS Service Provider Model

In this eight-to-six mapping model, two enterprise application classes will need to be merged with two others.

In this case, Network Control and Signaling are combined into a single service provider class of service; because both of these applications classes service control plane protocols, they fit well with each other. In this service provider model (as shown in Figure 32-3), both Network Control and Signaling will need to be re-marked to AF11 and CS1, respectively. Note that simply re-marking an application to a lower code point does not in itself translate to a lower quality of service; this is simply a requirement to gain admission into the intended service provider CoS.

The other application class merging is the Scavenger with Best Effort, which are combined into the service provider default class. Because the whole point of the Scavenger class is to throttle the flows in the case of congestion, a single chokepoint (the CE edge) is usually sufficient to achieve this effect. To achieve this mapping, Scavenger class traffic will need to be re-marked to DF.

Also, as noted in the previous chapter, it may be cost-effective to provision real-time video into a nonpriority service provider CoS. However, even though this class may be treated with a non-real-time service (for example, a CBWFQ) on the PE egress, there's no reason not to service it with strict-priority queuing (LLQ) on CE egress, as is done in this example.

Figure 32-3 shows an eight-class enterprise model mapped to a six-class service provider model.

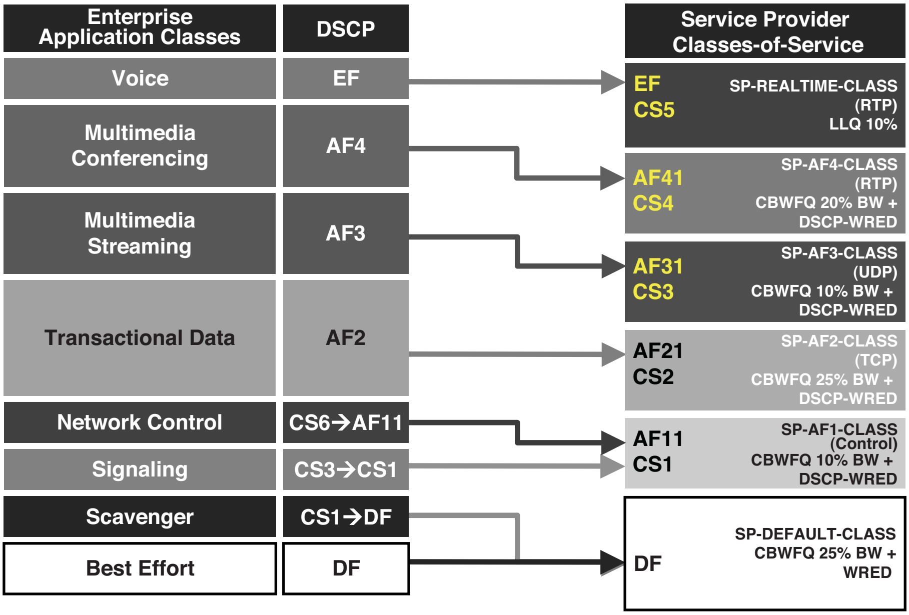

Enterprise Application Classes
DSCP
Service Provider Classes-of-Service
Voice
EF
Multimedia Conferencing
AF4
Multimedia Streaming
AF3
Transactional Data
AF2
Network Control
CS6→AF11
Signaling
CS3→CS1
Scavenger
CS1→DF
Best Effort
DF
EF
CS5
SP-REALTIME-CLASS (RTP) LLQ 10%
AF41
CS4
SP-AF4-CLASS (RTP) CBWFQ 20% BW + DSCP-WRED
AF31
CS3
SP-AF3-CLASS (UDP) CBWFQ 10% BW + DSCP-WRED
AF21
CS2
SP-AF2-CLASS (TCP) CBWFQ 25% BW + DSCP-WRED
AF11
CS1
SP-AF1-CLASS (Control) CBWFQ 10% BW + DSCP-WRED
DF
SP-DEFAULT-CLASS CBWFQ 25% BW + WRED

Figure 32-3 *Eight-Class Enterprise Model Mapped to Six-Class Service Provider Model*

Example 32-4 shows the CE router configuration corresponding to Figure 32-3. To round out the example, this queuing policy is applied to the hierarchical (sub-line-rate) shaping policy presented in Example 32-2.

Example 32-4 *Cisco ISR G2 (CE Router) Eight-Class Enterprise Model Mapped to a Six-Class Service Provider Model Configuration Example*

```
 ! The class maps for this policy are identical to the
 ! eight-class egress queuing model in Example 28-8
 ! and therefore will not be repeated here for the sake of brevity


 ! This section defines an eight-class enterprise to
 ! Six-class service provider queuing policy map
ISR-G2(config-cmap)# policy-map EIGHT-CLASS-ENT-TO-SIX-CLASS-SP
ISR-G2(config-pmap)#  class VOICE
ISR-G2(config-pmap-c)# priority percent 10
 ! Provisions the VOICE class with 10% LLQ
ISR-G2(config-pmap-c)# class MULTIMEDIA-CONFERENCING
ISR-G2(config-pmap-c)#  priority percent 20
 ! Provisions the MULTIMEDIA-CONFERENCING class with 20% LLQ
ISR-G2(config-pmap-c)# class MULTIMEDIA-STREAMING
ISR-G2(config-pmap-c)#  bandwidth percent 10
 ! Provisions the MULTIMEDIA-STREAMING class with 10% CBWFQ
ISR-G2(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ISR-G2(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ISR-G2(config-pmap-c)# class TRANSACTIONAL-DATA
ISR-G2(config-pmap-c)#  bandwidth percent 25
 ! Provisions the TRANSACTIONAL-DATA class with 25% CBWFQ
ISR-G2(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ISR-G2(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ISR-G2(config-pmap-c)# class NETWORK-CONTROL
ISR-G2(config-pmap-c)#  bandwidth percent 5
 ! Provisions the NETWORK-CONTROL class with 5% CBWFQ
ISR-G2(config-pmap-c)#  set dscp af11
 ! Remarks NETWORK-CONTROL to AF11 to map into SP Model
ISR-G2(config-pmap-c)# class SIGNALING
ISR-G2(config-pmap-c)#  bandwidth percent 5
 ! Provisions the SIGNALING class with 5% CBWFQ
ISR-G2(config-pmap-c)#  set dscp cs1
 ! Remarks SIGNALING to CS1 to map into SP Model
ISR-G2(config-pmap-c)# class SCAVENGER
ISR-G2(config-pmap-c)#  bandwidth percent 1
 ! Constrains the SCAVENGER class to 1% CBWFQ
```

```
ISR-G2(config-pmap-c)#  set dscp default
 ! Remarks SCAVENGER to DF to map into SP Model
ISR-G2(config-pmap-c)# class class-default
ISR-G2(config-pmap-c)#  bandwidth percent 24
 ! Provisions the default/Best-Effort class with 24% CBWFQ
ISR-G2(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ISR-G2(config-pmap-c)#  random-detect
 ! Enables WRED


 ! This section configures the hierarchical shaper
 ! with nested queuing policy
ISR-G2(config-pmap)# policy-map CE-EDGE-SHAPING
ISR-G2(config-pmap)# class class-default
ISR-G2(config-pmap-c)# shape average 40M 400K
ISR-G2(config-pmap-c)# service-policy EIGHT-CLASS-ENT-TO-SIX-CLASS-SP
 ! Provides backpressure on the IOS queuing subsystem for the
 ! EIGHT-CLASS-ENT-TO-SIX-CLASS-SP queuing policy to engage
 ! when the shaping limit is reached


 ! This section applies the sub-line-rate shaper to the VPN edge int
ISR-G2(config)#interface GigabitEthernet0/2
ISR-G2(config-if)# service-policy output CE-EDGE-SHAPING
 ! Attaches the HQoS shaper to the CE edge interface
```

Note As has been mentioned, some application classes may have lost their original DSCP values (in this case, Network Control, Signaling, and Scavenger). These may be easily restored (if required) by having the receiving CE router perform deep packet inspection on the VPN-edge link on ingress.

You can verify the configuration in Example 32-4 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Twelve-Class Enterprise Model Mapped to an Eight Class-of-Service Service Provider Model

In this particular 12-to-8 mapping model, two sets of three enterprise application classes will be merged into two service provider models.

First, the Broadcast Video, Real-Time Interactive, and Multimedia Conferencing application classes will be combined into a single (non-real-time) service provider CoS. All three of these applications classes are RTP/UDP-based and therefore can combine relatively effectively together. The main reason for collapsing these classes together would be to reduce the amount of expensive real-time service provider bandwidth that would be required to be purchased, as compared with the alternative design of having broadcast video and real-time interactive video traffic serviced in the SP real-time class, along with voice. Nonetheless, voice, broadcast video, and real-time interactive can all be serviced in LLQs on the CE edge. To achieve this mapping, broadcast video will need to be re-marked to CS4 in this example.

Second, the Network Control, Signaling, and OAM application classes are all combined into a service provider CoS. Because these application classes are primarily control plane protocols, these can peacefully coexist with relative ease. To achieve this mapping, both signaling and OAM will need to be re-marked to CS6 in this example.

Figure 32-4 shows this 12-class enterprise model mapped to an 8-class service provider model.

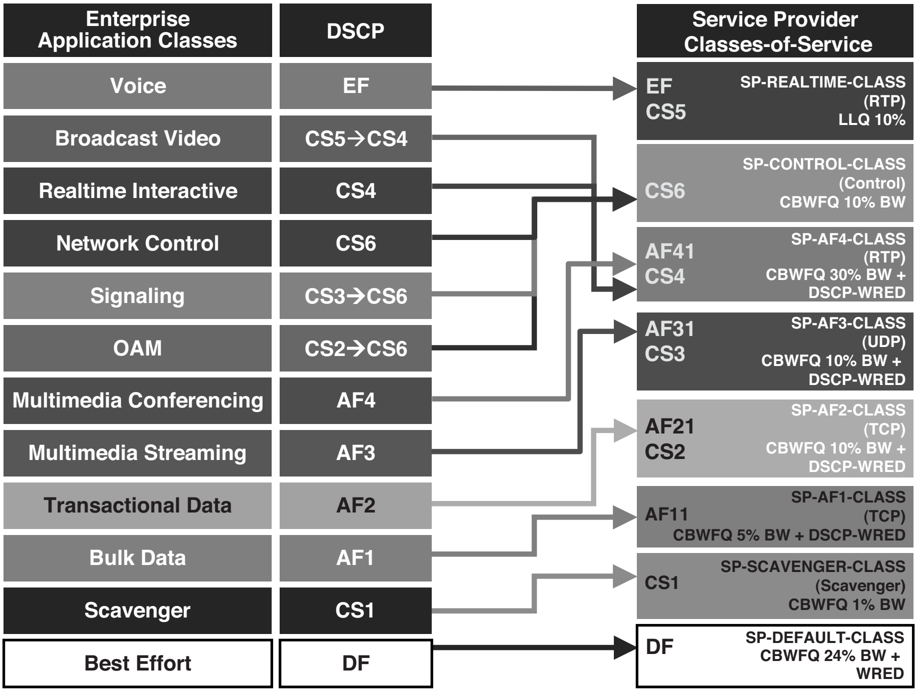

Enterprise Application Classes
DSCP
Service Provider Classes-of-Service
Voice
EF
Broadcast Video
CS5→CS4
Realtime Interactive
CS4
Network Control
CS6
Signaling
CS3→CS6
OAM
CS2→CS6
Multimedia Conferencing
AF4
Multimedia Streaming
AF3
Transactional Data
AF2
Bulk Data
AF1
Scavenger
CS1
Best Effort
DF
EF CS5
SP-REALTIME-CLASS (RTP) LLQ 10%
CS6
SP-CONTROL-CLASS (Control) CBWFQ 10% BW
AF41 CS4
SP-AF4-CLASS (RTP) CBWFQ 30% BW + DSCP-WRED
AF31 CS3
SP-AF3-CLASS (UDP) CBWFQ 10% BW + DSCP-WRED
AF21 CS2
SP-AF2-CLASS (TCP) CBWFQ 10% BW + DSCP-WRED
AF11
SP-AF1-CLASS (TCP) CBWFQ 5% BW + DSCP-WRED
CS1
SP-SCAVENGER-CLASS (Scavenger) CBWFQ 1% BW
DF
SP-DEFAULT-CLASS CBWFQ 24% BW + WRED

Figure 32-4 *Twelve-Class Enterprise Model Mapped to an Eight-Class Service Provider Model*

Example 32-5 shows the CE router configuration corresponding to Figure 32-4. To round out the examples, this queuing policy will be applied directly to a GE interface, which is connecting to the MPLS VPN SP at full line rate (1 Gbps).

Note The platform in this example is an ASR 1000 so that it can perform at line-rate GE speeds.

Example 32-5 *Cisco ASR 1000 (CE Router) 12-Class Enterprise Model Mapped to an 8-CoS Service Provider Model Configuration Example*

```
 ! The class maps for this policy are identical to the
 ! 12-class egress queuing model in example 28-9
 ! and therefore will not be repeated here for the sake of brevity



 ! This section defines a 12-class enterprise to
 ! 8-class service provider queuing policy map
ASR-1000(config-cmap)# policy-map TWELVE-CLASS-ENT-TO-EIGHT-CLASS-SP
ASR-1000(config-pmap)# class VOICE
ASR-1000(config-pmap-c)#  priority percent 10
 ! Provisions the VOICE class with 10% LLQ
ASR-1000(config-pmap-c)# class BROADCAST-VIDEO
ASR-1000(config-pmap-c)#  priority percent 10
 ! Provisions the BROADCAST-VIDEO class with 10% LLQ
ASR-1000(config-pmap-c)#  set dscp cs4
 ! Remarks BROADCAST-VIDEO to CS4 to map into SP Model
ASR-1000(config-pmap-c)# class REALTIME-INTERACTIVE
ASR-1000(config-pmap-c)#  priority percent 10
 ! Provisions the REALTIME-INTERACTIVE class with 10% LLQ
ASR-1000(config-pmap-c)# class NETWORK-CONTROL
ASR-1000(config-pmap-c)#  bandwidth percent 3
 ! Provisions the NETWORK-CONTROL class with 3% CBWFQ
ASR-1000(config-pmap-c)# class SIGNALING
ASR-1000(config-pmap-c)#  bandwidth percent 3
 ! Provisions the SIGNALING class with 3% CBWFQ
ASR-1000(config-pmap-c)#  set dscp cs6
 ! Remarks SIGNALING to CS6 to map into SP Model
ASR-1000(config-pmap-c)# class NETWORK-MANAGEMENT
ASR-1000(config-pmap-c)#  bandwidth percent 4
 ! Provisions the NETWORK-MANAGEMENT class with 4% CBWFQ
ASR-1000(config-pmap-c)#  set dscp cs6
 ! Remarks NETWORK-MANAGEMENT to CS6 to map into SP Model
ASR-1000(config-pmap-c)# class MULTIMEDIA-CONFERENCING
```

```
ASR-1000(config-pmap-c)#  bandwidth percent 10
 ! Provisions the MULTIMEDIA-CONFERENCING class with 10% CBWFQ
ASR-1000(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ASR-1000(config-pmap-c)#  queue-limit 400
 ! Sets the maximum queue depth to 400 packets
ASR-1000(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ASR-1000(config-pmap-c)#  random-detect dscp af41 320 400
 ! Tunes minimum DSCP-WRED threshold for AF41 to 80% queue-depth
ASR-1000(config-pmap-c)#  random-detect dscp af42 280 400
 ! Tunes minimum DSCP-WRED threshold for AF42 to 70% queue-depth
ASR-1000(config-pmap-c)#  random-detect dscp af43 240 400
 ! Tunes minimum DSCP-WRED threshold for AF43 to 60% queue-depth
ASR-1000(config-pmap-c)# class MULTIMEDIA-STREAMING
ASR-1000(config-pmap-c)#  bandwidth percent 10
 ! Provisions the MULTIMEDIA-STREAMING class with 10% CBWFQ
ASR-1000(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ASR-1000(config-pmap-c)#  queue-limit 400
 ! Sets the maximum queue depth to 400 packets
ASR-1000(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ASR-1000(config-pmap-c)#  random-detect dscp af31 320 400
 ! Tunes minimum DSCP-WRED threshold for AF31 to 80% queue-depth
ASR-1000(config-pmap-c)#  random-detect dscp af32 280 400
 ! Tunes minimum DSCP-WRED threshold for AF32 to 70% queue-depth
ASR-1000(config-pmap-c)#  random-detect dscp af33 240 400
 ! Tunes minimum DSCP-WRED threshold for AF33 to 60% queue-depth
ASR-1000(config-pmap-c)# class TRANSACTIONAL-DATA
ASR-1000(config-pmap-c)#  bandwidth percent 10
 ! Provisions the TRANSACTIONAL-DATA class with 10% CBWFQ
ASR-1000(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ASR-1000(config-pmap-c)#  queue-limit 400
 ! Sets the maximum queue depth to 400 packets
ASR-1000(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ASR-1000(config-pmap-c)#  random-detect dscp af21 320 400
 ! Tunes minimum DSCP-WRED threshold for AF21 to 80% queue-depth
ASR-1000(config-pmap-c)#  random-detect dscp af22 280 400
 ! Tunes minimum DSCP-WRED threshold for AF22 to 70% queue-depth
ASR-1000(config-pmap-c)#  random-detect dscp af23 240 400
 ! Tunes minimum DSCP-WRED threshold for AF23 to 60% queue-depth
ASR-1000(config-pmap-c)# class BULK-DATA
```

```
ASR-1000(config-pmap-c)#  bandwidth percent 5
 ! Provisions the BULK-DATA class with 5% CBWFQ
ASR-1000(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ASR-1000(config-pmap-c)#  queue-limit 200
 ! Sets the maximum queue depth to 200 packets
ASR-1000(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
ASR-1000(config-pmap-c)#  random-detect dscp af11 160 200
 ! Tunes minimum DSCP-WRED threshold for AF11 to 80% queue-depth
ASR-1000(config-pmap-c)#  random-detect dscp af12 140 200
 ! Tunes minimum DSCP-WRED threshold for AF12 to 70% queue-depth
ASR-1000(config-pmap-c)#  random-detect dscp af13 120 200
 ! Tunes minimum DSCP-WRED threshold for AF13 to 60% queue-depth
ASR-1000(config-pmap-c)# class SCAVENGER
ASR-1000(config-pmap-c)#  bandwidth percent 1
 ! Constrains the SCAVENGER class to 1% CBWFQ
ASR-1000(config-pmap-c)# class class-default
ASR-1000(config-pmap-c)# bandwidth percent 24
 ! Provisions the default/Best-Effort class with 24% CBWFQ
ASR-1000(config-pmap-c)#  fair-queue
 ! Enables a fair-queuing pre-sorter
ASR-1000(config-pmap-c)#  queue-limit 1000
 ! Sets the maximum queue depth to 1000 packets
ASR-1000(config-pmap-c)#  random-detect dscp-based
 ! Enables DSCP-based WRED
 (config-pmap-c)#  random-detect dscp default 800 1000
 ! Tunes minimum DSCP-WRED threshold for DF to 80% queue-depth


 ! This section binds the queuing policy directly to the
 ! GE interface (connecting to the SP at full 1000-Mbps line rate)
ASR-1000(config)# interface GigabitEthernet 0/0/2
ASR-1000(config-if)# service-policy output TWELVE-CLASS-ENT-TO-EIGHT-CLASS-SP
```

Note The queue limits and WRED thresholds in Example 32-5 have been tuned for an ASR 1000 Gigabit Ethernet interface.

You can verify the configuration in Example 32-5 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map interface**

Summary

This chapter outlined the QoS roles of an enterprise CE router and also identified both the Cisco ASR 1000 and ISR G2 family of routers as being well suited to this role.

Ingress QoS designs were commented on in passing, because these are identical in ingress LAN-edge deep packet inspection policies presented in the branch router design chapter. However, it was pointed out that these same policies can also be effectively deployed on the VPN ingress edge to restore any DSCP markings that may have been lost in transit or as a result of enterprise-to-service provider mapping requirements.

Next, egress QoS design was discussed, beginning with hierarchical shaping (with nested queuing) policies that are required for sub-line-rate Ethernet access scenarios. Two options were presented: the first where the committed burst of the service provider's ingress policer was known (in which case the CE shaper committed burst was set to half of that value), and the second was the case where the SP's Bc was unknown, in which case the CE shaper could be set to a generic recommended best-practice shaping interval of 10 ms.

After this, various enterprise-to-service provider mapping examples were presented in detail, including the following:

- A 4-class enterprise model mapped to a 4-CoS SP model
- An 8-class enterprise model mapped to a 6-CoS SP model
- A 12-class enterprise model mapped to an 8-CoS SP model

Further Reading

Cisco IOS Release 15M&T—QoS Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos/config_library/15-mt/qos-15-mt-library.html

Chapter 33

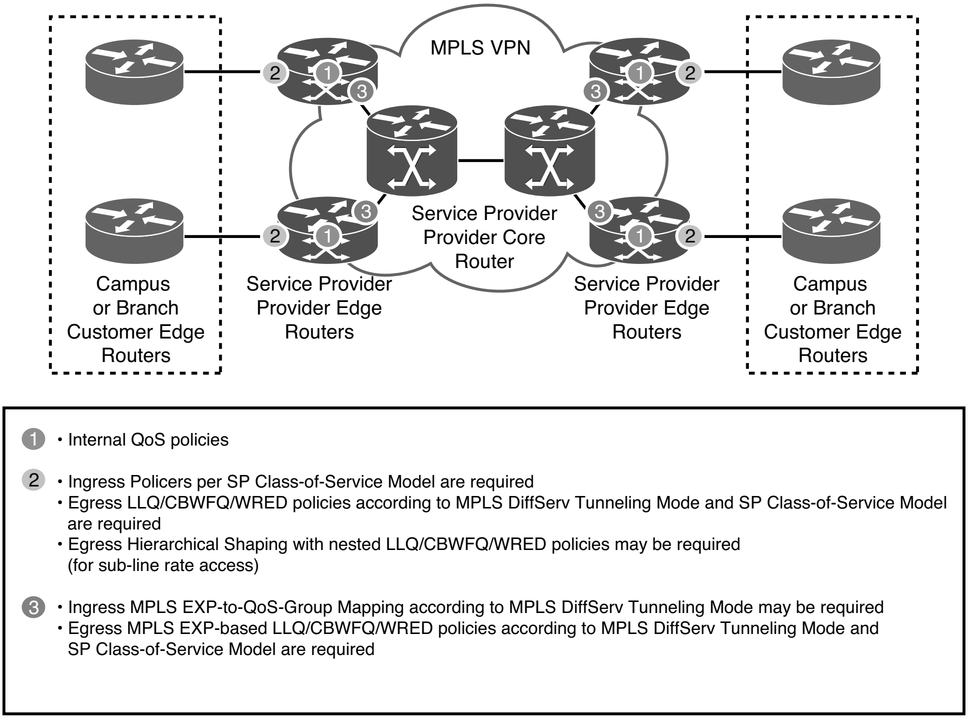
Service Provider Edge (Cisco ASR 9000) QoS Design

The service provider (SP) provider-edge (PE) router has many quality of service (QoS) roles to play, including the following

- Policing traffic by class of service (CoS) on ingress, according to the SP's CoS models
- Implementing Multiprotocol Label Switching (MPLS) differentiated services (DiffServ) tunneling mode QoS policies
- Queuing by MPLS EXP toward the MPLS virtual private network (VPN) core, according to the SP's CoS models
- Queuing toward the customer-edge routers, according to both the SP's CoS models and the MPLS DiffServ tunneling mode
- Shaping traffic toward customer-edge (CE) routers to contracted sub-line service rates (and performing hierarchical queuing and dropping within these shaped rates)

Figure 33-1 illustrates the interface-specific QoS roles of SP PE routers.

The Cisco ASR 9000 series routers are well suited to the role of SP PE routers and therefore are featured in this design chapter. To begin, let's overview this platform's QoS architecture.



1
- Internal QoS policies

2
- Ingress Policers per SP Class-of-Service Model are required
- Egress LLQ/CBWFQ/WRED policies according to MPLS DiffServ Tunneling Mode and SP Class-of-Service Model are required
- Egress Hierarchical Shaping with nested LLQ/CBWFQ/WRED policies may be required (for sub-line rate access)

3
- Ingress MPLS EXP-to-QoS-Group Mapping according to MPLS DiffServ Tunneling Mode may be required
- Egress MPLS EXP-based LLQ/CBWFQ/WRED policies according to MPLS DiffServ Tunneling Mode and SP Class-of-Service Model are required

Figure 33-1 *Service Provider PE Router Interface QoS Roles*

QoS Architecture

The Cisco Aggregation Services Router (ASR) 9000 series router is a distributed, redundant, and modular platform using next-generation hardware and IOS XR software.

On the hardware side, the main elements of an ASR 9000 system include the following:

- Route switch processors (RSPs), which are usually deployed in a redundant manner and perform control plane and management functions
- Network processors (NPs), which are the main forwarding application-specific integrated circuits (ASICs) for the data plane, providing Layer 2 and Layer 3 forwarding features (including QoS) and which are distributed along the linecards
- Fabric interface ASIC (FIA), which provides nonblocking data connections from the linecards to the switch fabric and manage internal system virtual output queues (VOQs)
- Pluggable physical interfaces (PHY), including Gigabit Ethernet (GE), 10GE, 40GE, and 100GE (in addition to WAN interfaces)

Figure 33-2 illustrates the interrelationship of these primary hardware components of the ASR 9000 architecture, highlighting a 24 × 10GE linecard example.

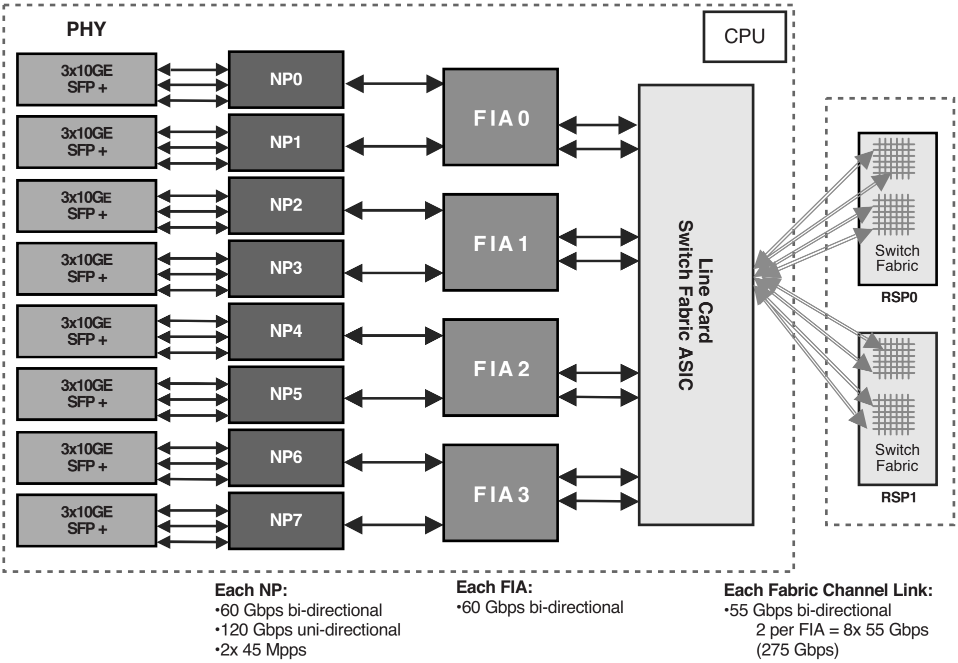


Figure 33-2 *Cisco ASR 9000 Hardware Architecture (A9K-24 × 10G Linecard Example)*

An important point to note in the ASR 9000 hardware architecture is that it is completely nonblocking, with no bottlenecks within the system.

For instance, consider the example shown in Figure 33-2 (featuring a 24 × 10GE linecard). The 10GE interfaces on this linecard are grouped into sets of three (for a total of 30 Gbps of potential throughput per group), with each pair of groups assigned a dedicated network processor. Each NP can provide 60 Gbps of bidirectional throughput. In turn, each pair of NPs is assigned a dedicated an FIA. Respectively, each FIA can support 60 Gbps of bidirectional throughput. And finally, each FIA is assigned a pair of fabric channel links (which support 55 Gbps of bidirectional throughput each) to connect to the switching fabrics and the RSPs. Therefore, of a total of (24 × 10GE) 240 Gbps of potential input traffic, the ASR 9000 architecture provisions (8 × 55 Gbps) 275 Gbps of switch fabric access, with no bottleneck at any point. Therefore, there is no need for ingress queuing to protect traffic from being dropped before reaching the switching fabric.

That said, regardless of how the internal architecture is designed, a router may be configured in the network in such a way as to aggregate traffic (after all, the ASR 9000 is engineered as an ASR) and potentially oversubscribe uplink interfaces to the SP core. In such a case, the ASR 9000 series has an effective and granular way with dealing with congestion, leveraging the use of ingress and egress queues in combination with VOQs, as shown in Figure 33-3.

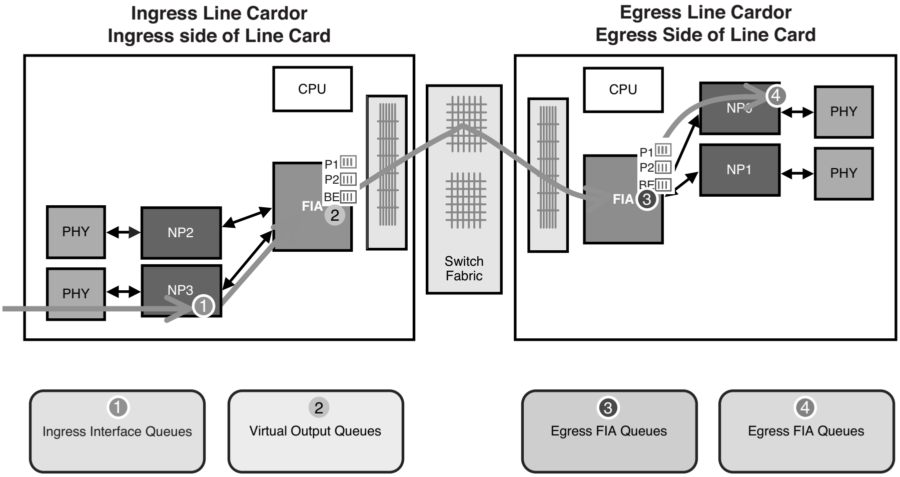


Figure 33-3 *ASR 9000 Queuing Subsystems*

The four queuing subsystems shown in Figure 33-3 are as follows:

- **Ingress interface queues:** These queuing policies are configured in Modular QoS command-line interface (MQC) and applied to an interface/subinterface via the **service-policy input** command. Four levels of hierarchy are supported, in addition to three implicit levels of priority, which are discussed in the next bullet (VOQ).

- **Virtual output queues** (VOQ): These queuing policies are implicitly configured within the MQC ingress or egress queuing policies and feature three implicit levels of priority: priority level 1, priority level 2, and normal (best effort) priority. Four VOQs are supported per each virtual port, with up to 4000 VOQs supported per FIA. The reason these are called *virtual* output queues is that there is no actual congestion being experienced at this node, but rather the egress FIA is signaling congestion to the switch fabric arbiter (on the RSP), which in turn forces the ingress FIA to begin buffering packets in its VOQs.

- **Egress FIA queues:** These queues operate in the same manner as the VOQs in that they support the same three implicit levels of hierarchy. However, rather than being called virtual queues, these are sometimes referred to as *real* queues because they are reflecting an actual congestion event by one of the NPs that they are controlling (as opposed to a virtual congestion event which is signaled to the ingress FIA by the RSP switch arbiters). Four egress FIA queues are supported per each virtual port.

- **Egress interface queues:** These queuing policies are configured in MQC and applied to an interface/subinterface via the **service-policy output** command. Four levels of hierarchy are supported, in addition to three implicit levels of priority.

To understand the relationships of these queuing subsystems, it is probably best to work backward. If the egress interface experiences congestion, the egress NP begins buffering packets in the egress interface queue. It then signals its FIA of the congestion, which in turn also begins buffering packets in its egress FIA queues. Correspondingly, the egress FIA signals the (active) RSP of the congestion. The RSP includes an arbiter that manages access to the switch fabric. The arbiter then denies the ingress FIA access to the switch fabric, thus creating a virtual congestion event. In response, the ingress FIA begins to buffer packets in VOQs. Finally, the ingress FIA correspondingly provides the backpressure to the ingress NP to engage its ingress interface queuing policies.

For the three-level internal system queues, priority level 1 packets will be exclusively and exhaustively serviced before priority level 2. Furthermore, priority level 2 can have its servicing interrupted by the arrival of a priority level 1 packet. In turn, normal traffic is serviced only when both priority queues have been fully serviced and may have their servicing interrupted at any time by the arrival of a priority level 1 or level 2 packet.

The three system priority levels on the ASR 9000 are configured within the ingress/egress MQC policies via the **priority level** command (or lack thereof). By default, if no level is explicitly specified, priority level 1 is assumed. Example 33-1 shows the configuration of a three-level queuing policy that can be applied to ingress or egress interfaces, and which will also implicitly control the VOQs and egress FIA queues.

Example 33-1 *Cisco ASR 9000 Three-Level Interface and VOQ/Egress FIA Queuing Policy*

```
RP/0/RSP0/CPU0:ASR9K(config-cmap)# policy-map THREE-LEVEL-QUEUING
RP/0/RSP0/CPU0:ASR9K(config-pmap)#  class PRIORITY-LEVEL-1
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  priority level 1
 ! Configures class to be serviced with priority level 1
RP/0/RSP0/CPU0:ASR9K(config-pmap)#  class PRIORITY-LEVEL-2
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  priority level 2
 ! Configures class to be serviced with priority level 2
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 65
 ! All CBWFQ classes will be serviced with priority-level normal
```

You can verify the configuration in Example 33-1 with the following commands:

- **show policy-map interface**
- **show controllers fabric crossbar statistics**
- **show controllers fabric fia stats**
- **show controllers fabric fia drops** [**ingress** | **egress**]

A final note on the ASR 9000 hardware architecture as it relates to QoS: Various linecards are supported in three distinct QoS categories, which relate to the memory options supported and thus the QoS queuing and policing scaling capabilities. These memory options are as follows:

- Extended (E), which has the highest QoS scaling capacities
- Base (B), which supports medium QoS scaling capacities
- Low (L), which supports the lowest QoS scaling capacities

All linecards have the same hardware QoS features and configuration syntax; it is only the number of queues or policies that they support that varies with the amount of memory in the E/B/L linecard designation. The specific details vary by linecard, so it is best to check the respective linecard data sheets.

QoS Design Steps

There are generally four steps to configuring a Cisco ASR 9000 in the role of a SP PE router.

1. Configure ingress policing on the PE-to-CE interface.
2. Configure egress queuing (and hierarchical shaping, if required) on the PE-to-CE interface.
3. Configure ingress MPLS EXP-to-QoS group mapping (for uniform and pipe MPLS DiffServ tunneling modes).
4. Configure egress MPLS EXP-based queuing on PE-to-P interface.

Note No explicit policies are required to configure internal QoS and VoQ because these policies are automatically set by the **priority** and **bandwidth remaining** statements in the interface QoS policies.

These steps and corresponding policies vary corresponding to the MPLS DiffServ tunneling mode in use and therefore are grouped accordingly. In addition, to better illustrate design options, four-class, six-class, and eight-class SP models will be used (as introduced in the previous chapter). Therefore, the policy combinations highlighted in this chapter include the following:

- Uniform Mode four-class SP model
- Pipe Mode six-class SP model
- Short Pipe Mode eight-class SP model

MPLS DiffServ Tunneling Models

As overviewed in Chapter 31, “MPLS VPN QoS Considerations and Recommendations,” there are three distinct MPLS DiffServ tunneling modes:

- **Uniform Mode:** Packets may be re-marked at Layer 2 or Layer 3, because it is assumed that the provider and the customer are sharing a single DiffServ domain.
- **Pipe Mode:** Packets may be re-marked by the provider only at Layer 2, and the final egress queuing policies are based on the *provider's (L2)* markings.
- **Short Pipe Mode:** Packets may be re-marked by the provider only at Layer 2, and the final egress queuing policies are based on the *customer's (L3)* markings

Each of these MPLS DiffServ tunneling modes is presented in the following configuration examples.

Uniform Mode MPLS DiffServ Tunneling

Uniform Mode MPLS DiffServ tunneling requires the following policies to be configured:

- Ingress policing on the PE-to-CE edge
- Egress queuing based on MPLS EXP on the PE-to-P edge
- Ingress MPLS EXP-to-QoS group mapping on the PE-to-P edge
- Egress queuing based on QoS group on the PE-to-CE edge

Note The reason that the policies are listed in this order is that it is the logical order of operations as a packet transits an MPLS VPN.

In this Uniform Mode example, the same four-class SP model introduced in the previous chapter (in Figure 32-2) is used. Figure 33-4 shows additional SP-specific details of this four-class SP model.

Note As shown in Figure 33-4, DSCP values are mapped to MPLS EXP bits, which are in turn mapped (at times) to QoS group (QG) values. QGs are internal marking values associated with packets that are used for QoS policy processing. QGs can be used as internal marking placeholders where MPLS EXP marking values can be copied prior to MPLS labels being popped and discarded—thus preserving these EXP values. Additional details about QGs are discussed later in this chapter. In these examples—for policy simplicity and consistency—the QG values are set to match the MPLS EXP values. (For example, MPLS EXP 5 is mapped to QG 5, MPLS EXP 3 is mapped to QG 3, and so on.)

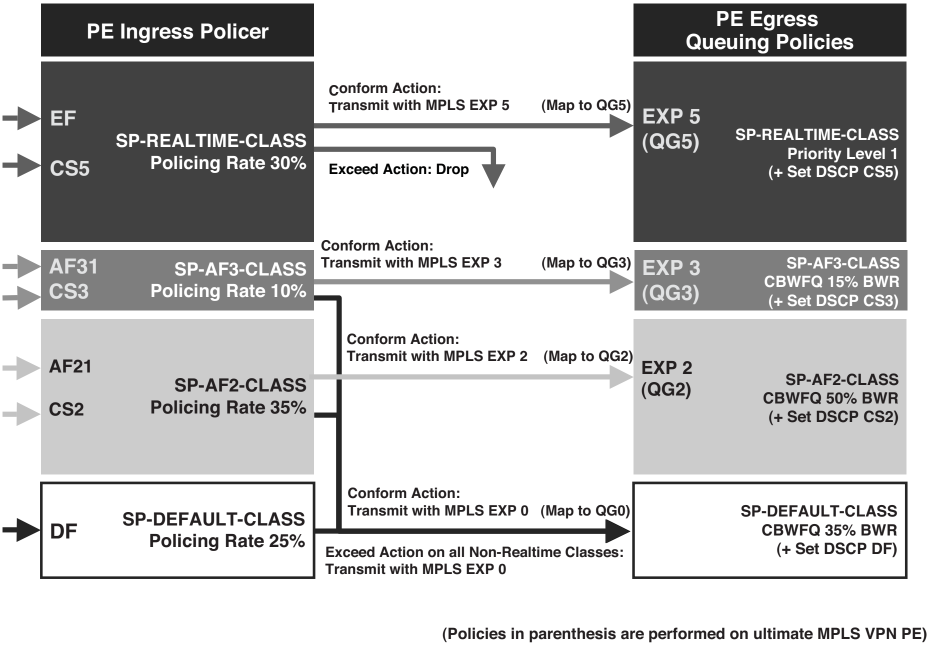

PE Ingress Policer
PE Egress Queuing Policies
EF
CS5
SP-REALTIME-CLASS Policing Rate 30%
Conform Action: Transmit with MPLS EXP 5
(Map to QG5)
Exceed Action: Drop
EXP 5 (QG5)
SP-REALTIME-CLASS Priority Level 1 (+ Set DSCP CS5)
AF31
CS3
SP-AF3-CLASS Policing Rate 10%
Conform Action: Transmit with MPLS EXP 3
(Map to QG3)
EXP 3 (QG3)
SP-AF3-CLASS CBWFQ 15% BWR (+ Set DSCP CS3)
AF21
CS2
SP-AF2-CLASS Policing Rate 35%
Conform Action: Transmit with MPLS EXP 2
(Map to QG2)
EXP 2 (QG2)
SP-AF2-CLASS CBWFQ 50% BWR (+ Set DSCP CS2)
DF
SP-DEFAULT-CLASS Policing Rate 25%
Conform Action: Transmit with MPLS EXP 0
(Map to QG0)
Exceed Action on all Non-Realtime Classes: Transmit with MPLS EXP 0
SP-DEFAULT-CLASS CBWFQ 35% BWR (+ Set DSCP DF)
(Policies in parenthesis are performed on ultimate MPLS VPN PE)

Figure 33-4 *Uniform Mode Four-Class SP Model*

Uniform Mode Ingress Policer

The first policy in this model is to police customer traffic to the contracted rates. Conforming traffic is transmitted with imposed MPLS labels with EXP bits set to match the first 3 bits of the differentiated services code point (DSCP) values of the traffic classes. Excess SP-REALTIME traffic is dropped. However, excess traffic for non-real-time traffic classes is re-marked to MPLS EXP 0. In the default (Best Effort) class, both conforming and exceeding traffic are imposed with MPLS labels marked to EXP 0; although this might seem pointless from a QoS policy perspective, the real purpose of this policer is to enable the metering of excess traffic for accounting and billing purposes.

Example 33-2 shows the configuration of the first part of this four-class Uniform Mode policy.

Example 33-2 *Cisco ASR 9000 Uniform Mode MPLS DiffServ Tunneling: Part 1: Ingress Policer*

```
 ! This section configures the class maps for the ingress policer
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-REALTIME
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs5
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp ef
 ! SP-REALTIME class is matched on DSCP CS5 or EF
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-AF3-CLASS
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs3
```

```
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af31
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af32
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af33
 ! SP-AF3-CLASS is matched on DSCP CS3 or AF3
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-AF2-CLASS
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs2
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af21
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af22
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af23
 ! SP-AF2-CLASS is matched on DSCP CS2 or AF2


 ! This section configures the Uniform-Mode four-class ingress policer
RP/0/RSP0/CPU0:ASR9K(config-cmap)# policy-map SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 30
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 5
 ! Conforming SP-REALTIME traffic (EF/CS5) is sent with MPLS EXP 5
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action drop
 ! Excess SP-REALTIME traffic is dropped
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF3-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  police rate percent 10
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 3
 ! Conforming SP-AF3-CLASS (CS3/AF3) is sent with MPLS EXP 3
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 0
 ! Exceeding SP-AF3-CLASS is sent with MPLS EXP 0
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF2-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  police rate percent 35
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 2
 ! Conforming SP-AF2-CLASS (CS2/AF2) is sent with MPLS EXP 2
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 0
 ! Exceeding SP-AF2-CLASS is sent with MPLS EXP 0
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  police rate percent 25
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 0
 ! Conforming SP-BE-CLASS (DF) is sent with MPLS EXP 0
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 0
 ! Exceeding SP-BE-CLASS (DF) is sent with MPLS EXP 0
 ! This section attaches the ingress policer to the PE-to-CE interface
RP/0/RSP0/CPU0:ASR9K(config)# interface GigabitEthernet 0/2/0/0
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy input SP-FOUR-CLASS-INGRESS-MODEL-
UNIFORM
```

Note Note that IOS XR requires class maps and service policy to be explicitly committed (either by exiting configuration mode or by issuing an **end** command following each class map and policy map). To simplify the configuration examples, these explicit commitments are not shown.

You can verify the configuration in Example 33-2 with the following commands:

- **show class-map reference list** (as shown in Example 33-3)
- **show policy-map pmap** *name* (as shown in Example 33-4)
- **show policy-map interface** (as shown in Example 33-5)
- **show qos interface** *interface w/x/y/z* **input** (as shown in Example 33-6)

Example 33-3 shows the class maps that have been configured in addition to which policies are using them.

Example 33-3 *Verifying Class Maps:* ***show class-map reference list***

```
RP/0/RSP0/CPU0:ASR9K# show class-map reference list
Thu Feb 14 11:33:45.978 PST
1) ClassMap: SP-AF3-CLASS    Type: qos
    Referenced by 1 Policymaps
       PolicyMapName: SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM  Type: qos

2) ClassMap: SP-REALTIME    Type: qos
    Referenced by 1 Policymaps
       PolicyMapName: SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM  Type: qos

3) ClassMap: SP-AF2-CLASS    Type: qos
    Referenced by 1 Policymaps
       PolicyMapName: SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM  Type: qos

RP/0/RSP0/CPU0:ASR9K#
```

Example 33-4 shows the policy map by name and its configuration details.

Example 33-4 *Verifying Policy Maps:* ***show policy-map pmap name***

```
RP/0/RSP0/CPU0:ASR9K# show policy-map pmap-name SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM
Thu Feb 14 11:31:25.433 PST
policy-map SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM
 class SP-REALTIME
  police rate percent 30
   conform-action set mpls experimental imposition 5
   exceed-action drop
  !
 !
 class SP-AF3-CLASS
  police rate percent 10
   conform-action set mpls experimental imposition 3
   exceed-action set mpls experimental imposition 0
  !
 !
 class SP-AF2-CLASS
  police rate percent 35
   conform-action set mpls experimental imposition 2
   exceed-action set mpls experimental imposition 0
  !
 !
 class class-default
  police rate percent 25
   conform-action set mpls experimental imposition 0
   exceed-action set mpls experimental imposition 0
  !
 !
 end-policy-map
!
RP/0/RSP0/CPU0:ASR9K#
```

Example 33-5 shows the service policy that has been applied to the GE interface and the per-class counters. Packets are being policed to their contracted rates, and 6148 exceeding packets are being reported in the default class.

Example 33-5 *Verifying Service Policies: show policy-map interface*

```
RP/0/RSP0/CPU0:ASR9K# show policy-map interface GigE 0/2/0/0
Thu Feb 14 11:27:11.255 PST

GigE0/2/0/0 input: SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM

```

```
Class SP-REALTIME
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :             1355287/174832056          82305
    Transmitted         : N/A
    Total Dropped       :                   0/0                    0
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :             1355287/174832056          83205
    Policed(exceed)     :                   0/0                    0
    Policed(violate)    :                   0/0                    0
    Policed and dropped :                   0/0
Class SP-AF3-CLASS
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            91711668/2356990176          6532
    Transmitted         : N/A
    Total Dropped       :                   0/0                    0
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :            91711668/2356990176          6532
    Policed(exceed)     :                   0/0                    0
    Policed(violate)    :                   0/0                    0
    Policed and dropped :                   0/0
Class SP-AF2-CLASS
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           782028416/1091711668736     146177
    Transmitted         : N/A
    Total Dropped       :                   0/0                    0
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :           782028416/1091711668736     146177
    Policed(exceed)     :                   0/0                    0
    Policed(violate)    :                   0/0                    0
    Policed and dropped :                   0/0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :             1594194/102028416            0
    Transmitted         : N/A
    Total Dropped       :                   0/0                    0
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :          2541454509/3802015946        253413
    Policed(exceed)     :                6148/7912476              0
    Policed(violate)    :                   0/0                    0
    Policed and dropped :                   0/0
GigE0/2/0/0 direction output: Service Policy not installed
RP/0/RSP0/CPU0:ASR9K#
```

Example 33-6 shows interface QoS configuration, including policer details.

Example 33-6 *Verifying Interface QoS:* ***show qos interface***

```
RP/0/RSP0/CPU0:ASR9K# show qos interface GigE 0/2/0/0 input
Thu Feb 14 13:03:46.088 PST
Interface: GigE0_2_0_0 input
Bandwidth configured: 1000000 kbps Bandwidth programmed: 1000000 kbps
ANCP user configured: 0 kbps ANCP programed in HW: 0 kbps
Port Shaper programed in HW: 0 kbps
Policy: SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM Total number of classes: 4
----------------------------------------------------------------------
Level: 0 Policy: SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM Class: SP-REALTIME
QueueID: 131074 (Port Default)
Policer Profile: 48 (Single)
Conform: 3000000 kbps (30 percent) Burst: 37500000 bytes (0 Default)
Child Policer Conform: set exp-imp 5
Child Policer Exceed: DROP
Child Policer Violate: DROP
----------------------------------------------------------------------
Level: 0 Policy: SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM Class: SP-AF3-CLASS
QueueID: 131074 (Port Default)
Policer Profile: 49 (Single)
Conform: 1000000 kbps (10 percent) Burst: 12500000 bytes (0 Default)
Child Policer Conform: set exp-imp 3
Child Policer Exceed: set exp-imp 0
----------------------------------------------------------------------
Level: 0 Policy: SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM Class: SP-AF2-CLASS
QueueID: 131074 (Port Default)
Policer Profile: 50 (Single)
Conform: 3500000 kbps (35 percent) Burst: 43750000 bytes (0 Default)
Child Policer Conform: set exp-imp 2
Child Policer Exceed: set exp-imp 0
----------------------------------------------------------------------
Level: 0 Policy: SP-FOUR-CLASS-INGRESS-MODEL-UNIFORM Class: class-default
QueueID: 131074 (Port Default)
Policer Profile: 51 (Single)
Conform: 2500000 kbps (25 percent) Burst: 31250000 bytes (0 Default)
Child Policer Conform: set exp-imp 0
Child Policer Exceed: set exp-imp 0
----------------------------------------------------------------------
RP/0/RSP0/CPU0:ASR9K#
```

Uniform Mode (MPLS EXP-Based) Egress Queuing Policy

The next policy that the packet will hit is the MPLS EXP-based egress queuing policy on the PE-to-P interface, as shown in Example 33-7.

Example 33-7 *Cisco ASR 9000 Uniform Mode MPLS DiffServ Tunneling: Part 2: Four-Class (MPLS EXP-Based) Egress Queuing Policy*

```
 ! This section configures the class maps for the four-class
 ! MPLS EXP-based egress queuing policy
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-REALTIME-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 5
 ! The SP-REALTIME-EXP class is matched by MPLS EXP 5
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF3-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 3
 ! The SP-AF3-CLASS-EXP class is matched by MPLS EXP 3
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF2-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 2
 ! The SP-AF2-CLASS-EXP class is matched by MPLS EXP 2


 ! This section configures the four-class EXP-based queuing policy map
RP/0/RSP0/CPU0:ASR9K(config-pmap)# policy-map SP-FOUR-CLASS-EXP-QUEUING
RP/0/RSP0/CPU0:ASR9K(config-pmap)#  class SP-REALTIME-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  priority
 ! SP-REALTIME-EXP will operate at priority level 1 (default)
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF3-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 15
 ! SP-AF3-CLASS-EXP will receive CBWFQ with 15% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF2-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 50
 ! SP-AF2-CLASS-EXP will receive CBWFQ with 50% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 35
 ! The default class will receive CBWFQ with 35% bandwidth remaining


 ! This section applies the service policy to the PE-to-P interface
RP/0/RSP0/CPU0:ASR9K(config)# interface tenGigE 0/1/0/2
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy output SP-FOUR-CLASS-EXP-QUEUING
```

Note Whereas in previous chapters the same percentage values were used for both **bandwidth** and **bandwidth remaining** statements for the sake of policy simplicity and modularity, in these SP models the **bandwidth remaining** percentages have been explicitly calculated, reflecting the greater accuracy required in SP network designs because of their contractual obligations to customers. Therefore, in these models, it is assumed that the SP-REALTIME class is allocated 30 percent and that the remaining classes are therefore contending for an aggregate remaining bandwidth of 70 percent. For example, this SP-AF3-CLASS is designated to be 10 percent (absolute) bandwidth, but will be provisioned via a **bandwidth remaining percent** statement with 15 percent (10 percent / 70 percent, rounded).

Note Class maps that match respective SP CoS by MPLS EXP marking values have been appended with an -EXP suffix to differentiate these from the class maps that identify these by DSCP values.

Note MPLS EXP-based Random Detect may optionally be enabled on some of these traffic classes; however, for the sake of keeping these (relatively complex) policies as simple as possible—so as to focus on the core policy elements—MPLS EXP-based weighted random early detection (WRED) has not been included in these examples. The only syntactical difference (from enabling DSCP-based WRED, which has been well covered in previous chapters) would be to use the **exp** keyword with the **random-detect** policy map command (specifically, **random-detect exp** *value min-threshold* [*units*] *max-threshold* [*units*]).

You can verify the configuration in Example 33-7 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **output**

Uniform Mode (MPLS EXP-to-QG) Ingress Mapping Policy

Now, as the packet enters the egress PE in the MPLS VPN, its last label is about to be popped. Therefore, the MPLS EXP value will be lost before the packet reaches the final egress interface. Therefore, before it is lost, this MPLS EXP value must be copied to an internal tag called a QoS group (QG) as it enters the PE router from the core. Then the final egress (PE-to-CE) queuing policy can be based on this QG value, which is indirect manner to queue based on the final MPLS EXP value.

Note Because the class maps that match on MPLS EXP are the same as for the PE-to-P egress queuing policy (shown in the previous example), these can be leveraged for the ingress MPLS EXP mapping policy and therefore are not repeated in this example.

Example 33-8 shows the MPLS EXP-to-QG ingress mapping policy.

Example 33-8 *Cisco ASR 9000 Uniform Mode MPLS DiffServ Tunneling: Part 3: MPLS EXP-to-QG Ingress Mapping*

```
 ! This section configures the ingress MPLS EXP-to-QG mapping
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-FOUR-CLASS-EXP-TO-QOS-GROUP
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  set qos-group 5
 ! MPLS EXP 5 is mapped to QoS group 5
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF3-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  set qos-group 3
 ! MPLS EXP 3 is mapped to QoS group 3
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF2-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  set qos-group 2
 ! MPLS EXP 2 is mapped to QoS group 2
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  set qos-group 0
 ! Everything else is mapped to QoS group 0



 ! This section applies the MPLS EXP-to-QG mapping
 ! to the PE-to-P interface in the ingress direction
RP/0/RSP0/CPU0:ASR9K(config)# interface tenGigE 0/1/0/2
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy input SP-FOUR-CLASS-EXP-TO-QOS-GROUP
```

You can verify the configuration in Example 33-8 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **input**

Uniform Mode (QG-Based) Egress Queuing Policy

The final policy that the packet will hit as it exits the MPLS VPN is the egress queuing policy from the PE to the CE. At this point, the MPLS labels have been popped, but their values have been copied to QGs, and so queuing can be performed on these QG values.

Note that at this point in Uniform Mode MPLS DiffServ tunneling that L2 EXP (previously copied to QGs) will be used to replace MPLS DiffServ values, potentially overwriting customer DSCP markings.

This final part of the Uniform Mode MPLS DiffServ tunneling policy set is shown in Example 33-9.

Example 33-9 *Cisco ASR 9000 Uniform Mode MPLS DiffServ Tunneling: Part 4: QG-Based Egress Queuing Policy*

```
 ! This section configures the QG-based class maps
RP/0/RSP0/CPU0:ASR9K(config)# class-map match-all SP-REALTIME-QG
RP/0/RSP0/CPU0:ASR9K(config-cmap)# match qos-group 5
 ! SP-REALTIME-QG traffic is matched against QoS group 5
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF3-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match qos-group 3
 ! SP-AF3-CLASS-QG traffic is matched against QoS group 3
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF2-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match qos-group 2
 ! SP-AF2-CLASS-QG traffic is matched against QoS group 2



 ! This section configures the Uniform Mode four-class QG-based
 ! egress queuing policy map
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-FOUR-CLASS-QG-QUEUING-UNIFORM
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME-QG
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  priority
 ! SP-REALTIME-QG receives priority level 1 service (by default)
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  set dscp cs5
 ! SP-REALTIME-QG class is re-marked to DSCP CS5 (from MPLS EXP 5)
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF3-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 15
 ! SP-AF3-CLASS-QG receives 15% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  set dscp cs3
 ! SP-AF3-CLASS-QG class is re-marked to DSCP CS3 (from MPLS EXP 3)
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF2-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 50
 ! SP-AF2-CLASS-QG receives 50% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  set dscp cs2
 ! SP-AF2-CLASS-QG class is re-marked to DSCP CS2 (from MPLS EXP 2)
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 35
 ! The default class receives 35% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  set dscp default
 ! Default class traffic is re-marked to DSCP DF
 ! This section attaches the policy to the PE-to-CE interface
RP/0/RSP0/CPU0:ASR9K(config)# interface GigE 0/2/0/0
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy output SP-FOUR-CLASS-QG-QUEUING-
UNIFORM
```

Note Class maps that match respective SP CoS by QG values have been appended with a -QG suffix to differentiate these from the class maps that identify these by DSCP or MPLS EXP values.

You can verify the configuration in Example 33-9 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **output**

Pipe Mode MPLS DiffServ Tunneling

Pipe Mode MPLS DiffServ tunneling requires the following policies to be configured:

- Ingress policing on the PE-to-CE edge
- Egress queuing based on MPLS EXP on the PE-to-P edge
- Ingress MPLS EXP-to-QG mapping on the PE-to-P edge
- Egress queuing based on QG on the PE-to-CE edge

Note The reason that the policies are listed in this order is that it is the logical order of operations as a packet transits an MPLS VPN.

A key difference in Pipe Mode versus Uniform Mode is that in Pipe Mode Layer 2 markings do not have to correspond to the first 3 bits of the DSCP (because these are locally significant to the service provider network), nor then are the final Layer 2 markings propagated to DSCP as packets egress the VPN. Therefore, the service is transparent from the customer’s perspective. (That is, whatever DSCP markings the packets have as they enter the MPLS VPN will be the same as they exit.)

In this Pipe Mode example, the same six-class SP model introduced in the previous chapter (in Figure 32-3) is used. Figure 33-5 shows additional SP-specific details of this six-class SP model.

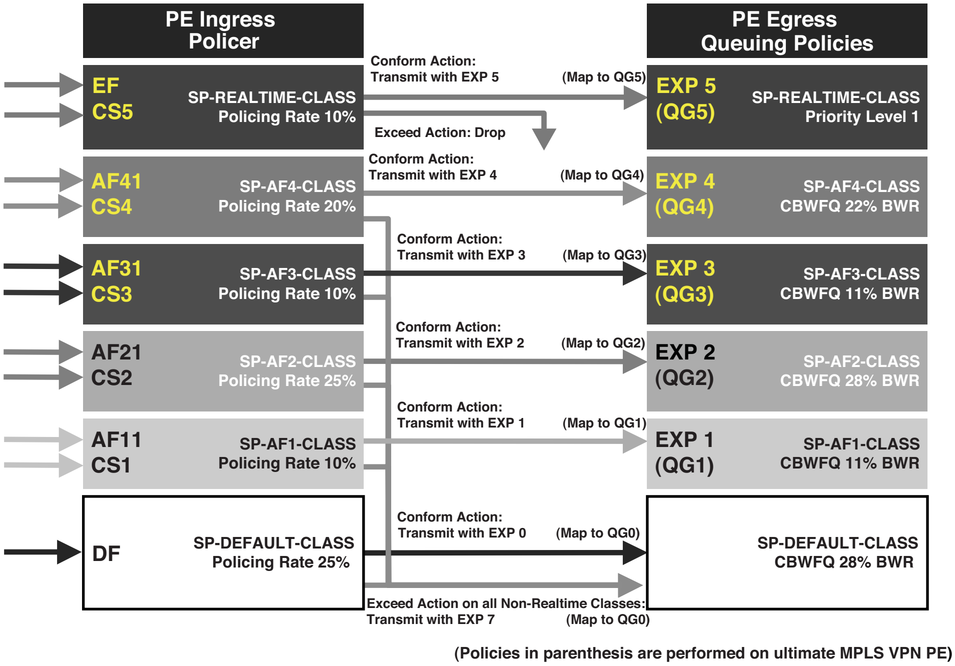


Figure 33-5 *Pipe Mode Six-Class SP Model*

Pipe Mode Ingress Policer

The first policy in this model is to police customer traffic to the contracted rates. Conforming traffic is transmitted with imposed MPLS labels with EXP bits that may or may not match the first 3 bits of the customer DSCP markings; MPLS EXP markings in this model are locally significant only and therefore are entirely at the SP's administrative preference.

In this policing model, excess SP-REALTIME traffic is dropped. However, excess traffic for non-real-time traffic classes is re-marked to MPLS EXP 7 (which in this example is the provider's locally significant marking value to identify out-of-profile flows). From this point forward, anything marked MPLS EXP 7 will be serviced in the default class.

Example 33-10 shows the configuration for the Pipe Mode ingress policer.

Example 33-10 *Cisco ASR 9000 Pipe Mode MPLS DiffServ Tunneling: Part 1: Ingress Policer*

```
 ! This section configures the class maps for the ingress policer
RP/0/RSP0/CPU0:ASR9K(config)# class-map match-any SP-REALTIME
RP/0/RSP0/CPU0:ASR9K(config-cmap)# match dscp cs5
RP/0/RSP0/CPU0:ASR9K(config-cmap)# match dscp ef
 ! SP-REALTIME is matched on DSCP CS5 or EF
```

```
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-AF4-CLASS
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs4
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af41
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af42
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af43
 ! SP-AF4-CLASS is matched on DSCP CS4 or AF4
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-AF3-CLASS
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs3
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af31
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af32
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af33
 ! SP-AF3-CLASS is matched on DSCP CS3 or AF3
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-AF2-CLASS
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs2
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af21
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af22
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af23
 ! SP-AF2-CLASS is matched on DSCP CS2 or AF2
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-AF1-CLASS
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs1
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af11
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af12
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af13
 ! SP-AF1-CLASS is matched on DSCP CS1 or AF1



 ! This section configures the Pipe Mode six-class ingress policer
RP/0/RSP0/CPU0:ASR9K(config-cmap)# policy-map SP-SIX-CLASS-INGRESS-MODEL-PIPE
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 10
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 5
 ! Conforming SP-REALTIME traffic (EF/CS5) is sent with MPLS EXP 5
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action drop
 ! Exceeding SP-REALTIME traffic is dropped
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF4-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 20
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 4
 ! Conforming SP-AF4-CLASS traffic (CS4/AF4) is sent with MPLS EXP 4
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding SP-AF4-CLASS traffic is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF3-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 10
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
```

```
imposition 3
 ! Conforming SP-AF3-CLASS traffic (CS3/AF3) is sent with MPLS EXP 3
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding SP-AF3-CLASS traffic is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF2-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 25
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 2
 ! Conforming SP-AF2-CLASS traffic (CS2/AF2) is sent with MPLS EXP 2
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding SP-AF2-CLASS traffic is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF1-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 10
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 1
 ! Conforming SP-AF1-CLASS traffic (CS1/AF1) is sent with MPLS EXP 1
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding SP-AF1-CLASS traffic is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 25
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 0
 ! Conforming default-class traffic is sent with MPLS EXP 0
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding default-class traffic is sent with MPLS EXP 7



 ! This section attaches the ingress policer to the PE-to-CE interface
RP/0/RSP0/CPU0:ASR9K(config)# interface GigabitEthernet 0/2/0/0
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy input SP-SIX-CLASS-INGRESS-MODEL-
PIPE
```

You can verify the configuration in Example 33-10 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **input**

Pipe Mode (MPLS EXP-Based) Egress Queuing Policy

The next policy that the packet will hit is the MPLS EXP-based egress queuing policy on the PE-to-P interface, as shown in Example 33-11.

Example 33-11 *Cisco ASR 9000 Pipe Mode MPLS DiffServ Tunneling: Part 2: Six-Class (MPLS EXP-Based) Egress Queuing Policy*

```
 ! This section configures the class maps for the six-class
 ! MPLS EXP-based egress queuing policy
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-REALTIME-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 5
 ! SP-REALTIME-EXP is matched by MPLS EXP 5
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF4-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 4
 ! SP-AF4-CLASS-EXP is matched by MPLS EXP 4
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF3-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 3
 ! SP-AF3-CLASS-EXP is matched by MPLS EXP 3
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF2-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 2
 ! SP-AF2-CLASS-EXP is matched by MPLS EXP 2
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF1-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 1
 ! SP-AF1-CLASS-EXP class is matched by MPLS EXP 1


 ! This section configures the six-class EXP-based queuing policy map
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-SIX-CLASS-EXP-QUEUING
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  priority
 ! SP-REALTIME-EXP will operate at priority level 1 (default)
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF4-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 22
 ! SP-AF4-CLASS-EXP will receive CBWFQ with 22% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF3-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 11
 ! SP-AF3-CLASS-EXP will receive CBWFQ with 11% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF2-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 28
 ! SP-AF2-CLASS-EXP will receive CBWFQ with 11% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF1-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 11
 ! SP-AF1-CLASS-EXP will receive CBWFQ with 28% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 28
```

```
 ! The default class will receive CBWFQ with 28% bandwidth remaining


 ! This section applies the service policy to the PE-to-P interface
RP/0/RSP0/CPU0:ASR9K(config)# interface tenGigE 0/1/0/2
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy output SP-SIX-CLASS-EXP-QUEUING
```

Note In this model, it is assumed that the SP-REALTIME class is allocated 10 percent and that therefore the remaining classes are contending for an aggregate remaining bandwidth of 90 percent. For example, this SP-AF4-CLASS is designated to be 20 percent (absolute) bandwidth, but will be provisioned via a **bandwidth remaining percent** statement with 22 percent (20 percent / 90 percent, rounded).

Note Class maps that match respective SP CoS by MPLS EXP marking values have been appended with an -EXP suffix to differentiate these from the class maps that identify these by DSCP values.

You can verify the configuration in Example 33-11 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **output**

Pipe Mode (MPLS EXP-to-QG) Ingress Mapping Policy

As with Uniform Mode MPLS DiffServ tunneling, as the packet enters the egress PE in the MPLS VPN its last label is about to be popped. Therefore, the MPLS EXP value will be lost before the packet reaches the final egress interface. Therefore, before it is lost, this MPLS EXP value must be copied to an internal tag called a QoS group as it enters the PE router from the core. Then the final egress (PE-to-CE) queuing policy can be based on this QG value, which is indirect manner to queue based on the final MPLS EXP value (that is, the SP's locally significant markings).

Note Because the class maps that match on MPLS EXP are the same as for the PE-to-P egress queuing policy (shown in the previous example), these can be leveraged for the ingress MPLS EXP mapping policy and therefore are not repeated in this example.

Example 33-12 shows this MPLS EXP-to-QG ingress mapping policy.

Example 33-12 *Cisco ASR 9000 Pipe Mode MPLS DiffServ Tunneling: Part 3: MPLS EXP-to-QG Ingress Mapping*

```
 ! This section configures the ingress MPLS EXP-to-QG mapping
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-SIX-CLASS-EXP-TO-QOS-GROUP
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# set qos-group 5
 ! MPLS EXP 5 is mapped to QoS group 5
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF4-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# set qos-group 4
 ! MPLS EXP 4 is mapped to QoS group 4
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF3-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# set qos-group 3
 ! MPLS EXP 3 is mapped to QoS group 3
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF2-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# set qos-group 2
 ! MPLS EXP 2 is mapped to QoS group 2
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF1-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# set qos-group 1
 ! MPLS EXP 1 is mapped to QoS group 1
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# set qos-group 0
 ! Everything else is mapped to QoS group 0



 ! This section applies the MPLS EXP-to-QG mapping
 ! to the PE-to-P interface in the ingress direction
RP/0/RSP0/CPU0:ASR9K(config)# interface tenGigE 0/1/0/2
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy input SP-SIX-CLASS-EXP-TO-QOS-GROUP
```

You can verify the configuration in Example 33-12 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **input**

Pipe Mode (QG-Based) Egress Queuing Policy

The final policy that the packet will hit as it exits the MPLS VPN is the egress queuing policy from the PE to the CE, based on the AP's (MPLS EXP) markings. Even though at this point the MPLS labels have been popped, their values have been copied to QGs, and so queuing can be performed based on these QG values.

Note that in Pipe Mode there is no copying of MPLS EXP bits to replace the customer's DSCP markings on exit (thus ensuring DSCP transparency).

Example 33-13 shows this final part of the Pipe Mode MPLS DiffServ tunneling policy set.

Example 33-13 *Cisco ASR 9000 Pipe Mode MPLS DiffServ Tunneling: Part 4: QG-Based Egress Queuing Policy*

```
 ! This section configures the QG-based class maps
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-REALTIME-QG
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match qos-group 5
 ! SP-REALTIME-QG traffic is matched against QoS group 5
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF4-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match qos-group 4
 ! SP-AF4-CLASS-QG traffic is matched against QoS group 4
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF3-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match qos-group 3
 ! SP-AF3-CLASS-QG traffic is matched against QoS group 3
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF2-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match qos-group 2
 ! SP-AF2-CLASS-QG traffic is matched against QoS group 2
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF1-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match qos-group 1
 ! SP-AF1-CLASS-QG traffic is matched against QoS group 1



 ! This section configures the Pipe Mode six-class QG-based
 ! egress queuing policy map
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-SIX-CLASS-QG-QUEUING-PIPE
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME-QG
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  priority
 ! SP-REALTIME-QG receives priority level 1 service (by default)
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF4-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 22
 ! SP-AF4-CLASS-QG receives 22% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF3-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 11
 ! SP-AF3-CLASS-QG receives 11% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF2-CLASS-QG
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 28
 ! SP-AF2-CLASS-QG receives 11% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF1-CLASS-QG
```

```
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 11
 ! SP-AF1-CLASS-QG receives 28% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 28
 ! The default class receives 28% bandwidth remaining



 ! This section attaches the policy to the PE-to-CE interface
RP/0/RSP0/CPU0:ASR9K(config)# interface GigE 0/2/0/0
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy output SP-SIX-CLASS-QG-QUEUING-PIPE
```

Note Class maps that match respective SP CoS by QG values have been appended with a -QG suffix to differentiate these from the class maps that identify these by DSCP or MPLS EXP values.

You can verify the configuration in Example 33-13 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **output**

Short Pipe Mode MPLS DiffServ Tunneling

Short Pipe Mode MPLS DiffServ tunneling requires the following policies to be configured:

- Ingress policing on the PE-to-CE edge
- Egress queuing based on MPLS EXP on the PE-to-P edge
- Egress queuing based on DSCP on the PE-to-CE edge

Note The reason that the policies are listed in this order is that it is the logical order of operations as a packet transits a MPLS VPN.

A key difference in the Short Pipe Mode versus Pipe Mode is that in the final (or ultimate) MPLS VPN PE egress queuing policies (toward the customer CE) are based on the customer's DSCP markings (which have remained unaltered as these have traversed the MPLS VPN). Therefore, there is no need for MPLS EXP-to-QG mapping at the ultimate PE.

In this Short Pipe Mode example, the same eight-class SP model introduced in the previous chapter (in Figure 32-4) is used. Additional SP-specific details of this eight-class SP model are shown in Figure 33-6.

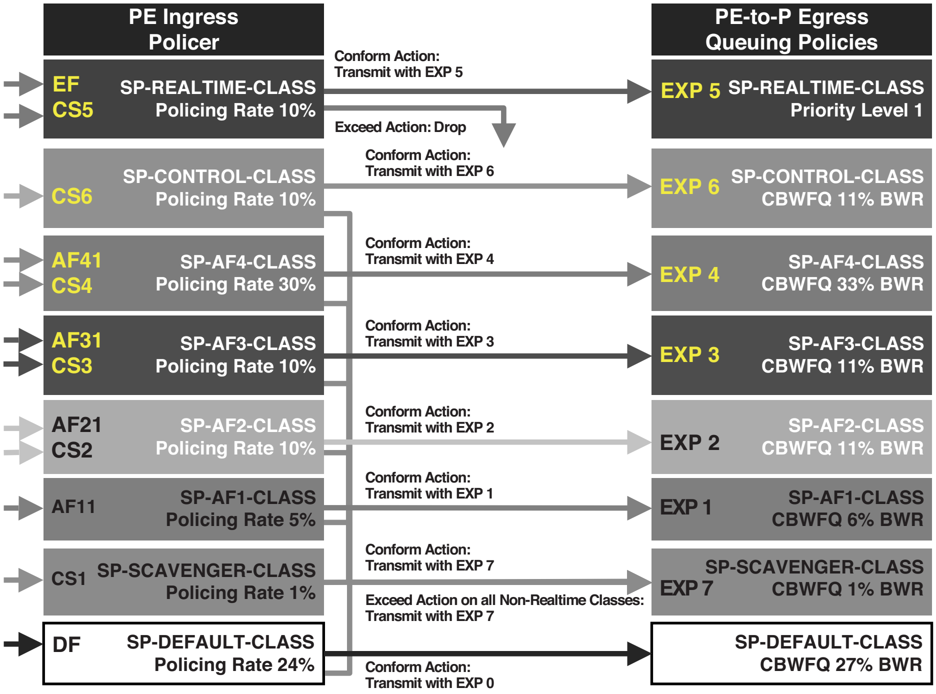

PE Ingress Policer
PE-to-P Egress Queuing Policies
EF
CS5
SP-REALTIME-CLASS Policing Rate 10%
Conform Action: Transmit with EXP 5
Exceed Action: Drop
EXP 5 SP-REALTIME-CLASS Priority Level 1
CS6
SP-CONTROL-CLASS Policing Rate 10%
Conform Action: Transmit with EXP 6
EXP 6 SP-CONTROL-CLASS CBWFQ 11% BWR
AF41
CS4
SP-AF4-CLASS Policing Rate 30%
Conform Action: Transmit with EXP 4
EXP 4 SP-AF4-CLASS CBWFQ 33% BWR
AF31
CS3
SP-AF3-CLASS Policing Rate 10%
Conform Action: Transmit with EXP 3
EXP 3 SP-AF3-CLASS CBWFQ 11% BWR
AF21
CS2
SP-AF2-CLASS Policing Rate 10%
Conform Action: Transmit with EXP 2
EXP 2 SP-AF2-CLASS CBWFQ 11% BWR
AF11
SP-AF1-CLASS Policing Rate 5%
Conform Action: Transmit with EXP 1
EXP 1 SP-AF1-CLASS CBWFQ 6% BWR
CS1
SP-SCAVENGER-CLASS Policing Rate 1%
Conform Action: Transmit with EXP 7
Exceed Action on all Non-Realtime Classes: Transmit with EXP 7
EXP 7 SP-SCAVENGER-CLASS CBWFQ 1% BWR
DF
SP-DEFAULT-CLASS Policing Rate 24%
Conform Action: Transmit with EXP 0
SP-DEFAULT-CLASS CBWFQ 27% BWR

Figure 33-6 *Short Pipe Mode Six-Class SP Model: Part 1: Ingress Policing and PE-to-P Egress Queuing Policies*

Short Pipe Mode Ingress Policer

The first policy in this model is to police customer traffic to the contracted rates. As with Pipe Mode, conforming traffic is transmitted with imposed MPLS labels with EXP bits that may or may not match the first 3 bits of the customer DSCP markings; MPLS EXP markings in this model are locally significant only and therefore are entirely at the SP's administrative preference.

In this policing model, excess SP-REALTIME traffic is dropped. However, excess traffic for non-real-time traffic classes is re-marked to MPLS EXP 7 (which in this example is the provider's locally significant marking value to identify out-of-profile flows). In addition, in this model MPLS EXP 7 is used to identify Scavenger class traffic as well.

Example 33-14 shows the configuration for the Short Pipe Mode ingress policer.

Example 33-14 *Cisco ASR 9000 Short Pipe Mode MPLS DiffServ Tunneling: Part 1: Ingress Policer*

```
 ! This section configures the class maps for the ingress policer
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-REALTIME
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs5
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp ef
 ! SP-REALTIME is matched on DSCP CS5 or EF
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-CONTROL
RP/0/RSP0/CPU0:ASR9K(config-cmap)# match dscp cs6
 ! SP-CONTROL class is matched on DSCP CS6
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-AF4-CLASS
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs4
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af41
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af42
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af43
 ! SP-AF4-CLASS is matched on DSCP CS4 or AF4
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-AF3-CLASS
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs3
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af31
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af32
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af33
 ! SP-AF3-CLASS is matched on DSCP CS3 or AF3
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-AF2-CLASS
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs2
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af21
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af22
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af23
 ! SP-AF2-CLASS is matched on DSCP CS2 or AF2
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-any SP-AF1-CLASS
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af11
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af12
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp af13
 ! SP-AF1-CLASS is matched on DSCP AF1
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-SCAVENGER
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match dscp cs1
 ! SP-SCAVENGER is matched on DSCP CS1


 ! This section configures the Short Pipe Mode
 ! Eight-Class ingress policer
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-EIGHT-CLASS-INGRESS-MODEL-SHORT-PIPE
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 10
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
```

```
imposition 5
 ! Conforming SP-REALTIME traffic (EF/CS5) is sent with MPLS EXP 5
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action drop
 ! Exceeding SP-REALTIME traffic is dropped
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-CONTROL
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 10
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 6
 ! Conforming SP-CONTROL traffic (CS6) is sent with MPLS EXP 6
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding SP-CONTROL traffic is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF4-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 30
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 4
 ! Conforming SP-AF4-CLASS traffic (CS4/AF4) is sent with MPLS EXP 4
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding SP-AF4-CLASS traffic is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF3-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 10
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 3
 ! Conforming SP-AF3-CLASS traffic (CS3/AF3) is sent with MPLS EXP 3
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding SP-AF3-CLASS traffic is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF2-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 10
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 2
 ! Conforming SP-AF2-CLASS traffic (CS2/AF2) is sent with MPLS EXP 2
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding SP-AF2-CLASS traffic is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF1-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 5
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 1
 ! Conforming SP-AF1-CLASS traffic (AF1) is sent with MPLS EXP 1
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding SP-AF1-CLASS traffic is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-SCAVENGER
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 1
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 7
```

```
 ! Conforming SP-SCAVENGER traffic (CS1) is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding SP-SCAVENGER traffic is sent with MPLS EXP 7
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# police rate percent 24
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# conform-action set mpls experimental
imposition 0
 ! Conforming default-class traffic is sent with MPLS EXP 0
RP/0/RSP0/CPU0:ASR9K(config-pmap-c-police)# exceed-action set mpls experimental
imposition 7
 ! Exceeding default-class traffic is sent with MPLS EXP 7



 ! This section attaches the ingress policer to the PE-to-CE interface
RP/0/RSP0/CPU0:ASR9K(config)# interface GigabitEthernet 0/2/0/0
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy input SP-EIGHT-CLASS-INGRESS-MODEL-
SHORT-PIPE
```

You can verify the configuration in Example 33-14 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **input**

Short Pipe Mode (MPLS EXP-Based) Egress Queuing Policy

The next policy that the packet will hit is the MPLS EXP-based egress queuing policy on the PE-to-P interface, as shown in Example 33-15.

Example 33-15 *Cisco ASR 9000 Short Pipe Mode MPLS DiffServ Tunneling: Part 2: Eight-Class (MPLS EXP-Based) Egress Queuing Policy*

```
 ! This section configures the class maps for the eight-class
 ! MPLS EXP-based egress queuing policy
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-REALTIME-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 5
 ! SP-REALTIME-EXP is matched by MPLS EXP 5
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-CONTROL-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 6
 ! SP-CONTROL-EXP is matched by MPLS EXP 6
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF4-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 4
 ! SP-AF4-CLASS-EXP is matched by MPLS EXP 4
```

```
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF3-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 3
 ! SP-AF3-CLASS-EXP is matched by MPLS EXP 3
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF2-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 2
 ! SP-AF2-CLASS-EXP is matched by MPLS EXP 2
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-AF1-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 1
 ! SP-AF1-CLASS-EXP is matched by MPLS EXP 1
RP/0/RSP0/CPU0:ASR9K(config-cmap)# class-map match-all SP-SCAVENGER-EXP
RP/0/RSP0/CPU0:ASR9K(config-cmap)#  match mpls experimental topmost 7
 ! SP-SCAVENGER-EXP is matched by MPLS EXP 7


 ! This section configures the eight-class EXP-based queuing policy map
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-EIGHT-CLASS-EXP-QUEUING
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  priority
 ! SP-REALTIME-EXP will operate at priority level 1 (default)
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-CONTROL-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 11
 ! SP-CONTROL-EXP will receive CBWFQ with 11% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF4-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 33
 ! SP-AF4-CLASS-EXP will receive CBWFQ with 33% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF3-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 11
 ! SP-AF3-CLASS-EXP will receive CBWFQ with 11% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF2-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 11
 ! SP-AF2-CLASS-EXP will receive CBWFQ with 11% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF1-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 6
 ! SP-AF1-CLASS-EXP will receive CBWFQ with 6% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-SCAVENGER-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 1
 ! SP-SCAVENGER-EXP will receive CBWFQ with 1% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 27
 ! The default class will receive CBWFQ with 27% bandwidth remaining


 ! This section applies the service policy to the PE-to-P interface
RP/0/RSP0/CPU0:ASR9K(config)# interface tenGigE 0/1/0/2
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy output SP-EIGHT-CLASS-EXP-QUEUING
```

Note Class maps that match respective SP CoS by MPLS EXP marking values have been appended with an -EXP suffix to differentiate these from the class maps that identify these by DSCP values.

You can verify the configuration in Example 33-15 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **output**

Short Pipe Mode (DSCP-Based) Egress Queuing Policy

The key difference between Short Pipe Mode and Pipe Mode is that the final egress queuing policies (toward the customer CE) are based on the customer's DSCP markings, which have been unaltered in transit over the MPLS VPN.

Figure 33-7 illustrates these final PE-to-CE egress queuing policies.

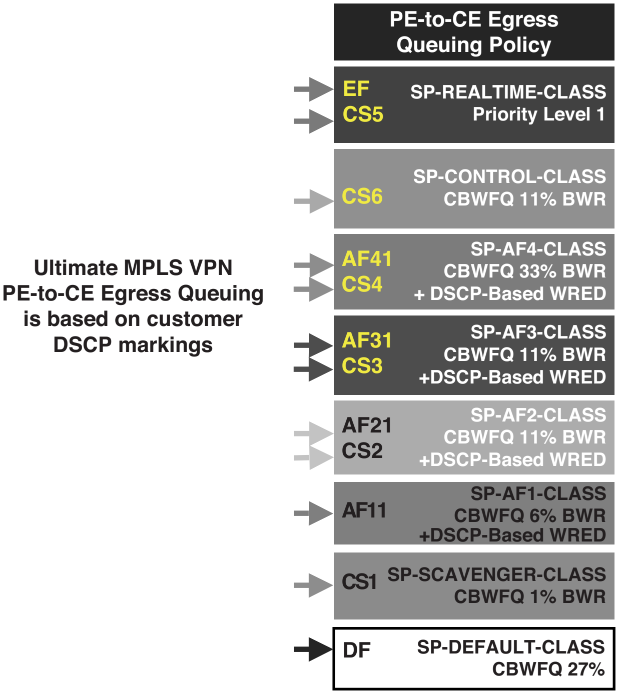


Figure 33-7 *Short Pipe Mode Six-Class SP Model: Part 2: Ultimate PE-to-CE Egress Queuing Policies*

Example 33-16 shows the configuration that corresponds to Figure 33-7.

Note Because the class maps for this policy are based on DSCP values (the same as the class maps for the ingress queuing policy were), these are identical to the class maps shown in Example 33-14 and therefore are not repeated in this example.

Example 33-16 *Cisco ASR 9000 Pipe Mode MPLS DiffServ Tunneling: Part 3: DSCP-Based Egress Queuing Policy*

```
 ! This section configures the Short Pipe Mode
 ! eight-class DSCP-based egress queuing policy map
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-EIGHT-CLASS-QUEUING-SHORT-PIPE
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  priority
 ! SP-REALTIME will operate at priority level 1 (default)
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-CONTROL
 ! SP-CONTROL will receive CBWFQ with 11% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 11
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF4-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 33
 ! SP-AF4-CLASS will receive CBWFQ with 33% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# random-detect dscp-based
 ! Enables DSCP-based WRED on SP-AF4-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF3-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 11
 ! SP-AF3-CLASS will receive CBWFQ with 11% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# random-detect dscp-based
 ! Enables DSCP-based WRED on SP-AF3-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF2-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 11
 ! SP-AF2-CLASS will receive CBWFQ with 11% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# random-detect dscp-based
 ! Enables DSCP-based WRED on SP-AF2-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-AF1-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 6
 ! SP-AF1-CLASS will receive CBWFQ with 6% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# random-detect dscp-based
 ! Enables DSCP-based WRED on SP-AF1-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-SCAVENGER
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 1
 ! SP-SCAVENGER will receive CBWFQ with 1% bandwidth remaining
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# bandwidth remaining percent 27
```

```
 ! The default class will receive CBWFQ with 27% bandwidth remaining



 ! This section applies the service policy to the PE-to-CE interface
RP/0/RSP0/CPU0:ASR9K(config)# interface GigabitEthernet 0/2/0/0
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy output SP-EIGHT-CLASS-QUEUING-SHORT-
PIPE
```

Note Because these final queuing policies are based on the customers' DSCP markings, DSCP-based WRED can be appropriately used at this node; MPLS EXP-based WRED would be used on nodes where (optional) congestion avoidance policies are based on service providers MPLS EXP markings.

You can verify the configuration in Example 33-16 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **output**

Summary

This chapter outlined the QoS roles of an SP PE router and also identified the Cisco ASR 9000 series Aggregation Services Routers as being well suited to this role.

The QoS architecture of the ASR 9000 was overviewed and was shown to have a non-blocking internal architecture. Nonetheless, because the ASR 9000 may be configured to aggregate traffic, it supports extensive internal queuing subsystems, including three levels of internal queuing (priority level 1, priority level 2, and normal) for VOQ and egress FIA queuing, in addition to ingress and egress queuing.

Next, three sets of comprehensive design models were shown that showed how to configure Uniform, Pipe, and Short Pipe Mode MPLS DiffServ tunneling, including the following:

- Ingress policing models
- MPLS EXP-based (PE-to-P) queuing models
- MPLS EXP-to-QoS group mapping models (where required)
- QoS group or DSCP-based (PE-to-P) queuing models

Furthermore, the examples were varied so as to illustrate the following:

- Four-class SP model
- Six-class SP model
- Eight-class SP model

Additional Reading

Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Configuration Guide, Release 4.3.x: http://www.cisco.com/en/US/docs/routers/asr9000/software/asr9k_r4.3/qos/configuration/guide/b_qos_cg43xasr.html

Cisco ASR 9000 Series Aggregation Services Router MPLS Layer 3 VPN Configuration Guide, Release 4.3.x: http://www.cisco.com/en/US/docs/routers/asr9000/software/asr9k_r4.3/lxvpn/configuration/guide/vcasr9kv343.html

This page intentionally left blank

Chapter 34

Service Provider Core (Cisco CRS) QoS Design

The primary quality of service (QoS) role of service provider (SP) core routers is to prevent drops in the event of congestion, whether the congestion be internal or at the interface.

Sometimes service providers take different approaches to dealing with packet drops on their core routers. One such approach is to carefully monitoring link-utilization rates and adding additional links when utilization rates hit 40 percent to 50 percent, thus obviating the need for QoS. Another approach is implementing MPLS Traffic Engineering. Nonetheless, for providers implementing differentiated services (DiffServ) QoS policies, Figure 34-1 illustrates the specific QoS roles required on SP core routers.

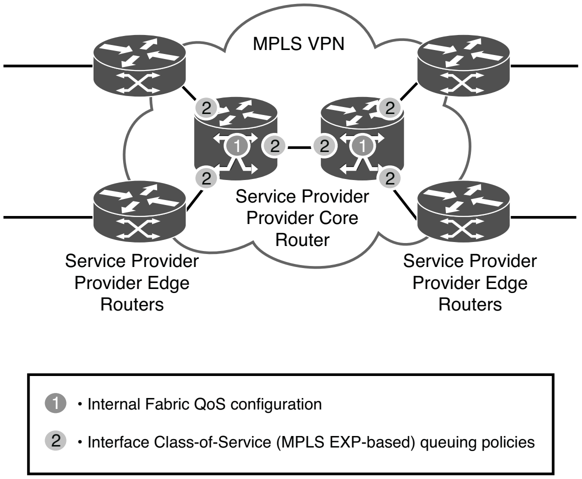

MPLS VPN
2
1
2
2
1
2
2
2
Service Provider Provider Core Router
Service Provider Provider Edge Routers
Service Provider Provider Edge Routers
1 • Internal Fabric QoS configuration
2 • Interface Class-of-Service (MPLS EXP-based) queuing policies

Figure 34-1 *Service Provider Core Router QoS Roles*

The Cisco CRS-3 series routers are custom engineered to serve the role of SP-edge routers. However, before diving into the designs, let's take a look at the hardware architecture of this platform.

QoS Architecture

The Cisco Carrier Services Router (CRS) is a fully modular and distributed routing engineered for SP networks. The CRS-3 represents the current generation of this routing system, which provides 3.5x the capacity of the original CRS-1 system, for a maximum capacity of 322 TB (for a multishelf system).

The CRS system includes multiple internal QoS mechanisms. To best understand why these are required and how they work, it is best to do a brief overview of the CRS hardware. The main elements of a CRS-3 system include the following:

- **Route processors (RPs),** which perform control plane and management functions.
- **Modular services cards (MSCs)** are the modular linecards, which provide hardware-based packet forwarding, policy features, and QoS packet-processing. The CRS-MSC-140G series of MSCs engineered for the CRS-3 connect to the switch fabric at 140 Gbps.
- **Physical Layer Interface Modules (PLIMs)** are physically separated modules that provide optical interfaces and framing hardware functions. A PLIM will have one or more PLIM application-specific interface cards (ASICs) (PLAs), which are used to send and receive Layer 3 packets to and from the MSC they connect to.
- **Midplane** provides direct connections between adjacent PLIMs and MSCs, in addition to direct connections between each MSC and the switching fabric.
- **Switch fabric** provides the main interconnect between MSCs. The CRS features a three-stage cell-based switching fabric that is QoS aware.

Figure 34-2 illustrates the interrelationship of these primary hardware components of the CRS-3 architecture.

There are various (potential) points of oversubscription within the CRS-3 architecture. To identify these points, it is necessary to zoom in one level further, so as to examine the components of the MSC (CRS-MSC-140G) in greater detail, as shown in Figure 34-3. In Figure 34-3, the top half represents the ingress path of packets to the switch fabric, and the bottom half represents the egress path (which may or may not be on the same MSC).

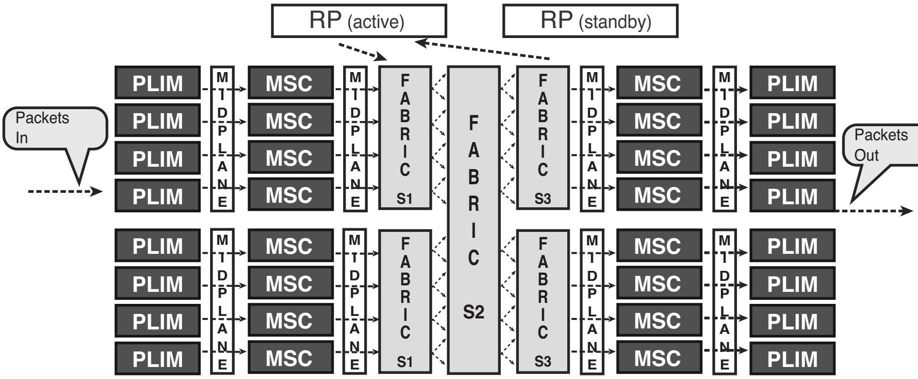


Figure 34-2 *CRS-3 System Hardware Architecture*

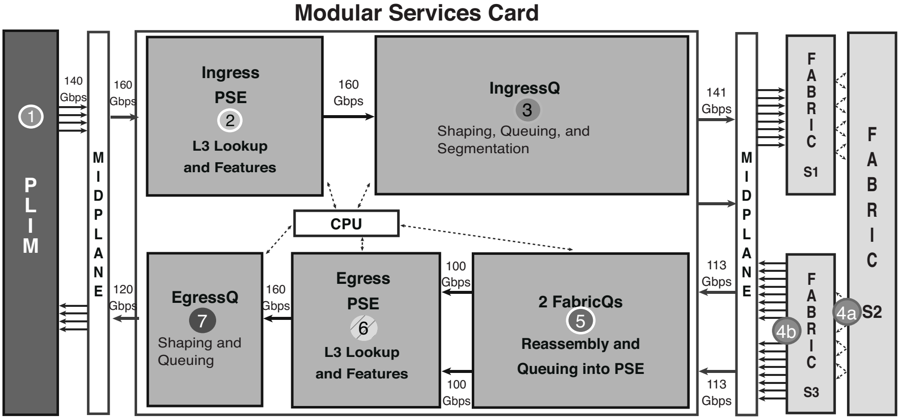


Figure 34-3 *CRS-3 MSC Hardware Architecture*

The MSC implements QoS operations at the following points:

1. **PLIM:** Each PLIM connects to the MSC via a 140-Gbps channel; some PLIMs allow for oversubscription (such as the 20-port 10GE LAN/WAN-PHY interface module, which has a total ingress capacity of 200 Gbps). To manage potential oversubscription, PLIM ASICs implement a two-level (high-priority/low-priority) queuing system.

2. **Ingress packet switching engine (PSE):** The primary L3 feature processing ASIC, applying security and QoS features (such as policing and weight random early detection (WRED); queuing and shaping are handled by the ingress and egress queue ASICs). There is one PSE in the receive path (ingress PSE) and another in the transmit path (egress PSE).

3. **Ingress Queue:** The receive queuing ASIC. Both the ingress and egress queueQ ASICs implement a packet-by-packet modified deficit round-robin (P2MDRR) queuing algorithm, in addition to low-latency/strict-priority queuing and shaping support. Packets enter the ingress queueQ ASIC at 160 Gbps, but leave as 136- byte cells at an effective throughput rate of 141 Gbps. This difference in ingress and egress speeds necessitates the need for queuing at this node.

4. **Switch fabric:** The switch fabric supports two-level (high-priority/low-priority) queuing for unicast traffic—and the same for multicast traffic—at the second and third stages of the fabric (S2 and S3, respectively).

5. **Fabric Queue:** There are two fabric queueASICs in the transmit path (from the fabric to the transmit PLIM). The primary role of the fabric queue ASICs is to reassemble cells back into packets. However, because the cells enter each Fabric queue at 113 Gbps and leave at 100 Gbps, queuing is also required at these nodes. The Fabric queue can be configured to service packets with three levels of service: High Priority (HI), Assured Forwarding (AF), and Best Effort (BE).

6. **Egress PSE:** The transmitting-side complement of the ingress PSE and supports egress policing and WRED.

7. **Egress Queue:** The transmitting-side complement of the ingress queue ASIC and supports P2MDRR, low-latency queuing, and shaping.

As can be noted, at every stage of potential internal congestion, the CRS-3 (with CRS-MSC-140G) supports (at a minimum) high-priority propagation, meaning that priority traffic will receive strict-priority service at every potential congestion node. Additional queuing granularity is available at the fabric queuess (with three levels of service: High Priority, Assured Forwarding, and Best Effort), and finally, the highest levels of queuing granularity are supported at the ingress and egress queue Q ASICs, which can support 64,000 for all interfaces managed.

Interface and internal QoS policies are configured in the following manner:

- A MQC-based *service policy* applied to an interface in the *input* direction will be implemented in the ingress PSE and ingress queue ASICs. Furthermore, any traffic class in this policy map that is configured with a *priority* policy map action will receive strict-priority queuing not only at the ingress queue but also in the PLIM ASICs.

- A MQC-based *service policy* applied to an interface in the *output* direction will be implemented in the egress PSE and egress queue ASICs.

- A MQC-based service policy applied to the *switch fabric* (presented within the IOS XR configuration as a virtual interface) will be implemented in the fabric queue ASICs. Furthermore, any traffic class in this policy map that is configured with a *priority* policy map action will receive strict-priority queuing not only in the fabric queue ASICs but also in the switch fabric (S2 and S3) queues *and* in the ingress queue ASIC fabric queues (the final queue-set in this ASIC that transmit cells to the switch fabric). Therefore, collectively such a policy is referred to as fabric QoS because it controls how packets/cells enter, traverse, and exit the switch fabric.

A key point should be noted relating to the interaction between fabric QoS policies and ingress interface QoS policies: Fabric QoS policies can control (or override) ingress QoS policies at the ingress queue fabric queue ASICs. Specifically, if an ingress interface QoS policy were to classify and mark particular traffic types as being **priority** and a fabric QoS policy were to be applied either marking alternative traffic as being **priority**—or not setting **priority** at all—the ingress interface policy **priority** statement would be effectively ignored at the ingress queue fabric queue (because it is overridden by the fabric QoS policy). In addition, the fabric QoS policy will control the queuing at the S2 ASICs, and the S3 ASICs, and the fabric queues.

Note Any **priority** class configured in the ingress interface policy would still be used for any shaping queues configured on the ingress queue ASIC. However, fabric QoS would control which packets get assigned priority in the ingress queue fabric queue.

For example, if an ingress policy that provisioned MPLS EXP 5 with **priority** was configured on the ingress interface, but no fabric QoS policy was configured, EXP 5 packets would *not* receive high-priority service at the ingress queue fabric queues (because by default only internal control traffic is placed in the high-priority queue in fabric QoS).

Alternatively, if a fabric QoS policy were configured with only MPLS EXP 6 packets assigned **priority** service, then again only these packets would receive strict-priority servicing at the ingress queue fabric queue, S2, S3, and at the fabric queues.

Therefore, it is recommended to ensure that no conflict exists between the ingress QoS policy and the fabric QoS policy.

QoS Design Steps

There are two main steps to configuring QoS on a Cisco CRS-3 router in the role of a provider core router:

1. Configure a fabric QoS policy.
2. Configure interface QoS policies (which may be applied in either/both the ingress and egress direction).

SP Core Class-of-Service QoS Models

Three sets of SP core QoS policies are presented—namely, a corresponding fabric and interface QoS policy for each of the following SP Class-of-Service models:

- Four-class SP model
- Six-class SP model
- Eight-class SP model

These SP core Class-of-Service models, which correspond to the SP models used in the previous chapters in this section, will now be presented.

Four-Class-of-Service SP Model

The four-class SP model is shown in Figure 32-2 and 33-4 in Chapters 32, “Enterprise Customer Edge (Cisco ASR 1000 and ISR G2) QoS Design,” and 33, “Service Provider Edge (Cisco ASR 9000) QoS Design,” respectively. The following sections present fabric QoS and interface QoS policies to correspond to this four-class SP model.

Four-Class-of-Service Fabric QoS Policy

Example 34-1 shows the fabric QoS policy for the four-class SP model. Because there are only three queues available for fabric QoS policies

- MPLS EXP 5 is mapped to the High-Priority fabric Q.
- MPLS EXP 2 and 3 are mapped to the Assured Forwarding fabric Q (which is provisioned at an aggregate bandwidth value to accommodate these combined traffic classes).
- All other EXP values are mapped (by class-default) to the Best Effort fabric Q.

Example 34-1 *Cisco CRS-3 Four-Class SP Model Fabric QoS Policy*

```
 ! This section configures the class maps for four-class fabric QoS
RP/0/RP0/CPU0:CRS-3(config-cmap)# class-map match-any SP-FABQOS-HI
RP/0/RP0/CPU0:CRS-3(config-cmap)#  match mpls experimental topmost 5
 ! Matches MPLS EXP 5 (corresponds to the SP-REALTIME class)
RP/0/RP0/CPU0:CRS-3(config-cmap)# class-map match-any SP-FABQOS-AF
RP/0/RP0/CPU0:CRS-3(config-cmap)#  match mpls experimental topmost 2 3
 ! Matches MPLS EXP 2 and 3 (corresponds to guaranteed BW SP classes)

 ! This section configures the four-class fabric QoS policy map
RP/0/RP0/CPU0:CRS-3(config-pmap)# policy-map SP-FOUR-CLASS-FABQOS
RP/0/RP0/CPU0:CRS-3(config-pmap)#  class SP-FABQOS-HI
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  priority
 ! The SP-FABQOS-HP gets high priority (strict priority)
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-FABQOS-AF
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 65
 ! The SP-FABQOS-AF gets assured forwarding with 65% BWR
```

```
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class class-default
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 35
 ! The default-class gets best effort with 35% BWR


 ! This section applies the fabric QoS policy to the switch fabric
RP/0/RP0/CPU0:CRS-3(config)# switch-fabric
RP/0/RP0/CPU0:CRS-3(config-switch-fabric)# service-policy SP-FOUR-CLASS-FABQOS
 ! Applies the fabric QoS policy to the switch fabric
```

Note The **match-any** operator is used in all the class maps in these examples because the **match-all** operator is not supported in IOS XR for the CRS-3 (as of Release 4.3.x).

Note A fabric QoS policy map class configured with a **priority** statement will be assigned to the High-Priority (HI) Fabric Q queue, a (nondefault) policy map class configured with **bandwidth remaining** will be assigned to the Assured Forwarding (AF) fabric Q queue, and a class-default traffic will be assigned to the Best Effort (BE) fabric Q (and specifying bandwidth on this default class is optional).

Note The **service-policy** statement applied to the **switch-fabric** does not require (or support) a specified direction (such as input or output). This is because the fabric QoS policy actually configures both ingress queuing policies (such as at the ingress Q fabric queues) and egress queuing policies (such as at the switch fabric S2 and S3 stages, and at the fabric Qs).

You can verify the configuration in Example 34-1 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show controllers fabricq queues** (as shown in Example 34-2)
- **show controllers fabricq statistics** (as shown in Example 34-3)

Example 34-2 highlights that there are three fabric Q queues per interface: HI (High-Priority), AF (Assured Forwarding) and BE (Best Effort), in addition to configuration details on how these queues are provisioned. In addition, it shows that there are the same three levels of queuing for multicast traffic, in addition to a single high-priority queue dedicated to CPU control traffic.

Example 34-2 *Verifying Fabric QoS: show controllers fabric queues*

```
RP/0/RP0/CPU0:CRS-3# show controllers fabricq queues loc 0/5/CPU0
Fabric Queue Manager Queue Information:
=======================================
Location: 0/5/CPU0
Asic Instance: 0
Fabric Destination Address: 0
CpuCtrl Cast range : 0 - 7
Multicast Range : 8 - 71
Unicast Quanta in KBytes : 58, Multicast Quanta : 14
+--------------------------------------------------------------------------------
--+
|Type/Ifname            |Port| Queue| Q |P-quanta|Q-quanta|HighW |LowW  | Q Len |
BW |
|                       | num|  num |pri| KBytes | KBytes |KBytes|KBytes| KBytes|
(kbps)|
+--------------------------------------------------------------------------------
--+
|Cpuctrl Cast           |   0| 0 - 7| HI|     13|       13| 1021|    919|     0|
N/A|
|Multicast              |   0|     9| BE|     13|       13| 5118|   4606|     0|
N/A|
|Multicast              |   0|    10| AF|     13|       13| 5118|   4606|     0|
N/A|
|Multicast              |   0|    11| HI|     13|     1905| 5118|   4606|     0|
N/A|
|GigabitEthernet0/5/0/0|   1|  1025| BE|     13|      187| 3661|   3295|     0|
1000000|
|GigabitEthernet0/5/0/0|   2|  2049| AF|     13|      187| 3661|   3295|     0|
1000000|
|GigabitEthernet0/5/0/0|   3|  3073| HI|   1905|      187| 3661|   3295|     0|
1000000|


<snip>
```

Example 34-3 highlights that even though there are 8,715,884,484 packets reassembled from cells in this Fabric Q, there have been 0 packet drops.

Example 34-3 *Verifying Fabric QoS: show controllers fabric statistics*

```
RP/0/RP0/CPU0:CRS-3# show controllers fabricq statistics loc 0/1/CPU0
Fabric Queue Manager Packet Statistics
======================================
Location: 0/1/CPU0
Asic Instance: 0
Fabric Destination Address:          4
```

```
BP Asserted Count :                  1 (+                0 )
MC BP Asserted Count :               0 (+                0 )

Input Cell counters:
+----------------------------------------------------------+
Data cells :                5202891134 (+                4 )
Control cells :             2133618291 (+            34000 )
Idle cells :            31374246993871 (+        499897823 )

Reassembled packet counters
+----------------------------------------------------------+
Ucast pkts :                8715884484 (+                0 )
Mcast pkts :                         0 (+                0 )
Cpuctrlcast pkts :             1422623 (+                2 )

Dropped packets
+----------------------------------------------------------+
Ucast pkts :                         0 (+                0 )
Mcast pkts :                         0 (+                0 )
Cpuctrlcast pkts :                   0 (+                0 )
Vital denied pkts :                  0 (+                0 )
NonVital denied pkts :               0 (+                0 )
Unicast lost pkts :                  0 (+                0 )
Ucast partial pkts :                 0 (+                0 )
PSM OOR Drops :                      0 (+                0 )
```

Four-Class-of-Service Interface QoS Policy

Example 34-4 shows the interface QoS policy for the four-class SP model. Note that unlike other platforms, a traffic class configured with **priority** must include an explicit policer (on a CRS-MSC-140G).

Example 34-4 *Cisco CRS-3 Four-Class SP Model Interface QoS Policy*

```
 ! The class maps for this policy are essentially the same
 ! as in Example 33-7
 ! (with the exception of substituting "match-all" with "match-any")
 ! and so, these will not be repeated here for the sake of brevity


 ! This section configures the four-class interface QoS policy map
RP/0/RP0/CPU0:CRS-3(config-pmap)# policy-map SP-FOUR-CLASS-EXP-QUEUING
RP/0/RP0/CPU0:CRS-3(config-pmap)#  class SP-REALTIME-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# police rate percent 30
```

```
 ! An explicit policer is required on the priority class (CRS-MSC-140G)
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  priority level 1
 ! The SP-REALTIME-EXP class will receive strict-priority servicing
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-AF3-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 15
 ! The SP-AF3-CLASS-EXP class will receive 15% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-AF2-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 50
 ! The SP-AF2-CLASS-EXP class will receive 50% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class class-default
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 35
 ! Class-default will receive 35% bandwidth remaining


 ! This section applies the four-class interface QoS policy map to a
 ! core-to-edge (P-to-PE) interface in both directions
RP/0/RP0/CPU0:CRS-3(config)# interface TenGigE 0/1/0/2
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy input SP-FOUR-CLASS-EXP-QUEUING
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy output SP-FOUR-CLASS-EXP-QUEUING


 ! This section applies the four-class interface QoS policy map to a
 ! core-to-core (P-to-P) interface in both directions
RP/0/RP0/CPU0:CRS-3(config)# interface HundredGigE 0/1/1/0
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy input SP-FOUR-CLASS-EXP-QUEUING
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy output SP-FOUR-CLASS-EXP-QUEUING
```

You can verify the configuration in Example 34-4 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **output**

Six-Class-of-Service SP Core Model

The six-class SP model is shown in Figure 32-3 and Figure 33-5 in Chapters 32 and 33, respectively. The following sections present fabric QoS and interface QoS policies that correspond with this six-class SP model.

Six-Class-of-Service Fabric QoS Policy

Example 34-5 shows the fabric QoS policy for the six-class SP model. Because there are only three queues available for fabric QoS policies

- MPLS EXP 5 is mapped to the High-Priority fabric Q.
- MPLS EXPs 1 through 4 are mapped to the Assured Forwarding fabric Q (which is provisioned at an aggregate bandwidth value to accommodate these combined traffic classes).
- All other EXP values are mapped (by class-default) to the Best Effort fabric Q.

Example 34-5 *Cisco CRS-3 Six-Class SP Model Fabric QoS Policy*

```
 ! This section configures the class maps for six-class fabric QoS
RP/0/RP0/CPU0:CRS-3(config-cmap)# class-map match-any SP-FABQOS-HI
RP/0/RP0/CPU0:CRS-3(config-cmap)#  match mpls experimental topmost 5
 ! Matches MPLS EXP 5 (corresponds to the SP-REALTIME class)
RP/0/RP0/CPU0:CRS-3(config-cmap)# class-map match-any SP-FABQOS-AF
RP/0/RP0/CPU0:CRS-3(config-cmap)#  match mpls experimental topmost 1 2 3 4
 ! Matches MPLS EXP 1-4 (corresponds to guaranteed BW SP classes)

 ! This section configures the six-class fabric QoS policy map
RP/0/RP0/CPU0:CRS-3(config-pmap)# policy-map SP-SIX-CLASS-FABQOS
RP/0/RP0/CPU0:CRS-3(config-pmap)#  class SP-FABQOS-HI
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  priority
 ! The SP-FABQOS-HP gets High-Priority (strict-priority)
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-FABQOS-AF
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 72
 ! The SP-FABQOS-AF gets Assured Forwarding with 72% BWR
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class class-default
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 28
 ! The default-class gets Best Effort with 28% BWR



 ! This section applies the fabric QoS policy to the switch fabric
RP/0/RP0/CPU0:CRS-3(config)# switch-fabric
RP/0/RP0/CPU0:CRS-3(config-switch-fabric)# service-policy SP-SIX-CLASS-FABQOS
 ! Applies the Fabric QoS policy to the switch fabric
```

You can verify the configuration in Example 34-5 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show controllers fabricq queues**
- **show controllers fabricq statistics**

Six-Class-of-Service Interface QoS Policy

Example 34-6 shows the interface QoS policy for the six-class SP model.

Example 34-6 *Cisco CRS-3 Six-Class SP Model Interface QoS Policy*

```
 ! The class maps for this policy are essentially the same as
 ! in Example 33-11
 ! (with the exception of substituting "match-all" with "match-any")
 ! and so, these will not be repeated here for the sake of brevity


 ! This section configures the six-class interface QoS policy map
RP/0/RP0/CPU0:CRS-3(config-pmap)# policy-map SP-SIX-CLASS-EXP-QUEUING
RP/0/RP0/CPU0:CRS-3(config-pmap)#  class SP-REALTIME-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# police rate percent 10
 ! An explicit policer is required on the priority-class (CRS-MSC-140G)
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  priority level 1
 ! The SP-REALTIME-EXP class will receive strict-priority servicing
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-AF4-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 22
 ! The SP-AF4-CLASS-EXP class will receive 22% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-AF3-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 11
 ! The SP-AF3-CLASS-EXP class will receive 11% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-AF2-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 11
 ! The SP-AF2-CLASS-EXP class will receive 11% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-AF1-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 28
 ! The SP-AF1-CLASS-EXP class will receive 28% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class class-default
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 28
 ! Class-default will receive 28% bandwidth remaining
 ! This section applies the six-class interface QoS policy map to a
 ! core-to-edge (P-to-PE) interface in both directions
RP/0/RP0/CPU0:CRS-3(config)# interface TenGigE 0/1/0/2
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy input SP-SIX-CLASS-EXP-QUEUING
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy output SP-SIX-CLASS-EXP-QUEUING


 ! This section applies the six-class interface QoS policy map to a
 ! core-to-core (P-to-P) interface in both directions
RP/0/RP0/CPU0:CRS-3(config)# interface HundredGigE 0/1/1/0
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy input SP-SIX-CLASS-EXP-QUEUING
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy output SP-SIX-CLASS-EXP-QUEUING
```

You can verify the configuration in Example 34-6 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **output**

Eight-Class-of-Service SP Core Model

The eight-class SP model is shown in Figure 32-4 and Figure 33-6 in Chapters 32 and 33, respectively. The following sections present fabric QoS and interface QoS policies that correspond with this eight-class SP model.

Eight-Class-of-Service Fabric QoS Policy

Example 34-7 shows the fabric QoS policy for the eight-class SP model. Because there are only three queues available for fabric QoS policies

- MPLS EXP 5 is mapped to the High-Priority fabric Q.
- MPLS EXPs 1, 2, 3, 4, 6, and 7 are mapped to the Assured Forwarding fabric Q (which is provisioned at an aggregate bandwidth value to accommodate these combined traffic classes).
- All other MPLS EXP values (which in this case is only MPLS EXP 0) are mapped (by class-default) to the Best Effort fabric Q.

Example 34-7 *Cisco CRS-3 Eight-Class SP Model Fabric QoS Policy*

```
 ! This section configures the class maps for eight-class fabric QoS
RP/0/RP0/CPU0:CRS-3(config-cmap)# class-map match-any SP-FABQOS-HI
RP/0/RP0/CPU0:CRS-3(config-cmap)#  match mpls experimental topmost 5
 ! Matches MPLS EXP 5 (corresponds to the SP-REALTIME class)
RP/0/RP0/CPU0:CRS-3(config-cmap)# class-map match-any SP-FABQOS-AF
RP/0/RP0/CPU0:CRS-3(config-cmap)#  match mpls experimental topmost 1 2 3 4 6 7
 ! Matches MPLS EXP 1-4 and 6 & 7 (guaranteed BW SP classes)

 ! This section configures the eight-class fabric QoS policy map
RP/0/RP0/CPU0:CRS-3(config-pmap)# policy-map SP-EIGHT-CLASS-FABQOS
RP/0/RP0/CPU0:CRS-3(config-pmap)#  class SP-FABQOS-HI
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  priority
 ! The SP-FABQOS-HP gets high-priority (strict-priority)
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-FABQOS-AF
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 73
 ! The SP-FABQOS-AF gets assured forwarding with 73% BWR
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class class-default
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 27
 ! The default-class gets best effort with 27% BWR


 ! This section applies the fabric QoS policy to the switch fabric
RP/0/RP0/CPU0:CRS-3(config)# switch-fabric
RP/0/RP0/CPU0:CRS-3(config-switch-fabric)# service-policy SP-EIGHT-CLASS-FABQOS
 ! Applies the fabric QoS policy to the switch fabric
```

You can verify the configuration in Example 34-7 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show controllers fabricq queues**
- **show controllers fabricq statistics**

Eight-Class-of-Service Interface QoS Policy

Example 34-8 shows the interface QoS policy for the eight-class SP model.

Example 34-8 *Cisco CRS-3 Eight-Class SP Model Interface QoS Policy*

```
 ! The class-maps for this policy are essentially the same as
 ! in Example 33-15
 ! (with the exception of substituting "match-all" with "match-any")
 ! as such, these will not be repeated here for the sake of brevity


 ! This section configures the eight-class interface QoS policy-map
RP/0/RP0/CPU0:CRS-3(config-pmap)# policy-map SP-EIGHT-CLASS-EXP-QUEUING
RP/0/RP0/CPU0:CRS-3(config-pmap)#  class SP-REALTIME-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# police rate percent 10
 ! An explicit policer is required on the priority-class (CRS-MSC-140G)
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  priority level 1
 ! SP-REALTIME-EXP class will receive strict-priority servicing
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-CONTROL-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 11
 ! The SP-CONTROL-EXP class will receive 11% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-AF4-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 33
 ! The SP-AF4-CLASS-EXP class will receive 33% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-AF3-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 11
 ! The SP-AF3-CLASS-EXP class will receive 11% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-AF2-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 11
 ! The SP-AF2-CLASS-EXP class will receive 11% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-AF1-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 6
 ! The SP-AF1-CLASS-EXP class will receive 6% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class SP-SCAVENGER-CLASS-EXP
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 1
 ! The SP-SCAVENGER-EXP class will receive 1% bandwidth remaining
RP/0/RP0/CPU0:CRS-3(config-pmap-c)# class class-default
RP/0/RP0/CPU0:CRS-3(config-pmap-c)#  bandwidth remaining percent 27


 ! This section applies the eight-class interface QoS policy map to a
 ! core-to-edge (P-to-PE) interface in both directions
RP/0/RP0/CPU0:CRS-3(config)# interface TenGigE 0/1/0/2
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy input SP-EIGHT-CLASS-EXP-QUEUING
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy output SP-EIGHT-CLASS-EXP-QUEUING


 ! This section applies the eight-class interface QoS policy map to a
 ! core-to-core (P-to-P) interface in both directions
RP/0/RP0/CPU0:CRS-3(config)# interface HundredGigE 0/1/1/0
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy input SP-EIGHT-CLASS-EXP-QUEUING
RP/0/RP0/CPU0:CRS-3(config-if)# service-policy output SP-EIGHT-CLASS-EXP-QUEUING
```

You can verify the configuration in Example 34-8 with the following commands:

- **show class-map reference list**
- **show policy-map pmap** *name*
- **show policy-map interface**
- **show qos interface** *interface w/x/y/z* **output**

Summary

This chapter began by identifying the key QoS role of an SP core router as preventing packet drops due to either internal or interface congestion. In addition, the Cisco CRS-3 was recognized as being a platform custom engineered to serve in this role.

The internal hardware architecture of the CRS-3 routing system was overviewed, identifying the various points where QoS is applied internally and the mechanisms for doing so. Following this, the methods of administering (ingress and egress) interface QoS policies, in addition to internal fabric QoS policies, were discussed. Special emphasis was placed on how fabric QoS policies can control (or override) ingress interface QoS policies, if these are not aligned.

Then, with the groundwork laid, sets of configurations for fabric QoS and interface QoS policies were presented for each of the SP models presented in this section:

- Four-Class-of-Service SP model
- Six-Class-of-Service SP model
- Eight-Class-of-Service SP model

Additional Reading

Cisco IOS XR Modular Quality of Service Configuration Guide for the Cisco CRS Router, Release 4.3.x: http://www.cisco.com/en/US/docs/routers/crs/software/crs_r4.3/qos/configuration/guide/b_qos_cg43xcrs.html

Chapter 35

MPLS VPN QoS Design Case Study

Tifosi has decided to migrate their private WAN and branch networks to Layer 3 MPLS VPN services to realize cost-savings and to standardize all their (LAN and WAN) networking interfaces to Gigabit Ethernet.

Tifosi will continue to use their Cisco ASR 1000 series routers (each with ESP-20s and SIP-10s) to connect their main campus networks to the MPLS VPN service provider. Similarly, they will continue to use their Cisco 3925E ISR G2s or Cisco 3945E ISR G2s as their (unmanaged) customer-edge routers to connect their branches to the MPLS VPN.

They have decided to use a six-CoS model from Maranello Networks (their local service provider), which allows customers to define their per-class bandwidth allotments in 5 percent increments. However, Maranello Networks charges premiums rates for the amount of bandwidth allocated to their priority service class.

That being the case, Tifosi has decided to transmit Cisco TelePresence traffic (provisioned over their private networks as the Real-Time Interactive application class) over the MPLS VPN in a nonpriority service class because this option seemed more economically viable and the quality of the streams was still very high.

The QoS policies to be applied to Tifosi's and Maranello Networks's MPLS VPN are shown in Figure 35-1.

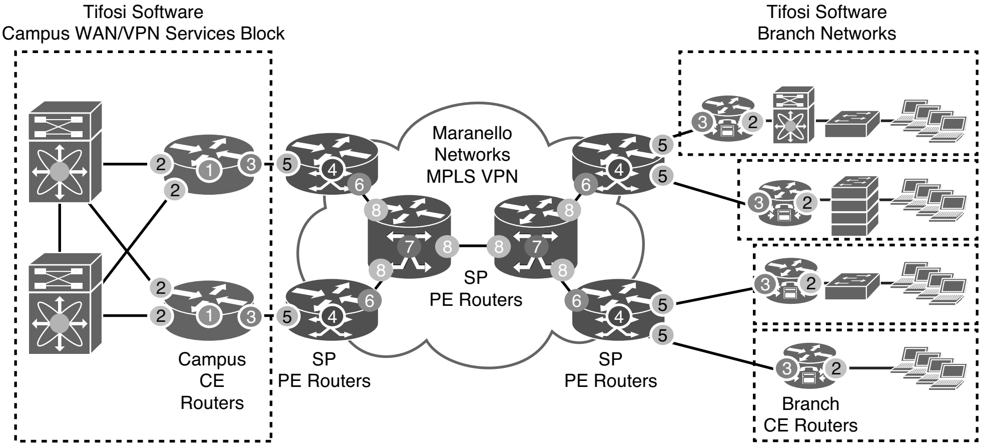


Figure 35-1 *Tifosi Software Case Study: MPLS VPN Network QoS Design*

Tifosi's MPLS VPN customer-edge router QoS policies are as follows:

- **Policy 1:** CE router internal QoS (Cisco ASR 1000): Enable SPA-based PLIM on GE interfaces
- **Policy 2:** CE router LAN-edge QoS policies: Perform NBAR2 protocol classification on ingress
- **Policy 3:** CE router VPN-edge QoS policies
  - Perform NBAR2 protocol classification on ingress (to restore any re-marked DSCP values to their original markings)
  - Provision eight-class queuing model—including enterprise-to-service provider mapping—on egress
- **Policy 4:** PE router internal QoS (Cisco ASR 9000): Provision fabric interface ASIC and virtual output queuing policy
- **Policy 5:** PE router customer-edge QoS:
  - Provision six-CoS SP model featuring Short Pipe Mode policing on ingress
  - Provision a six-CoS SP model featuring shaping on egress with nested Short Pipe Mode (DSCP-based) queuing policy
- **Policy 6:** PE router core-edge QoS: Six-CoS (MPLS EXP-based) queuing on egress
- **Policy 7:** P router internal QoS (Cisco CRS-3): Fabric QoS policy
- **Policy 8:** P router interface QoS: Six-CoS (MPLS EXP-based) queuing on egress

These policies are detailed in turn in the following sections.

Policy 1: CE Router Internal QoS (Cisco ASR 1000)

SPA-based internal PLIM QoS policies should be enabled on Cisco ASR 1000 routers GE interfaces connecting campus networks connecting to the MPLS VPN; SIP-based PLIM policies are no longer required because all Ethernet interfaces leverage SPA-based PLIM.

Tifosi's SPA-based PLIM policy remains unchanged and is shown in Example 30-2.

Note To minimize redundancy, where configuration examples remain unchanged from previous design or case study chapters, these are referenced rather than repeated.

Policy 2: CE Router LAN-Edge QoS Policies

The CE LAN-edge QoS policies will trust DSCP on ingress (which is a Cisco IOS default) and will perform detailed deep packet inspection via NBAR2. Tifosi's LAN-edge ingress QoS policy remains unchanged and is shown in Example 30-3.

Policy 3: CE Router VPN-Edge QoS Policies

Tifosi has purchased bandwidth to correspond to their previous private WAN/branch-edge models. However, they must map their eight-class enterprise model into a six-class service provider model. To do so, they have decided to combine their Real-Time Interactive class with their Signaling class (therefore, signaling traffic will need to be re-marked CS4). Similarly, their Bulk and Scavenger classes will be combined (but no re-marking will be required in this second instance). An additional re-marking will be required for their Multimedia Conferencing class (from AF41 to AF31) so that it will receive its own CoS (separate from Real-Time Interactive).

To restore these markings on packets that have traversed the MPLS VPN, Tifosi has chosen to apply the same NBAR2 LAN-edge policies to the ingress of their VPN edges.

While the Real-Time Interactive service class will be treated with a nonpriority service as it transits the MPLS VPN, on their customer-edge router Tifosi has elected to retain a priority servicing of this class.

Finally, the queuing policy is nested within a 50 Mbps (contracted rate) shaping policy. Because Tifosi does not know their provider's PE ingress policing burst value (Maranello Networks has declined to share this, claiming that was proprietary information), they are shaping their VPN access GE links to a 10-ms interval.

Figure 35-2 illustrates Tifosi's enterprise-to-service provider mappings for the CE VPN edge.

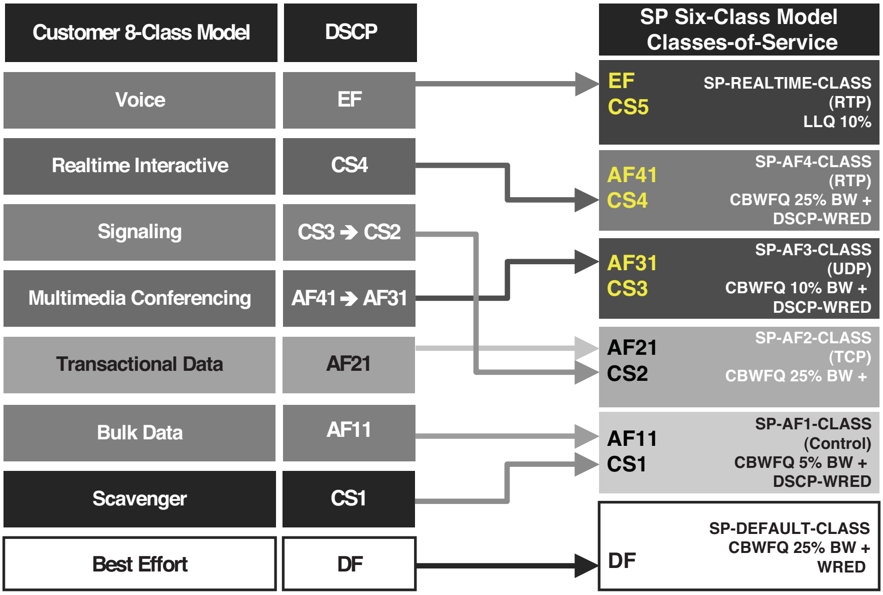

Customer 8-Class Model
DSCP
Voice
EF
Realtime Interactive
CS4
Signaling
CS3 → CS2
Multimedia Conferencing
AF41 → AF31
Transactional Data
AF21
Bulk Data
AF11
Scavenger
CS1
Best Effort
DF
SP Six-Class Model Classes-of-Service
EF CS5
SP-REALTIME-CLASS (RTP) LLQ 10%
AF41 CS4
SP-AF4-CLASS (RTP) CBWFQ 25% BW + DSCP-WRED
AF31 CS3
SP-AF3-CLASS (UDP) CBWFQ 10% BW + DSCP-WRED
AF21 CS2
SP-AF2-CLASS (TCP) CBWFQ 25% BW +
AF11 CS1
SP-AF1-CLASS (Control) CBWFQ 5% BW + DSCP-WRED
DF
SP-DEFAULT-CLASS CBWFQ 25% BW + WRED

Figure 35-2 *Tifosi Software Case study: CE VPN-Edge Enterprise-to-Service Provider Mapping*

Example 35-1 shows the corresponding configuration.

Example 35-1 *Tifosi Case Study: CE VPN-Edge QoS Policy*

```
 ! This section configures the DSCP-matching class maps
 ! for the Tifosi eight-class VPN-edge egress policy
ASR1K/ISR-G2(config-cmap)# class-map match-all VOICE
ASR1K/ISR-G2(config-cmap)#  match dscp ef
ASR1K/ISR-G2(config-cmap)# class-map match-all REALTIME-INTERACTIVE
ASR1K/ISR-G2(config-cmap)#  match dscp cs4
ASR1K/ISR-G2(config-cmap)# class-map match-all SIGNALING
ASR1K/ISR-G2(config-cmap)#  match dscp cs3
ASR1K/ISR-G2(config-cmap)# class-map match-all MULTIMEDIA-CONFERENCING
ASR1K/ISR-G2(config-cmap)#  match dscp af41
ASR1K/ISR-G2(config-cmap)# class-map match-all TRANSACTIONAL-DATA
ASR1K/ISR-G2(config-cmap)#  match dscp af21
ASR1K/ISR-G2(config-cmap)# class-map match-all BULK-DATA
ASR1K/ISR-G2(config-cmap)#  match dscp af11
ASR1K/ISR-G2(config-cmap)# class-map match-all SCAVENGER
ASR1K/ISR-G2(config-cmap)#  match dscp cs1


 ! This section defines the eight-class VPN-edge egress policy
```

```
ASR1K/ISR-G2(config-cmap)# policy-map VPN-EDGE
ASR1K/ISR-G2(config-pmap)#  class VOICE
ASR1K/ISR-G2(config-pmap-c)# priority percent 10
ASR1K/ISR-G2(config-pmap-c)# class REALTIME-INTERACTIVE
ASR1K/ISR-G2(config-pmap-c)#  priority percent 23
ASR1K/ISR-G2(config-pmap-c)# class SIGNALING
ASR1K/ISR-G2(config-pmap-c)#  bandwidth percent 2
ASR1K/ISR-G2(config-pmap-c)#  set dscp cs4
ASR1K/ISR-G2(config-pmap-c)# class MULTIMEDIA-CONFERENCING
ASR1K/ISR-G2(config-pmap-c)#  bandwidth percent 10
ASR1K/ISR-G2(config-pmap-c)#  set dscp af31
ASR1K/ISR-G2(config-pmap-c)#  fair-queue
ASR1K/ISR-G2(config-pmap-c)#  random-detect dscp-based
ASR1K/ISR-G2(config-pmap-c)# class TRANSACTIONAL-DATA
ASR1K/ISR-G2(config-pmap-c)#  bandwidth percent 25
ASR1K/ISR-G2(config-pmap-c)#  fair-queue
ASR1K/ISR-G2(config-pmap-c)#  random-detect dscp-based
ASR1K/ISR-G2(config-pmap-c)# class BULK-DATA
ASR1K/ISR-G2(config-pmap-c)#  bandwidth percent 4
ASR1K/ISR-G2(config-pmap-c)#  fair-queue
ASR1K/ISR-G2(config-pmap-c)#  random-detect dscp-based
ASR1K/ISR-G2(config-pmap-c)# class SCAVENGER
ASR1K/ISR-G2(config-pmap-c)#  bandwidth percent 1
ASR1K/ISR-G2(config-pmap-c)# class class-default
ASR1K/ISR-G2(config-pmap-c)#  bandwidth percent 25
ASR1K/ISR-G2(config-pmap-c)#  fair-queue
ASR1K/ISR-G2(config-pmap-c)#  random-detect


 ! This section configures the hierarchical shaper
 ! with nested queuing policy
ASR1K/ISR-G2(config-pmap)# policy-map CE-EDGE-SHAPING
ASR1K/ISR-G2(config-pmap)# class class-default
ASR1K/ISR-G2(config-pmap-c)# shape average 50M 500K
ASR1K/ISR-G2(config-pmap-c)# service-policy VPN-EDGE


 ! This section applies the sub-line-rate shaper to an ASR CE VPN edge
ASR1K(config)#interface GigabitEthernet0/0/2
ASR1K(config-if)# service-policy input LAN-EDGE
 ! The (NBAR2) LAN-EDGE policy is shown in Example 30-3
ASR1K(config-if)# service-policy output CE-EDGE-SHAPING


 ! This section applies the sub-line-rate shaper to an ISR CE VPN edge
```

```
ISR-G2(config)#interface GigabitEthernet0/2
ISR-G2(config-if)# service-policy input LAN-EDGE
 ! The (NBAR2) LAN-EDGE policy is shown in Example 30-3
ISR-G2(config-if)# service-policy output CE-EDGE-SHAPING
```

Policy 4: PE Router Internal QoS (Cisco ASR 9000)

No explicit internal QoS policy is required to be configured on the Cisco ASR 9000 PE router. However, the IOS XR software on the ASR 9000 will automatically set the internal queuing policies for the fabric interface ASIC and virtual output queues to correspond to the interface QoS policies.

In this specific case, the SP-REALTIME class from Example 34-2 will receive an internal strict-priority service at these internal nodes; all the other traffic classes will receive a default service.

Policy 5: PE Router Customer-Edge QoS

The PE edge will have a six-class Short Pipe Mode ingress policer, which is identical to the six-class Pipe Mode ingress policer detailed in Example 33-10.

As for the egress PE policy, Maranello Networks is using a six-class Short Pipe queuing model that is nested within a contracted-rate shaper to 50 Mbps.

Example 35-2 details these combined ingress and egress policies.

Example 35-2 *Tifosi Case Study: Maranello Networks's PE (Customer-Facing) Edge QoS Policy*

```
 ! This section configures the Short Pipe Mode six-class
 ! egress queuing policy map
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-SIX-CLASS-SHORT-PIPE
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  priority
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF4-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 28
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# random-detect dscp-based
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF3-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 11
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# random-detect dscp-based
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF2-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 28
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# random-detect dscp-based
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF1-CLASS
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 5
```

```
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# random-detect dscp-based
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 28

! This section configures the shaper policy map
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-SHAPER-50M
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# shape average 50M 500K
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# service-policy SP-SIX-CLASS-SHORT-PIPE


 ! This section attaches the policy to the PE-to-CE interface
RP/0/RSP0/CPU0:ASR9K(config)# interface GigE 0/2/0/0
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy input SP-SIX-CLASS-INGRESS-MODEL-
SHORT-PIPE
 ! This policing policy is identical to Example 33-10
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy output SP-SHAPER-50M
```

Policy 6: PE Router Core-Edge QoS

After traffic has been policed at the ingress PE edge, it is forwarded to the core and queued based on Maranello Networks's locally significant MPLS EXP markings, as shown in Example 35-3.

Example 35-3 *Tifosi Case Study: Maranello Networks's PE (Core-Facing) Edge QoS Policy*

```
  ! These class maps for this six-class MPLS EXP-based queuing policy
  ! are identical to those detailed in Example 33-11


 ! This section configures Maranello Networks's
 ! six-class EXP-based queuing policy map
RP/0/RSP0/CPU0:ASR9K(config)# policy-map SP-SIX-CLASS-EXP-QUEUING
RP/0/RSP0/CPU0:ASR9K(config-pmap)# class SP-REALTIME-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  priority
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF4-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 28
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF3-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 11
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF2-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 28
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class SP-AF1-CLASS-EXP
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 5
```

```
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:ASR9K(config-pmap-c)#  bandwidth remaining percent 28


 ! This section applies the service policy to the PE-to-P interface
RP/0/RSP0/CPU0:ASR9K(config)# interface tenGigE 0/1/0/2
RP/0/RSP0/CPU0:ASR9K(config-if)# service-policy output SP-SIX-CLASS-EXP-QUEUING
```

Policy 7: P Router Internal QoS (Cisco CRS-3)

The Cisco CRS-3 will require a fabric QoS policy that corresponds to interface QoS policies. The six-class SP-model fabric QoS policy for Maranello Networks is identical to that shown in Example 34-5.

Policy 8: P Router Interface QoS

The core interface QoS policies are essentially identical to Example 34-6, the only difference being that all **match-all** logical operators in the class maps must be replaced with **match-any** operators on the CRS-3.

Summary

This chapter continued the case study example of Tifosi Software and applied their strategic eight-class end-to-end QoS model in their migration from private WAN networks to a MPLS VPN. Tifosi has chosen a six-class CoS model from Maranello Networks, a local service provider.

Therefore, Tifosi had to manage some minor (but critical) configuration changes on their Cisco ASR 1000 and ISR-G2 routers because these now assumed the role of customer-edge routers. These changes related principally to collapsing traffic classes to manage their enterprise-to-service provider mapping, in addition to nesting their queuing policies within a contracted-rate shaper.

To round out the case study, configuration details were also provided (or referenced) for Maranello Networks's service provider MPLS VPN QoS policies.

Additional Reading

Classifying and Scheduling Packets for ASR 1000 Series: http://www.cisco.com/en/US/partner/docs/interfaces_modules/shared_port_adapters/configuration/ASR1000/ASRimpqos.html

Cisco IOS Release 15M&T: QoS Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos/config_library/15-mt/qos-15-mt-library.html

Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Configuration Guide, Release 4.3.x: http://www.cisco.com/en/US/docs/routers/asr9000/software/asr9k_r4.3/qos/configuration/guide/b_qos_cg43xasr.html

Cisco IOS XR Modular Quality of Service Configuration Guide for the Cisco CRS Router, Release 4.3.x: http://www.cisco.com/en/US/docs/routers/crs/software/crs_r4.3/qos/configuration/guide/b_qos_cg43xcrs.html

This page intentionally left blank

Chapter 36

IPsec VPN QoS Considerations and Recommendations

In recent years, the IPsec suite of technologies has become a standard and ubiquitous element of most IP networks and continues to see wide adoption across the industry. IPsec virtual private networks (VPNs) can be found everywhere—from site-to-site data center interconnects, to hub-and-spoke networks connecting large numbers of remote locations to a head office, to remote access for home users. In fact, the list of possible use cases for IPsec VPNs is almost limitless.

Of course, there are many ways to create VPNs other than IPsec, such as QinQ tunneling in an L2 MAN network, L2 and L3 MPLS VPNs, or even dedicated circuits using legacy protocols. The list of different VPN technologies is extensive. However, IPsec offers the unique blend of proven security, simplicity, flexible deployment models, and a relatively low entry cost that has made it appealing to many network engineers, no matter the size of the network.

Deploying IPsec VPNs introduces many new topics for consideration when using real-time applications that require quality of service (QoS), including the following:

- IPsec VPN topologies
- QoS classification of IPsec packets
- The IOS preclassify feature
- MTU considerations
- Compression strategies over VPN
- Antireplay considerations

IPsec VPN Topologies

Recommendation:

- Understand the different VPN topologies, and in particular the impact packet encapsulation on QoS capabilities.

One of the most attractive features of IPsec technology is its adaptability to a variety of different types of network topology. Some of the most common IPsec VPN architectures that are in use today are as follows:

- Standard IPsec VPNs
- IPsec with GRE
- Remote-access VPN
- Tunneled VPN architectures (For example, DMVPN and FlexVPN)
- Tunnel-less VPN (for example, GETVPN)

The following sections discuss the first three of these. DMVPN and GETVPN are discussed thoroughly in the chapters that follow.

Standard IPsec VPNs

Standard IPsec is the most traditional of all VPN topologies and it finds a wide range of support in both Cisco and other vendors' networking products. Standard IPsec can be used in both point-to-point and point-to-multipoint topologies, depending on the feature support in the headend device being used. Standard IPsec VPNs are further broken down into the two following categories: tunnel mode and transport mode VPNs.

Tunnel Mode

Tunnel mode is the default IPsec mode of operation in Cisco IOS routers. With tunnel mode, the entire IP packet is protected by IPsec, meaning that the sending VPN router encrypts the entire original IP packet and adds a new IP header to this packet. Tunnel mode is a popular deployment option for IPsec because it supports routing protocols, such as Open Shortest Path First (OSPF) Protocol and Intermediate System-to-Intermediate System (IS-IS) Protocol. A key requirement in running these interior gateway protocols (IGPs) over a VPN tunnel is the ability to support multicast, which is a key feature of IPsec tunnel mode. Figure 36-1 illustrates an example of an IPsec tunnel mode packet structure.

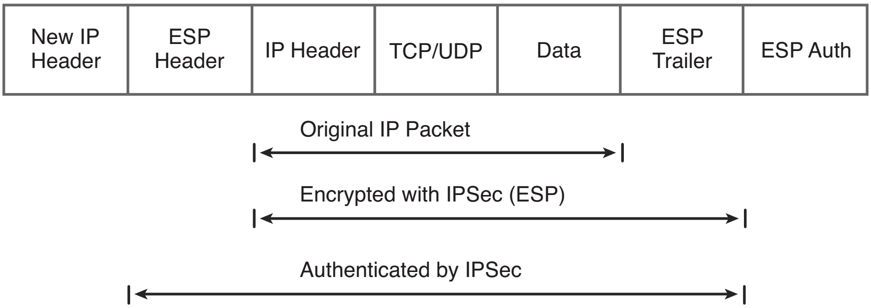


Figure 36-1 *IPsec Tunnel Mode IP Packet Using ESP*

Transport Mode

Transport mode is a common deployment mode for IPsec and is often used for encrypted peer-to-peer communications. Unlike tunnel mode, transport mode does not encase the original IP packet into a new packet. Rather, with IPsec transport mode only the payload of the IP packet is encrypted while the original IP headers are preserved, in effect being copied to the outside of the new IPsec packet. Because transport mode leaves the original IP packet header intact, it is not possible to combine transport mode with certain IP services such as multicast, or other routing protocols that rely on multicast (for example, OSPF and Enhanced Interior Gateway Routing Protocol [EIGRP]). For example, transport mode is often used in conjunction with generic routing encapsulation (GRE), where the entire GRE tunnel is encrypted with IPsec (described in more detail in the following section). Figure 36-2 illustrates an example of an IPsec transport mode packet structure.

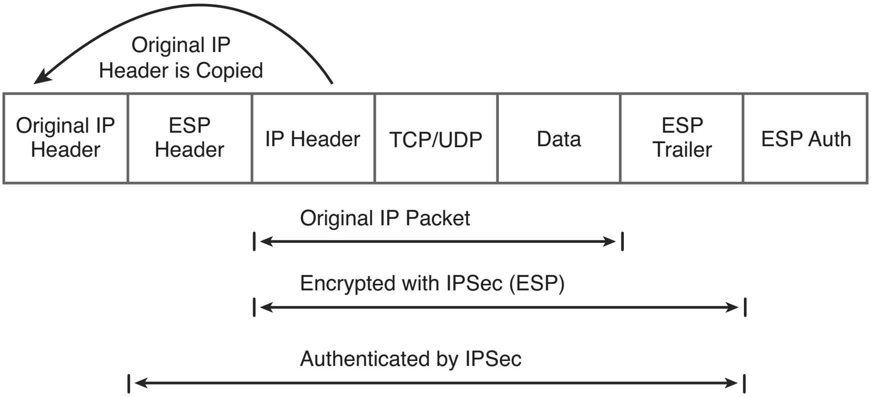


Figure 36-2 *IPsec Transport Mode IP Packet using ESP*

IPsec with GRE

Many networks today utilize generic routing encapsulation or GRE (IP Protocol 47) to enable VPN services that connect disparate networks. For example, GRE is one of the key building blocks of VRF Lite, a technology allowing related virtual routing and forwarding (VRF) instances running on different routers to be interconnected across an IP network, while maintaining their separation from both the global routing table and the other VRFs. In this case, GRE tunnels each IP packet into a new IP packet, complete with a new IP header. This approach is often a cost-effective alternative to deploying a new MPLS network to connect VRFs, and is thus called VRF Lite.

Another valuable use case for GRE comes from its capability to carry non-IP protocols, such as IPX or DECnet. Although this is a rapidly diminishing use case for GRE, there are still many networks in existence today that utilize non-IP protocols and use GRE. Beyond these basic examples, GRE has many other applications where disconnected networks can be brought together across an IP cloud.

When GRE is used as a VPN or tunneling technology, it is often desirable to encrypt the GRE tunnel so that the privacy and authentication of the connection can be ensured. Integrating GRE with either IPsec tunnel mode or transport mode has been debated. One point of consideration is that tunnel mode adds an additional 20 bytes to the total packet size. Essentially, either tunnel or transport mode work in a GRE over IPsec implementation. However, several restrictions with transport mode should be considered. For example, if the crypto tunnel transits either a Network Address Translation (NAT) or Port Address Translation (PAT) device, tunnel mode is required.

Because all of these IPsec and GRE VPN topologies involve new packet headers, the way QoS classification is handled is different from normal QoS packet classification. Figure 36-3 illustrates the structure of a GRE over IPsec packet.

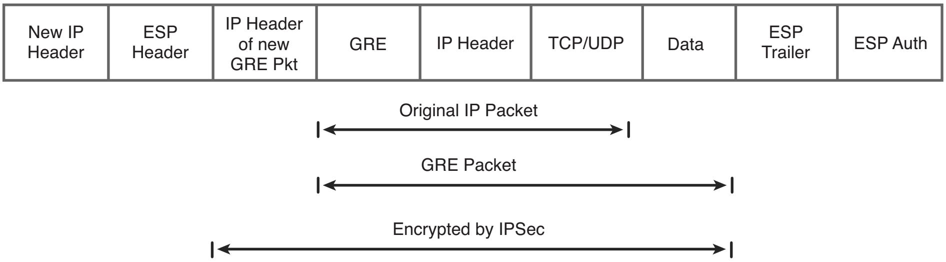


Figure 36-3 *A GRE Packet Encrypted with IPsec*

Remote-Access VPNs

Cisco's primary remote-access VPN client is the AnyConnect Secure Mobility Client, which supports both IPsec and Secure Sockets Layer (SSL) encryption. AnyConnect client termination is supported on a wide variety of headend devices, such as Adaptive Security Appliances (ASA) firewalls and Integrated Services Routers (ISR) routers. From a client perspective, AnyConnect is supported on everything from Windows, OS X, Linux, iOS, Android, Google Chrome, and others.

Although AnyConnect does not support any specific QoS classification, queuing, or policing tools, it does have one important feature that significantly improves the quality of experience when using real-time applications. This feature is called Data Transport Layer Security (DTLS). Before discussing how DTLS works, let's take a look at the application performance problem that DTLS was designed to fix.

When running a secure SSL connection over AnyConnect, the client establishes a TCP-based SSL connection to the remote headend concentrator (such as an ASA firewall). This means that all traffic from the client, including voice, video, and data, will all traverse the SSL TCP connection. However, when real-time UDP-based applications are transported over a connection-oriented TCP tunnel, there is usually a major impact to real-time application performance.

TCP works by using a sliding window mechanism to ensure that data is correctly received at the far station. Now, if a packet inside this sliding widow is not correctly received by the receiving host (due to packet loss), the connection is momentarily paused while the sending station is asked by the receiving station to resend the entire segment of data from the window with the missing data. Once the data has been properly received, the next block of data can be sent.

With data applications, this is something that helps immensely since anything that has been lost due to "network events" can be recovered in a fairly speedy way. Unfortunately, this does not work so well for real-time applications. If you were to tunnel a UDP-based real-time application through a TCP tunnel, if even a single packet is lost or corrupted in transit, the session will pause, thus severely delaying the real-time application. For example, if this was a video call, it would mean that even the slightest packet loss would cause the entire video session to stop and start continually, resulting in extremely poor video quality.

To overcome this challenge and improve the performance for real-time applications, DTLS was added to AnyConnect to provide a way to optimize latency-sensitive traffic, such as VoIP and video traffic. Unlike normal SSL connections, DTLS is a datagram technology, meaning it uses UDP packets, not TCP.

DTLS in the AnyConnect client works in the following way: When AnyConnect first connects to the headend device, a TCP-based SSL tunnel is set up. Once the SSL session is fully established, the client negotiates a new UDP-based DTLS tunnel, which is reserved for the exclusive use of real-time applications. The UDP nature of DTLS allows the RDP voice and video packets to be transmitted unhindered. If there are any sudden packet losses or unexpected network events, the session does not pause and the lost packets are not re-sent. Rather, the next packets continue to flow as normal, thus giving a more fluid experience to the voice or video endpoint.

The decision of how to send the data, meaning which tunnel to send it over, is dynamic. As each network bound data packet is processed, there is a point in the code where the decision is made by the AnyConnect application to use either the regular TCP SSL tunnel or the DTLS tunnel. Therefore, AnyConnect has the intelligence to dynamically select the type of tunnel used, depending on the type of traffic that is being sent at each moment. This implementation greatly enhances the overall user experience when voice, video, or other real-time applications are used in conjunction with AnyConnect.

QoS Classification of IPsec Packets

Recommendation:

- Understand the default capabilities of Cisco VPN routers to automatically copy the ToS byte from the inner packet to the VPN packet header.

As has been discussed throughout this book, QoS classification is the process of matching one or more fields in a packet's TCP/IP header and then placing that packet in a group or class of traffic. With the help of packet classification, you can divide network traffic into varying priority levels, or classes of service. As you have seen in the preceding chapters, classification for QoS is primarily based on matching the Type of Service (ToS) byte field, and in particular, the differentiated services code point (DSCP) markings. With IPsec encryption, however, a problem emerges. When an IP packet is encrypted through IPsec, the entire packet, including the original packet header, is encrypted, thus making any QoS mechanism ineffective when applied to the original packet. In fact, because the payload is encrypted by IPsec, the router loses the ability to even know where the original packet headers begin and end.

To deal with this, Cisco routers by default copy the ToS field from the original IP packet and write it into the new IPsec packet header, thus allowing classification to still be accomplished by matching DSCP values. The great thing about this feature is that it is done by default and requires no extra configuration steps to enable.

The case is similar for GRE encapsulation. The default behavior of Cisco routers is to dynamically copy the ToS byte from the original IP packet to the IP header of the encapsulating GRE packet. If the GRE packet is in turn encrypted by IPsec, the ToS byte is in turn automatically copied from the GRE packet into the header of the IPsec packet, meaning that the original ToS byte is copied twice. Figure 36-4 illustrates this behavior of an IPsec-encrypted GRE packet.

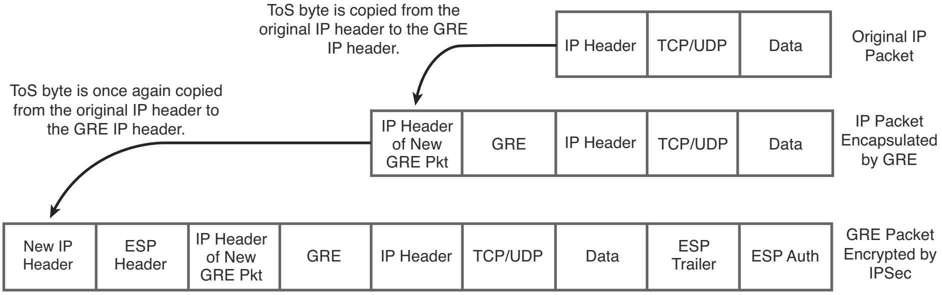

ToS byte is copied from the original IP header to the GRE IP header.
IP Header
TCP/UDP
Data
Original IP Packet
ToS byte is once again copied from the original IP header to the GRE IP header.
IP Header of New GRE Pkt
GRE
IP Header
TCP/UDP
Data
IP Packet Encapsulated by GRE
New IP Header
ESP Header
IP Header of New GRE Pkt
GRE
IP Header
TCP/UDP
Data
ESP Trailer
ESP Auth
GRE Packet Encrypted by IPSec

Figure 36-4 *IP ToS Byte Preservation*

Note here that only the ToS byte, and no other part of the original IP header is copied to either the GRE or the IPsec headers. You can imagine that in some cases it might be desirable to classify packets based on other information in the IP header, such as source IP, destination IP, ports, flags, and so on. Because IOS by default only copies the ToS byte to the new IPsec packet header, another mechanism is required to support QoS classification requirements for these other fields.

The IOS Preclassify Feature

Recommendations:

- Be aware of the limitations of QoS classification when using something other than the ToS byte.
- Use the IOS preclassify feature for all non-ToS types of QoS classification.
- As a best practice, enable this feature for all VPN connections.

The preclassify feature has been billed as the "QoS for VPNs" feature in Cisco's IOS documentation. This has led some people to the mistaken belief that the feature is absolutely required to enable any QoS on all IPsec or GRE VPN connections. Of course, this is not the full story, as was discussed in the previous section. As you have seen, by default IOS will copy the ToS byte to the new IP header so that classification based on DSCP will work as expected, requiring no special user configuration for either IPsec or GRE. However, if classification is desired based on other key fields in the IP header, such as the source or destination IP addresses, TCP or UDP ports, flags, and so on, the IOS preclassify feature is required.

By default in IOS, encryption is performed before QoS classification, as illustrated in Figure 36-5.

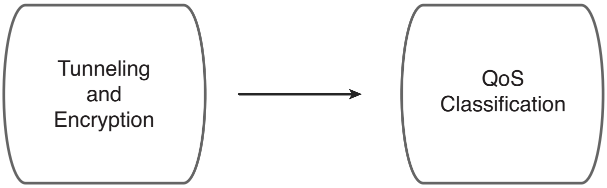


Figure 36-5 *Normal Order of Operations*

The preclassify feature has the *effect* of reversing this process, allowing QoS classification to be performed before tunneling and encryption, as illustrated in Figure 36-6.

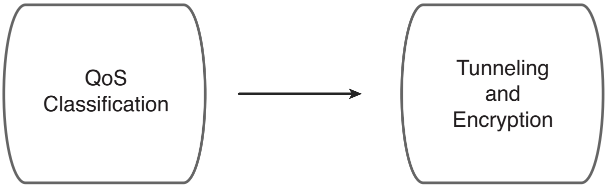
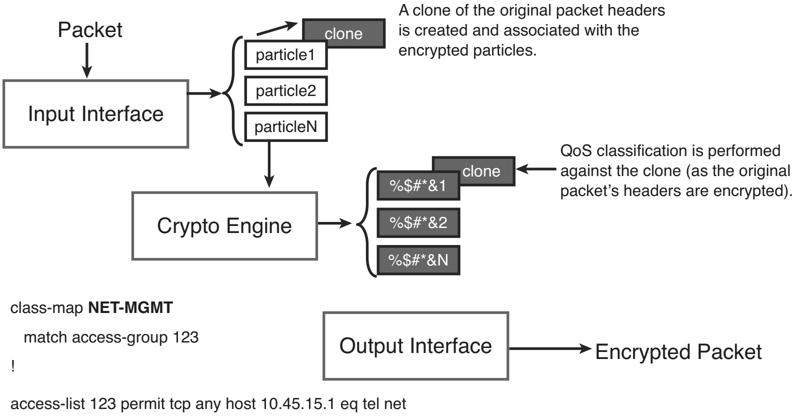


Figure 36-6 *How the Preclassify Feature Affects the Effective Order of Operations*

Technically speaking, the order of operations is not actually changing here. When the preclassify feature is being used, although this is a useful way to think of things. In reality, the original packet header is actually being cloned and kept in the router's memory so that it can be used by the QoS classification process *after* tunneling and encryption. Figure 36-7 illustrates how the original packet header is cloned to be used later for classification.


Packet
Input Interface
clone
particle1
particle2
particleN
A clone of the original packet headers is created and associated with the encrypted particles.
Crypto Engine
clone
%$#*&1
%$#*&2
%$#*&N
QoS classification is performed against the clone (as the original packet's headers are encrypted).
Output Interface
Encrypted Packet

```
class-map NET-MGMT
  match access-group 123
!
access-list 123 permit tcp any host 10.45.15.1 eq tel net
```

Figure 36-7 *The Operation of the Preclassify Feature*

A key point to remember here is that the preclassify feature is applicable only at the encrypting router's outbound interface (either physical or tunnel interface). The clone that is made of the original IP packet header is only available to the encrypting router, meaning that any downstream routers in the path will not be able to make QoS classification decisions based on anything other than the ToS byte information. Because QoS is deployed on a per-hop basis (PHB), this could pose a problem if there are routers between the encrypting stations that need to classify on something other than the ToS field information.

As a general best-practices rule, it is recommended that you always enable the preclassify feature on the VPN router, even if classification is only being done based on the ToS byte. This approach eliminates the possibility that its configuration will be overlooked if the QoS policy is later changed to include matching criteria on other fields in the IP header. Furthermore, testing has also found that enabling the preclassify feature has very little impact to the router's overall performance, meaning there are no real drawbacks to implementing this feature as a best practice.

For IPsec tunnels, the **pre-classify** command is applied on the crypto map, which allows the configuration to be supported on a per-tunnel basis. In this way, QoS features such as shaping, policing, queuing, and weighted random early detection (WRED) on the physical interface carrying the crypto map are able to classify packets based on what ever field is desired.

For GRE tunnels, the **pre-classify** command works in the same way as for IPsec tunnels, in that the router makes a clone of the original IP packet header that can be applied to the new GRE packet. In the case of GRE, however, the **pre-classify** command is applied on the tunnel interface.

Example 36-1 demonstrates how the preclassify feature is used in an IPsec tunnel.

Example 36-1 *Preclassify Used on an IPsec Tunnel*

```
crypto map MYMAP 10 ipsec-isakmp
  set peer 10.1.1.1
  set transform-set MYSET
  match address ACL
  qos pre-classify
!
interface FastEthernet 0/0
  ip address 10.1.1.100
  crypto map MYMAP
```

Example 36-2 demonstrates how to enable QoS preclassify for a GRE tunnel.

Example 36-2 *Preclassify used on a GRE Tunnel*

```
interface Tunnel500
  ip address 192.168.1.0 255.255.255.252
  qos pre-classify
  tunnel source 192.168.1.1
  tunnel destination 192.168.2
!
interface FastEthernet 0/0
  ip address 10.1.1.100
```

Naturally, there are many cases where you will find it necessary to encrypt the GRE tunnel itself, and you may also need to classify the traffic on *both* DSCP and other fields in the IP header. This is a fairly common requirement because it provides both the benefits of GRE to extend a virtual network over a cloud (like the Internet) while at the same time offering necessary encryption services. Figure 36-8 illustrates an example of this common deployment model.

In this example, the natural question is if you were to use the **pre-classify** command, where should it be configured? Should it be on the VPN tunnel interface, or on the physical interface where encryption happens? In a case like this, the **pre-classify** command should be configured on *both* the physical and tunnel interfaces, as demonstrated in Example 36-3.

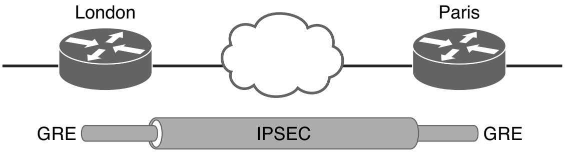


Figure 36-8 *IPsec Encryption of a GRE Tunnel*

Example 36-3 *Preclassify Used on a GRE over IPsec Tunnel*

```
crypto map MYMAP 10 ipsec-isakmp
  set peer 10.1.1.1
  set transform-set MYSET
  match address ACL
  qos pre-classify
!
interface Tunnel500
  ip address 192.168.1.0 255.255.255.252
  qos pre-classify
  tunnel source 192.168.1.1
  tunnel destination 192.168.2
!
interface FastEthernet 0/0
  ip address 10.1.1.100
  crypto map MYMAP
```

MTU Considerations

Recommendations:

- Be aware that MTU issues can severely impact network connectivity and the quality of user experience in VPN networks.
- Use the best-practice strategies outlined in this section to deal with MTU problems.

Whenever tunneling technologies are used in a network, there is always the risk of exceeding the maximum transmission unit (MTU) somewhere in the path. In fact, unless jumbo frames are available end to end, MTU issues will almost always need be addressed when dealing with any kind of VPN technology.

For example, many times network administrators will build their IPsec/GRE tunnel over a cloud network, such as the Internet. At first, they find that everything works as expected. They are able to ping and make Voice over IP (VoIP) calls over this VPN link. Then, all of a sudden, they discover that certain applications are not working at all. Email, file server connections, and many other applications do not work, and it seems to be a mystery why these applications are failing while Internet Control Messaging Protocol (ICMP) and VoIP seems to work fine. Upon further investigation, it is revealed that incorrect handling of the MTU limitation has caused these problems.

How GRE Handles MTU Issues

When a packet that is near or at the available MTU size arrives at a VPN router, the router encapsulates the original packet in a series of new headers, thus increasing the overall size of the packet. Take the following example of how MTU issues can cause a connection issue when using a basic GRE tunnel:

1. A network with a normal 1500 byte MTUs in being used.
2. A user sends a 1500-byte data packet across the network (1460 bytes of data + 20 bytes for the TCP header and 20 bytes for the IP header).
3. The packet goes into a GRE tunnel. GRE requires a 24-byte header, meaning that the MTU of the original packet before entering the GRE tunnel cannot exceed 1476 bytes (1476 + 24 = 1500, which is the physical interface MTU).
4. The original packet is already in excess of the MTU, so the router attempts to fragment the packet into two parts.
5. The router notices that the DF bit (meaning Don't Fragment) in the original header is set to 1, meaning that this packet should not be fragmented. (Note that the DF bit is set by the sending application, and having a DF=1 is very common with many applications.)
6. The router then has no choice but to drop the original packet and the connection fails.

As you can see from this example, dealing with the 1500-byte MTU problem can cause severe network connection issues when dealing with GRE. The obvious solution here seems to point toward increasing the MTU across all network interfaces to support the larger packets. Unfortunately, because the IP packet is usually processed at the MAC layer into an Ethernet frame with a maximum size of 1518 bytes, this solution is not viable—unless, of course, your network supports jumbo frames end to end.

How IPsec Handles MTU Issues

IPsec behaves a little differently than GRE when it comes to large MTUs. Instead of adding only 24 bytes, IPsec adds to the original packet length by a *maximum* of 58 bytes, which is significantly more than GRE. When the original 1500-byte packet arrives at the IPsec crypto engine, it is thus increased to a maximum of 1558 bytes, and must be fragmented by the outbound interface. Take this example of how IPsec deals with MTU issues:

1. A host sends a 1500-byte IP packet across the network.

2. The packet is received by the VPN router and is encrypted, adding a 52-byte header to the original IP packet (Note: Although a 58-byte header is the maximum, 52-bytes is more common).
3. The packet is now 1552 bytes, and must be fragmented into two parts.
4. The IPsec router at the other end receives the two fragments, strips off the outer IP headers and recombines the two fragments.
5. IPsec at the receiving router now decrypts the recombined packet and sends the original IP packet off to the destination host.

As you can see from this example, unlike GRE, IPsec has a built-in mechanism to deal with excessive MTUs. In IOS, the IPsec process always does path MTU discovery (PMTUD) both for data packets and for its own control packets. This allows the IPsec VPN router to know that even if it is outbound MTU is acceptable, there may be routers along the path that will cause problems, and the router can then take action to preemptively fragment the packet. Even if the original packet has the DF bit set to 1, the IPsec mechanism can override this and fragment anyway because it knows it will reassemble it at the other end.

So, what is the difference between IPsec and GRE in this case? The main difference is that IPsec has the capability to reassemble the original IP packet at the receiving router, thus sending a single unfragmented packet to the destination host. In the case of pure GRE, if the DF bit were to be manually overridden on egress from the VPN router (through the use of a route map, for example), the two GRE fragments would never be reassembled on the receiving VPN router and the receiving host may likely drop the two fragments, refusing to reassemble them.

All this being said, fragmentation is usually something that you should try to avoid when using IPsec. Although hardware-based encryption is featured on almost all modern Cisco routers, packet fragmentation and reassembly is a process-switched activity, meaning that the CPU will take a major hit when fragmentation and reassembly occurs. This has the effect of reducing the hardware-based crypto engine to the level of a software-based router, which will severely impact the performance of all applications using this router.

It is beyond the scope of this book to describe all the various ways to deal with MTU issues, and strictly speaking, it does not have much to do with QoS in the traditional sense, although as can be seen, MTU issues do have a dramatic effect on application performance and quality of experience (QoE) across the network. However, it is still vital that network engineers and administrators have a good grasp of the problems that MTU issues can introduce, and the methods available to deal with these issues. The following section examines the most common approach in dealing with large MTUs caused by VPN packet headers.

Using the TCP Adjust-MSS Feature

One of the most common ways to deal with the MTU issue is through the use of the IOS *TCP adjust-MSS* feature. The TCP maximum segment size (MSS) defines the maximum amount of payload data that a host is willing to accept in a single TCP/IP datagram. During a TCP connection setup between two hosts (TCP SYN), the MSS for each side of the connection is reported to each other. Ultimately, it is the sending host that is responsible to limit the size of the datagram in each TCP segment to a value equal to or less than the MSS reported by the receiving host.

The MSS is similar to the MTU, but refers to the actual payload data portion of the IP packet, whereas MTU refers to the size of the entire IP packet, which includes the TCP and IP headers.

For example, if the MTU of an IP packet is 1500 bytes, it will include 20 bytes of IP header and 20 bytes of TCP header information. Therefore, in the case of this 1500 byte packet, the MSS would actually be 1460 bytes, as illustrated in Figure 36-9.

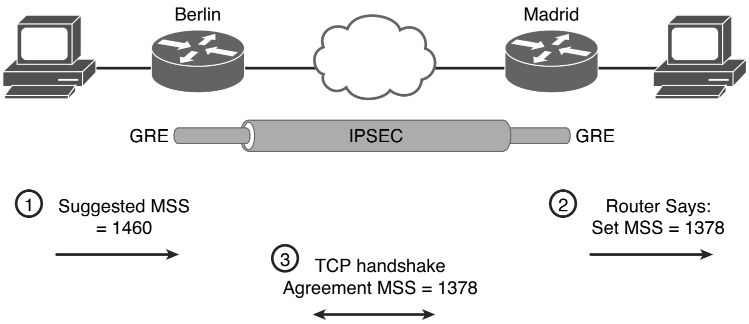

Berlin
Madrid
GRE
IPSEC
GRE
1 Suggested MSS = 1460
2 Router Says: Set MSS = 1378
3 TCP handshake Agreement MSS = 1378

Figure 36-9 *Adjusting the TCP Session MSS Value During the 3-Way TCP Handshake Process*

In Figure 36-9, a sending host begins the TCP session setup process by suggesting to the receiving host an MSS size of 1460 bytes. Instead of letting the receiving host reply with a positive acknowledgment of this (neither the sender nor the receiver know that traffic is passing through an IPSec/GRE tunnel and thus may encounter MTU fragmentation issues), the Cisco VPN router called Madrid intercepts the TCP session establishment and inserts a new suggested MSS of 1378 bytes in the session setup communication (the TCP SYN). Because this MSS value is guaranteed to equate to a total packet size that is less than 1500 bytes, it will allow traffic to pass the VPN tunnel without fragmentation. Also, because this MSS is a smaller value than what the receiver would typically reply with, the smaller value is agreed on and the TCP MSS is now established to be 1378 bytes for this session.

In this sense, the TCP adjust-MSS feature in IOS allows the router to be a middleman during the TCP setup process giving it the opportunity to override the MSS agreement between the two stations. By doing this, the router is able to influence the agreed upon

MSS so that in effect the router is really the one setting the connection MSS, thus preventing the MTU from ever being exceeded for TCP sessions that go across a VPN tunnel. Example 36-4 illustrates how to manually set the MSS for all TCP sessions going through a GRE tunnel on a Cisco router.

Example 36-4 *The TCP Adjust-MSS Feature Configuration*

```
interface Tunnel500
  ip address 192.168.1.0 255.255.255.252
  ip tcp adjust-mss 1378
  tunnel source 192.168.1.1
  tunnel destination 192.168.2
!
```

To understand where the MSS value of 1378 bytes comes from in this example, consider the following breakdown:

- Original IP packet = 1500 bytes
- Subtract 20 bytes for IP header = 1480 bytes
- Subtract 20 bytes for TCP header = 1460 bytes
- Subtract 24 bytes for GRE header = 1436 bytes
- Subtract a maximum of 58 bytes for IPsec = 1378 bytes

In this example, the ideal MSS to accommodate the GRE and IPsec overhead to avoid any packet fragmentation is 1378 bytes. It is common to add a little “fudge” room here and lower the MSS even more, but this decision can be left to the network engineer.

One important point to remember about adjusting the TCP MSS is that it is a CPU-intensive process for the router that could possibly have a significant impact on overall performance, thus severely impacting application performance. Because this could be such a performance-impacting feature, the question arises where to enable this feature.

Because adjusting the MSS can be done at any point in the path of a TCP connection setup, and only needs to be done in one place in the path, it is not necessary to configure this at both the headend and remote sites. In fact, it is generally recommended that this feature be enabled on the tunnel interface at the *remote* site only, not at the headend. The rationale for this is that if there are a large number of tunnels terminating at the headend site and possibly a large volume of TCP sessions, having the TCP adjust-MSS feature enabled on a single headend router may severely impact it is performance because every single new TCP session will require control plane processing. However, if this feature were distributed to the remote sites, the CPU-intensive task performed here would be spread among these remote routers, thus leaving the headend router unscathed by the CPU impacting nature of this feature.

It is also important to note that the TCP adjust-MSS feature applies only to TCP traffic, and has no effect on UDP traffic types. Although it would be very uncommon to see UDP traffic with large MTUs, the potential does exist. For example, while Cisco video codes do have the ability to customize the MTU, it is possible that the MTU could be set too large and thus impact video performance. As mentioned earlier, although various methods exist for dealing with the MTU issue over VPN links, it is preferable not to fragment IP packets, especially in the case of real-time traffic, such as video conferencing. As a best practice, it is recommended to check the MTU settings on all video-conferencing equipment before using it over a VPN tunneling technology.

Compression Strategies Over VPN

Recommendations:

- When possible, use a compression strategy to improve overall throughout, latency, and user experience on VPN connections.
- Be alert to WAN compression technologies that tunnel and hide the fields used for QoS classification.

When designing an IPsec VPN between two sites, various methods to reduce the overall amount of bandwidth are often considered. Data compression has the effect of increasing the actual real throughput to a value much higher than the theoretical bandwidth of the VPN link, in addition to greatly improving the latency issues that are inherent with VPNs over longer distances.

TCP Optimization Using WAAS

One key technology that greatly improves the user experience over a VPN WAN is the Cisco Wide Area Application Services (WAAS). WAAS uses a variety of compression technologies, such as LZ compression, date redundancy elimination (DRE), and specific application optimizers (AOs) that significantly reduce the amount of data that is sent across the WAN or VPN. The effect of WAN acceleration technologies like WAAS and WAAS Express (WAAS Express is an IOS feature set that does not require additional hardware or modules) is that it not only increases the effective bandwidth of the link but also dramatically improves the round-trip time (RTT) latency of the VPN.

For example, one real-world implementation of Cisco WAAS showed a computer software company located in North America that needed to backup 75 GB of data daily to a remote development office in India. The link to India was a T1 and suffered from a 400-ms RTT. Unfortunately for this company the daily backup would normally take up to 3 days. When WAAS was first implemented, the first backup dropped from 75 hours down to 15. Subsequent daily backups then dropped to 45 minutes, due to the WAAS capability to cache redundant data at the byte or “chunk” level. Naturally, for a technology like WAAS to benefit an encrypted link, the compression must take place before encryption.

Compression technologies like this can have a significant effect on the overall user QoE. One limitation of technologies like WAAS, however, is that the primary benefit is for TCP-based traffic only. One key point of consideration when using WAN compression technologies is to ensure that the WAN accelerators themselves do not break QoS. For example, with Cisco WAAS, only the data portion of the packet is compressed, thus leaving the original ToS byte visible and available for QoS classification. It has been noted that several WAN acceleration solutions available on the market today tunnel all traffic between the WAN accelerators, and thus obfuscate the original IP header, making QoS classification impossible.

Using Voice Codecs over a VPN Connection

WAN acceleration technologies, such as WAAS, are great tools for dealing with TCP traffic, but they are not able to deal with UDP-based applications, such as voice or video.

To improve overall voice capacity over a bandwidth-constrained VPN link, administrators often opt to use voice compression codecs, such as ILBC or G.729.

As a comparison, a typical G.711 codec will consume 80 Kbps per voice call, whereas if a G.729 codec is used, the required bandwidth is reduced to 24 Kbps. The effect of this compression strategy is that almost three times as many calls can be placed over the VPN link as compared to the G.711 codec, which uses no compression at all.

Although this might seem very attractive on the surface, care must be taken when using voice codes over VPN connections that use the raw Internet. Because G.729 functions to compress so much of the voice data into fewer packets, the impact of significant packet loss on an unreliable network, such as often occurs on the Internet, can be very noticeable. To overcome this challenge on lossy networks, Cisco's implementation of G.729 has a built-in protection mechanism to help cope with this problem.

Consider how G.729 can help to compensate for packet loss over a VPN link. Take the example of four compressed voice packets sent over a VPN link. Let's say that because of an unexpected network event one of the four packets was lost. In this case, the G.729 codec on the receiving end performs a special concealment strategy to hide the fact that one of the packets was lost. For example, if the third packet in a sequence of four is lost, the listening station waits for it until the jitter buffer expires. At this point, the codec replays the second packet in the sequence in an effort to hide the fact that the third packet was lost. What the codec is really trying to do is to trick the listener by concealing the gap between packets, even though the data that is replayed is not correct. Thanks to this concealment strategy, G.729 is able to tolerate up to 5 percent packet loss averaged across an entire call.[1]

In recent years, a new open source voice codec has emerged to address these issues. This is the Internet Low Bitrate Codec (iLBC), developed by Global IP Solutions (now part of Google), and defined in RFC 3951. iLBC offers either 15.2-Kbps or 13.33-Kbps bitrates, making it a very attractive option for lossy VPN links. The iLBC codec running at 15.2

[1] *Voice Over IP Fundamentals*, 2nd Edition, by Manoj Bhatia, Jonathan Davidson, Satish Kalidindi, Sudipto Mukherjee, James Peters

Kbps offers very similar mean opinion score (MOS) performance to G.729 with no packet loss. However, in the presence of packet loss (1 percent to 10 percent), iLBC performs significantly better. Figure 36-10 illustrates the comparison of various voice codes in the presence of packet loss.

iLBC is widely available in the current suite of Cisco IP phones and gateways.

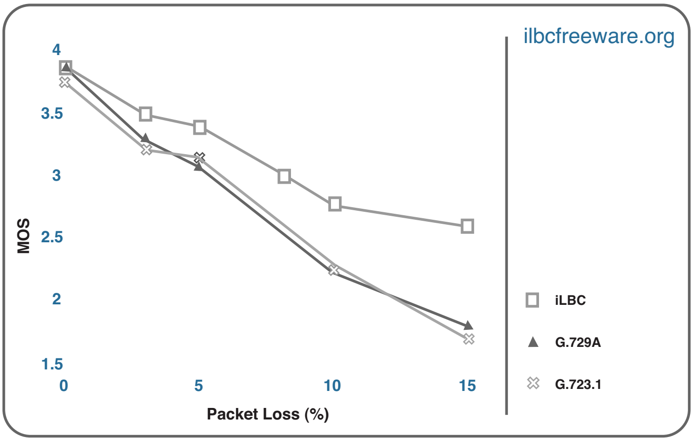


Figure 36-10 *MOS Score Versus Packet Loss for Various Codes Note (This diagram is provided under permission from Google.)*

cRTP and IPsec Incompatibilities

One of the key challenges faced when dealing with encryption is that most, if not the entire, original IP packet that is referenced for QoS (and other) purposes is no longer readable. If not for specific IOS features, such as already discussed in this chapter, QoS would be all but impossible on VPN link. One feature that relies on visibility into the packet is compressed Real-Time Protocol (cRTP), thus making cRTP and IPsec inherently incompatible networking tools.

Examining the order of operations in the IOS stack helps to explain this. Because the original RTP packet is encrypted by the time the compressor is called upon, it is impossible to compress the already encrypted packet. Therefore, because cRTP cannot associate the encrypted RTP packet with a known media stream, compression cannot occur and cRTP is not possible. The encrypted IP/UDP/RTP packet simply bypasses the compression process and continues uncompressed to the transmit queue.

It is also important to recognize that cRTP functions on a hop-by-hop basis, whereas IPsec usually spans multiple intermediate hops between IPsec endpoints. This distinction further exacerbates incompatibility between the features. Therefore, cRTP cannot be used to achieve bandwidth savings in an IPsec VPN environment.

Antireplay Implications

Recommendation:

- Understand the potential risk that antireplay drops may introduce in an IPsec VPN network with QoS enabled.

IPsec offers inherent message-integrity mechanisms to provide a means of identifying whether an individual packet is being replayed by an interceptor or hacker. This concept is called *connectionless integrity*. IPsec also provides partial sequence integrity, preventing the arrival of duplicate packets. These concepts are outlined in RFC 2401, *Security Architecture for the Internet Protocol*.

When ESP authentication (**esp-sha-hmac**) is configured in an IPsec transform set, for each security association the receiving IPsec peer verifies that packets are received only once. Because two IPsec peers can send millions of packets, a 64-packet sliding window is implemented to bound the amount of memory required to tally the receipt of a peer's packets. Packets can arrive out of order, but they must be received within the scope of the window to be accepted. If they arrive too late (outside the window), they are dropped.

The operation of the antireplay window protocol is as follows:

1. The sender assigns a unique sequence number (per security association) to encrypted packets.
2. The receiver maintains a 64-packet sliding window; the right edge of which includes the highest sequence number received. In addition, a Boolean variable is maintained to indicate whether each packet in the current window was received.
3. The receiver evaluates the received packet's sequence number:

   a If a received packet's sequence number falls within the window and was not received previously, the packet is accepted and marked as received.

   b If the received packet's sequence number falls within the window and previously was received, the packet is dropped and the replay error counter is incremented.

   c If the received packet's sequence number is greater than the highest sequence in the window, the packet is accepted and marked as received, and the sliding window is moved "to the right."

   d If the received packet's sequence number is less than the lowest sequence in the window, the packet is dropped and the replay error counter is incremented.

In a converged IPsec VPN implementation with QoS enabled, lower-priority packets are delayed so that higher-priority packets receive preferential treatment. This has the unfortunate side effect of reordering the packets to be out of sequence from an IPsec antireplay sequence number perspective. Therefore, there is a concern that through the normal QoS prioritization process the receiver might drop packets as antireplay errors, when, in fact, they are legitimately sent or received packets.

Figure 36-11 illustrates this process. In this example, voice packets 4 through 67 have been received, and data packet 3 was delayed and transmitted following voice packet 68. When the antireplay logic is called to process packet 3, it is dropped because it is outside the left edge of the sliding window. Packets can be received out of order, but they must fall within the window to be accepted.

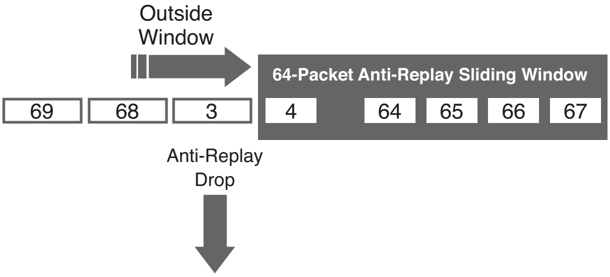


Figure 36-11 *Antireplay Operation*

Antireplay drops can be eliminated in a pure IPsec tunnel design (no encrypted IP GRE tunnel) by creating separate security associations for voice, video, and data. That is, voice, video, and data packets must match a separate line in the access list referenced by the crypto map. This is implemented easily if the IP phones are addressed by network addresses (such as private RFC 1918 addresses) separate from the workstations.

Consider the effect of packet loss on a TCP-based application: TCP is connection oriented and incorporates a flow-control mechanism within the protocol. The TCP application cannot see why a packet was dropped. From a sending host's perspective, a packet dropped by a service policy on a congested output interface is no different to the application than a packet lost by an antireplay drop. From a network perspective, however, it would be more efficient to drop the packet before sending it over the WAN link (where bandwidth is the most expensive, only to have it dropped by the antireplay mechanism on the receiving IPsec VPN router), but the location or nature of the packet loss is immaterial to the TCP driver.

Antireplay drops of data traffic flows are usually in the order of 1 percent to 1.5 percent on IPsec VPN links that experience sustained congestion and have queuing engaged (without any additional policy tuning).

Output drops on the output WAN interface, however, tend to be few, if any, and certainly are far fewer than those dropped by antireplay. This is because antireplay triggers packet drops more aggressively than the output service policy, which is a function of the size of the output queues and the number of defined classes.

By default, each CBWFQ class receives a queue with a length of 64 packets. This can be verified with the **show policy interface** verification command. Meanwhile, the receiving IPsec peer has a single 64-packet antireplay window (per IPsec security association) with which to process all packets from all low-latency queuing (LLQ) and class-based weighted fair queuing (CBWFQ) bandwidth classes.

So, it stands to reason that the antireplay mechanism on the receiving VPN router will be more aggressive at dropping packets delayed by QoS mechanisms preferential to VoIP than the service policy at the sending router. This is because of the size mismatch of the queue depth on the sender's output interface (multiple queues of 64 packets each) compared to the width of the receiver's antireplay window (a single sliding window of 64 packets per SA). As more bandwidth classes are defined in the policy map, this mismatch increases. As mentioned, this is an inefficient use of expensive WAN/VPN bandwidth because packets are transmitted only to be dropped before decryption.

The default value of 64 packets per CBWFQ queue is designed to absorb bursts of data traffic and delay rather than drop those packets. This is optimal behavior in a non-IPsec-enabled network.

Note Chapter 37, "DMVPN QoS Design," explores the subject of per-SA QoS design, particularly in a DMVPN deployment.

When IPsec authentication is configured (**esp-sha-hmac**) in the network, the scenario can be improved by reducing the queue limit (**max threshold**) of the bandwidth classes of the sender's output service policy so that it becomes more aggressive at dropping packets than buffering or delaying them. Extensive lab testing has shown that such queue-limit tuning can reduce the number of antireplay drops from 1 percent to less than a tenth of percent (< 0.1 percent). This is because decreasing the service policy queue limits causes the sender's output service policy to become more aggressive in dropping instead of significantly delaying packets (which occurs with large queue depths). This, in turn, decreases the number of antireplay drops. As a rule of thumb, the queue limits should be reduced in descending order of application priority.

Note In many networks, the default queue limits and IPsec antireplay performance are considered acceptable. A modification of queue-limit values entails side effects on the QoS service policy and related CPU performance. When queue limits are tuned, a careful eye should be kept on CPU levels.

Note IOS offers another potential remedy to QoS/IPsec antireplay issues with the ability to expand or disable the IPsec Anti-Replay window. However, it should be kept in mind that the IETF IPsec standards define the antireplay window-sizes to be 64 packets and therefore, this would not be a standards-compliant solution. However, the presented solution of tuning of the QoS queue limits is fully standards compliant.

Summary

This chapter focused on some of the challenges involved when deploying QoS in a VPN environment, specifically those faced in IPsec and GRE networks. The chapter began with a look at the various VPN deployment models and how these affect the visibility of the fields used for QoS classification. When DSCP-based classification is used, the default behavior of IOS to copy the ToS field to the new IPsec header. However, when classification needs to be done on other fields in the TCP/IP header, the QoS preclassify feature clones the original header, thus enabling classification based on the other fields in the original header.

When dealing with VPNs, it is always important to be aware of the extra packet headers that can lead to severe MTU fragmentation and reassembly issues. Furthermore, when designing VPNs the overall quality of experience can be greatly enhanced by the use of compression technologies, such as WAAS and the appropriate selection of voice and video codecs. It is critical to note that cRTP and IPsec are inherently incompatible networking tools. The IPsec antireplay technology also introduces several challenges as the QoS queuing mechanism may reorder packets, thus dropping traffic that appears out of order.

The following three chapters focus on how to deploy QoS in specific VPN topologies, including DMVPN, GETVPN, and the home teleworker scenario.

Additional Reading

Voice over IP Fundamentals, 2nd Edition, by Manoj Bhatia, Jonathan Davidson, Satish Kalidindi, Sudipto Mukherjee, James Peters

iLBC freeware: http://ilbcfreeware.org/

This page intentionally left blank

Chapter 37

DMVPN QoS Design

Dynamic Multipoint VPN (DMVPN) is a key IPsec use case for site-to-site VPN deployments. In days past, one of the largest challenges that network engineers and administrators faced was how to easily scale their virtual private network (VPN) deployment. For example, if static point-to-point IPsec tunnels were used between a hub and a large number of spoke routers, not only would the configuration of the headend router become massive and almost unmanageable, but the day-to-day administration of the network would also become difficult.

You can imagine a situation where a large retail chain might have several hundred or even several thousand stores connected to the head office or data center across a public cloud network where encryption is necessary. Not only does a situation like this require significant IPsec management, but it also requires a fairly sophisticated level of QoS. For example, this retail company may pass a variety of different traffic types from the branches back to the head office network, including voice, web, and credit card transactions. In a case like this, the credit card transactions obviously need to be transported securely, but they also very likely need to be transported with a high level of QoS because the ability to process these credit card transactions in a timely fashion is the very lifeblood of the company.

If you were faced with the challenge of manually configuring hundreds or even thousands of point-to-point tunnels, plus adding QoS policies to each tunnel at the headend router, the chance of configuration mistakes, typos, and just the sheer amount of bulk configuration required would make the job almost impossible. For example, statically configuring the VPN endpoints for remote sites requires that the each VPN tunnel interface be manually configured with IPsec encryption policies, security profiles, and quality of service (QoS) policies. Additionally, troubleshooting could easily become a nightmare.

DMVPN has the distinct advantage that it enables you to easily deploy large-scale VPNs at the headend and offers a powerful yet simple way to deploy QoS on a per-tunnel basis with minimal configuration. Another attractive aspect of DMVPN is that it allows the spokes to dynamically set up spoke-to-spoke communication with no extra manual configuration. The spoke simply communicates to the hub router, learns of the IP address used on the remote spoke it wants to communicate with, and a new VPN tunnel is dynamically established between the two spokes, meaning traffic does not need to loop through the hub router.

Whether you are looking to deploy a VPN with only a few remote sites, or thousands of sites, DMVPN is an appealing VPN technology.

One key architectural advantage of DMVPN is that is uses only a single multipoint generic routing encapsulation (mGRE) interface on the headend router, instead of having a multitude of unique and separate GRE tunnels connected to each spoke router. In traditional IPsec VPN environments, the only two options for deploying QoS were to either use a QoS policy map on the physical interface facing the WAN or to attach a separate QoS policy map on each GRE tunnel interface. In the first case, the policy map facing the WAN simply manages traffic for the whole egress traffic stream, but does not provide any per-spoke (or per-tunnel) traffic management. The second case is a better QoS solution, but it creates an enormous amount of configuration on the router and is simply not practical to deploy. For example, if a separate VPN tunnel were created on the hub router for each remote spoke, for QoS to work it would require a unique outbound QoS policy map on each GRE tunnel interface. This introduces a serious scalability concern because most routers limit the total number of policy and class maps that may be created on a system.

To summarize, the DMVPN architecture presents several advantages with respect to QoS:

- Reduction of overall hub router QoS configuration
- Scalability to thousands of sites, with QoS for each tunnel on the hub router
- Zero-touch QoS support on the hub router for new spokes
- Flexibility of both hub-and-spoke and spoke-to-spoke (full mesh) deployment models
- Wide support across most enterprise-class Cisco routers, such as the ISR G2 and ASR 1000 families

This chapter explores the following topics:

- The role of QoS in a DMVPN network
- DMVPN QoS architecture
- DMVPN QoS design example
- Comparing DMVPN and FlexVPN QoS models

The Role of QoS in a DMVPN Network

DMVPN is one of the most widely deployed Cisco VPN architectures today and has become the gold standard for site-to-site IPsec VPNs over untrusted networks, such as the Internet.

As mentioned in Chapter 36, "IPsec VPN QoS Considerations and Recommendations," a common requirement of VPNs is the need to support dynamic routing protocols, such as Open Shortest Path First (OSPF) and Enhanced Interior Gateway Routing Protocol (EIGRP), in addition to multicast. For this reason, encrypted GRE tunnels are often used for site-to-site VPNs. However, when GRE and IPsec are combined, the configuration complexity is increased, thus challenging the scalability of traditional static site-to-site VPN tunnels.

DMVPN Building Blocks

DMVPN combines mGRE and IPsec encryption with Next-Hop Resolution Protocol (NHRP) in a way that allows the simple and seamless deployment of tunnel endpoints and thus scales to thousands of spoke sites. The key components of DMVPN are as follows:

- **mGRE:** Instead of using a unique GRE tunnel interface for each tunnel endpoint, mGRE allows you to create a single GRE tunnel interface on the hub router that can serve a large number of remote spokes. (The scalability of the hub differs for each router platform.) Using the mGRE tunnel interface, you are thus able to apply a single outbound QoS policy instead of one on every static GRE tunnel endpoint.
- **Dynamic discovery of IPsec tunnel endpoints and crypto profiles:** This capability of DMVPN enables the dynamic creation of crypto maps, meaning that you do not have to statically build crypto maps for each tunnel endpoint. Similar to the advantages of QoS for DMVPNs, this greatly reduces the configuration requirements of IPsec.
- **NHRP:** This allows the spoke to be deployed with dynamically configured IP addresses, meaning that the remote router can pick up a Dynamic Host Configuration Protocol (DHCP) address from the local service provider and connect back to the hub router. NHRP also enables a zero-touch deployment model that makes DMVPN spokes easy to set up and manage. In the case of NHRP in a DMVPN topology, it is helpful to think of the hub router as a "next-hop server" rather than as a traditional VPN router. NHRP is also a crucial component of the per-tunnel QoS feature that is examined in this chapter. In this context, NHRP is the mechanism by which a spoke announces to the hub what QoS policy to use.

How QoS Is Implemented in a DMVPN

Combining the three building blocks just discussed, DMVPN can be quickly implemented to scale to hundreds or even thousands of remote nodes while providing QoS to each spoke.

Because DMVPN is part of the IP transit network, there is an implicit trust of the DSCP markings of the packets that are received on the LAN interface by both the hub and spoke VPN routers. Only if there were a very specific reason not to trust the markings would it be necessary for you to implement a re-marking strategy at this point in the network. Therefore, the primary QoS functions in a DMVPN topology are outbound hierarchical shaping and queuing of the specific traffic classes over the VPN tunnels.

Figure 37-1 illustrates where QoS is implemented in a simple DMVPN network.

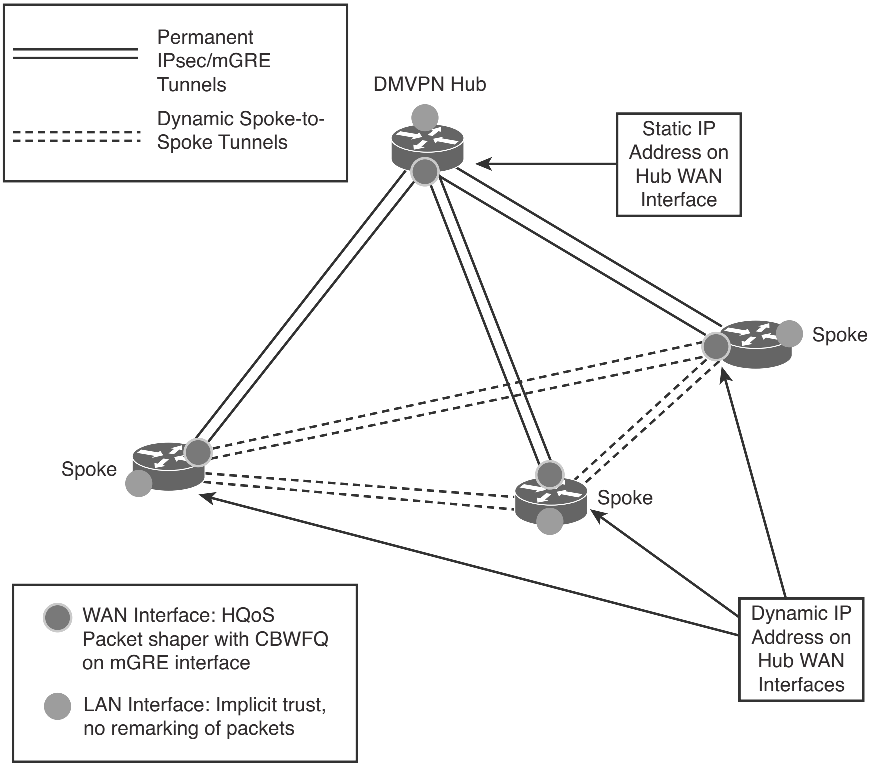

Permanent IPsec/mGRE Tunnels
Dynamic Spoke-to-Spoke Tunnels
DMVPN Hub
Static IP Address on Hub WAN Interface
Spoke
Spoke
Spoke
Dynamic IP Address on Hub WAN Interfaces
WAN Interface: HQoS Packet shaper with CBWFQ on mGRE interface
LAN Interface: Implicit trust, no remarking of packets

Figure 37-1 *QoS over a Simple DMVPN Network Using IPsec and mGRE with Three Remote Sites*

DMVPN QoS Configuration

Because DMVPN relies on mGRE tunnel interfaces, traditional QoS policies at the interface level do not work. To solve this problem, the per-tunnel QoS for DMVPN feature is used to provide QoS for each and every tunnel terminating on the mGRE tunnel. Just as DMVPN itself relies on NHRP for endpoint discovery and routing, the per-tunnel QoS for DMVPN feature also greatly leverages NHRP. This is the subject of the following section.

Next-Hop Routing Protocol

Of the three design principles in the preceding list, the feature that allows for the creation of a dynamic full mesh of tunnels between spokes and also provides per-tunnel QoS is NHRP. As shown in Figure 37-2, when the DMVPN is first created, each spoke router has a permanent IPsec tunnel to the hub router. However, when necessary, each spoke can also create a dynamic IPsec tunnel to any other spoke router as well. This has the distinct advantage of eliminating the needless turnaround of traffic at the hub site when one spoke needs to communicate with another, thus both improving performance on the hub router and decreasing the round-trip time (RTT) between spokes.

To support this functionality, NHRP on the hub router maintains a database of the real WAN IP addresses of each spoke router. When a new spoke router joins the network and connects to the hub router via an encrypted mGRE tunnel, it registers its real WAN IP address with the NHRP database on the hub router.

Now, suppose you are on a network at one of the remote spokes and you want to make a Voice over IP (VoIP) call to a phone at another remote branch. Ideally, you want your phone call data path to go from spoke to spoke, rather than doing a hairpin from one VPN tunnel into another at the hub router. To facilitate this, the spoke router at the first location first queries the hub router's NHRP database to discover the WAN address of the destination spoke router at the other end of your phone call.

When this information is returned to the spoke router, it is able to dynamically build a new encrypted mGRE tunnel to the other spoke router and VoIP data will now bypass the hub router and go direct from spoke to spoke.

Figure 37-2 illustrates this process.

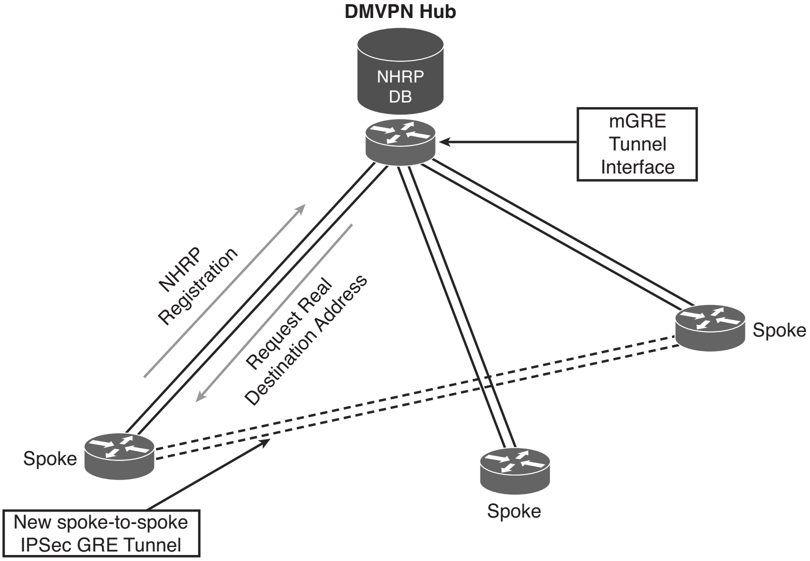


Figure 37-2 *NHRP in a DMVPN*

From a QoS perspective, NHRP also plays a key role. When a spoke router goes through the NHRP registration process, it also announces to the DMVPN hub router what NHRP group it belongs to, and thus in turn allows the hub router to apply the correct outbound QoS policy to each spoke router.

The Need for a Different Approach to QoS in DMVPNs

One of the challenges with applying QoS to a DMVPN environment is that all the hub-and-spoke tunnels terminate on a single mGRE tunnel interface on the hub router. With more traditional IPsec and GRE deployments, we can deploy a QoS policy directly onto each tunnel interface and apply it as traffic flows outbound through each tunnel, as illustrated in Figure 37-3. This has the effect of providing a per-site QoS policy.

What about just applying a single all-inclusive outbound QoS policy on the WAN-facing interface of the VPN router instead of on each tunnel interface? Although this approach might ensure that the overall WAN pipe is meeting your QoS policy, there would be no way to ensure that the traffic going to each spoke was meeting the QoS requirements. The negative result of this might mean that a high volume of traffic going to some spokes could far exceed the traffic going to other spokes, in effect starving mission-critical traffic from getting through to certain spokes.

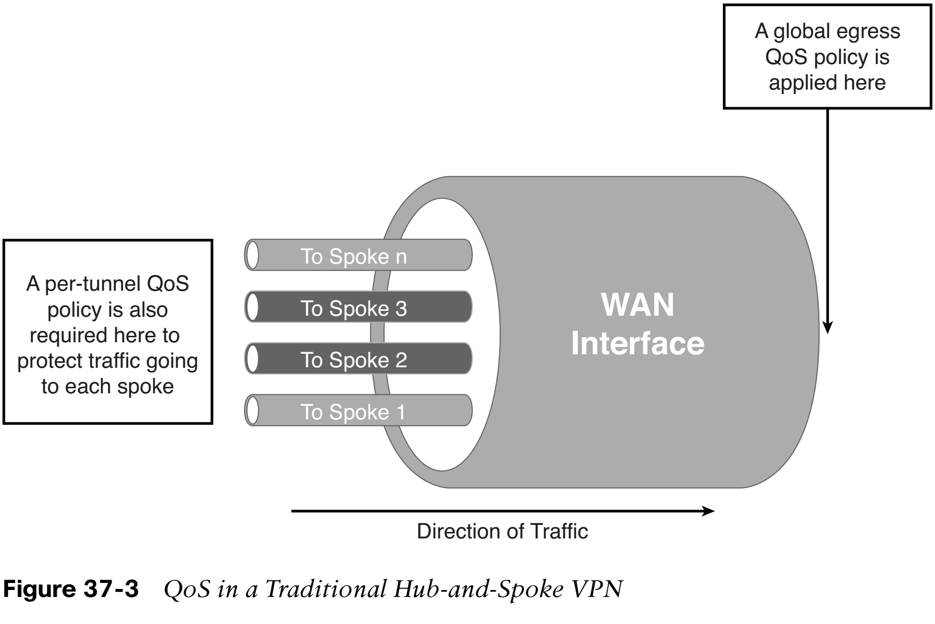

A global egress QoS policy is applied here
A per-tunnel QoS policy is also required here to protect traffic going to each spoke
To Spoke n
To Spoke 3
To Spoke 2
To Spoke 1
WAN Interface
Direction of Traffic

Figure 37-3 *QoS in a Traditional Hub-and-Spoke VPN*

While applying a unique QoS policy per tunnel is possible with traditional site-to-site VPNs if separate GRE tunnel interfaces are used per spoke, with DMVPN there is only one single mGRE interface that all the spoke tunnels terminate on, meaning that we cannot easily attach a per-site QoS policy to this interface. So although DMVPN indeed solved some problems, it appears to have created another one. To address this challenge, and to enable a proper implementation of QoS in DMVPN environments on a per-tunnel basis, Cisco developed a new feature called *per-tunnel QoS for DMVPN*.

The Per-Tunnel QoS for DMVPN Feature

As the name indicates, the *per-tunnel QoS for DMVPN* feature allows you to enable QoS on a per-tunnel or per-spoke basis when using DMVPN. This feature allows you to apply an outbound QoS service policy on the mGRE tunnel interface on the DMVPN hub, which is applied by the router to each spoke tunnel that the service policy is associated with.

One effect of this approach is that it protects your spokes from each other. For example, certain spokes may tend to hog bandwidth and pass so much data that the less-busy spokes are starved for bandwidth. Another benefit of this feature is that it can limit the hub router from sending too much data toward a smaller bandwidth spoke and overrunning it.

This feature has another key advantage in that it provides a certain amount of automation to the system. For example, the QoS policy at the DMVPN hub is generated automatically for each tunnel as the spokes register with the hub, thus greatly reducing manual configuration of QoS on a per-spoke basis.

As discussed numerous times throughout this book, queuing mechanisms kick in only when there is actual congestion on the egress interface. With DMVPN tunnels all terminating on a single virtual mGRE interface, it is necessary to somehow signal to the router's QoS mechanism that there is congestion on a particular VPN tunnel, even if the mGRE interface is passing a relatively small amount of traffic. As when dealing with other cases of a mismatch between the router's physical interface speed and the offered WAN link speed, you first need to shape the traffic flows to the real VPN tunnel bandwidth to produce artificial backpressure. With per-tunnel QoS for DMVPN, a shaper is automatically applied by the system to each and every tunnel. This in turn allows the router to implement differentiated services for the various data flows corresponding to each tunnel. This technique is called Hierarchical Queuing Framework (HQF).

Using HQF, the QoS policy applied to the mGRE interface on the DMVPN hub allows you to shape the tunnel traffic to each spoke (the parent policy) and then to apply differentiate services for the various data flows going through that tunnel with class-based weighted fair queuing (CBWFQ) (the child policy).

Using NHRP, multiple spokes can be grouped together and have the same QoS policy applied to all spokes in the group. Although there may be multiple spokes in the group, this feature applies the traffic shaper individually to each tunnel so that if any one tunnel exceeds its allowed bandwidth, backpressure can be artificially applied, and the child policy can come into effect, as illustrated in Figure 37-4.

Note that when this feature is enabled, QoS features such as queuing and shaping are actually performed at the outbound physical interface. This feature takes into account the extra GRE and IPsec headers so that they are included in the maximum transmission unit (MTU) calculations for shaping and queuing of packets at the outbound interface.

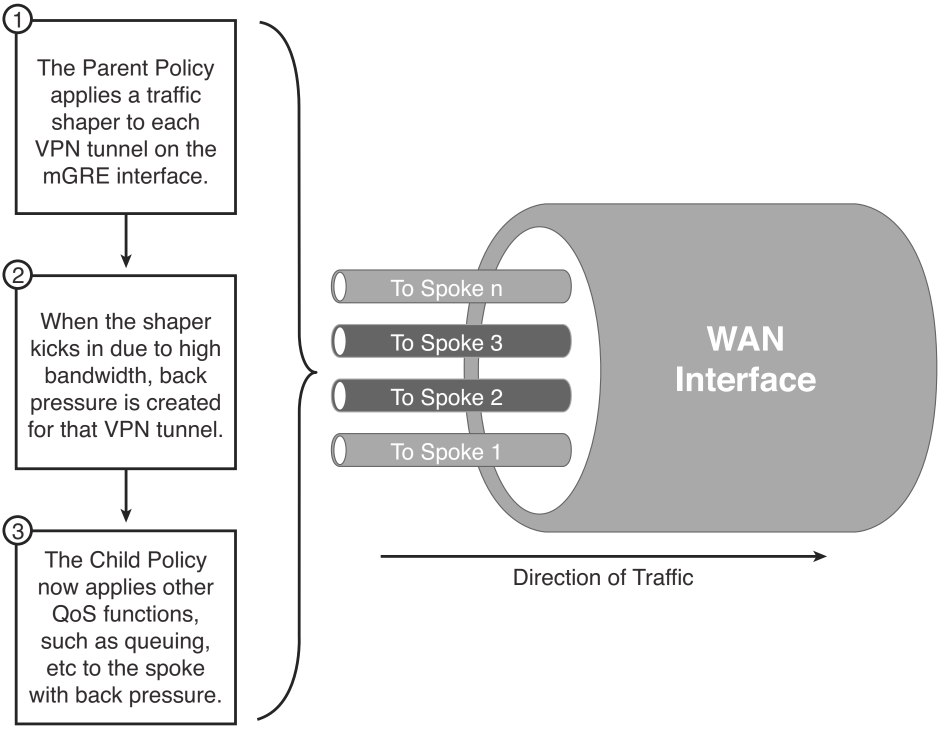

1
The Parent Policy applies a traffic shaper to each VPN tunnel on the mGRE interface.
2
When the shaper kicks in due to high bandwidth, back pressure is created for that VPN tunnel.
3
The Child Policy now applies other QoS functions, such as queuing, etc to the spoke with back pressure.
To Spoke n
To Spoke 3
To Spoke 2
To Spoke 1
WAN Interface
Direction of Traffic

Figure 37-4 *Per-Tunnel QoS for DMVPN in Action*

DMVPN QoS Design Example

As mentioned previously in this chapter, the traditional approach of applying a QoS service policy to a GRE tunnel interface does not work with mGRE since multiple tunnels are terminating on a single logical interface. The per-tunnel QoS for DMVPN feature is unique in that it allows different polices to be applied to different NHRP groups so that various QoS models can be implemented, such as the 4-, 8-, and 12-class models.

The per-tunnel QoS service policy is applied to the mGRE tunnel interface through an NHRP group. The NHRP group functionality was specifically introduced into DMVPN to support per-tunnel QoS and is the entity that is signaled by a DMVPN spoke when contacting the hub. This feature is supported both in IOS and in IOS XE for the ASR 1000 family. The DMVPN hub uses the information from the NHRP group to apply the correct QoS policy for the spoke in question.

Instead of building a long list of access control lists (ACLs) to identify which spoke belongs in which group (which would be a hard thing to do on the hub, because the spokes are often dynamically addressed), you identify the spoke by configuring the NHRP group membership on the spoke, which is in turn announced to the hub router and stored in its NHRP database. This has a huge advantage that when the spoke router registers with the hub, it is in effect telling the hub router what NHRP group it should be a part of. The hub router in turn updates the NHRP database with this information and can apply the correct QoS policy as traffic leaves the hub. So, although the spoke routers are responsible for telling the hub which group they are a part of, it is the hub router that does the actual NHRP group to QoS policy mappings.

Note A spoke can belong to only one NHRP group per GRE interface. If a spoke is part of two or more DMVPN networks (such as a high-availability design), the spoke can have different NHRP group names on each GRE tunnel interface.

Note If an NHRP group string is not received from the spoke, a QoS policy is not applied to the spoke, and any existing QoS policy applied to that spoke is removed.

Note The technique presented here applies QoS only from the hub in the egress direction toward the spoke. If QoS from the spoke toward the hub is needed, one of the standard three QoS reference designs (4-, 8, and 12-class model) can be used in the egress direction at each spoke.

Consider the design example outlined in Table 37-1 of a large DMVPN network that has three NHRP groups (meaning, three classes of spokes). The first is a group of spokes that are connected via 1.5 Mbps and have some basic applications, such as VoIP and miscellaneous data. The second group of spokes is given a larger bandwidth of 10 Mbps per site, and additionally run some multimedia conferencing and streaming. The third group of spokes is given a much larger bandwidth of 50 Mbps per site and additionally has some broadcast video applications. Table 37-1 summarizes the requirements for the three types of spokes in your network.

Table 37-1 *Design Example: Three Types of Spokes*

Type of Spoke	Bandwidth per Spoke	Types of Applications in Use	QoS Model
SPOKE-1-GROUP	1.5 Mbps	Voice Signaling Data	4-class model

Type of Spoke	Bandwidth per Spoke	Types of Applications in Use	QoS Model
SPOKE-2-GROUP	10 Mbps	Voice Data Video conferencing	8- class model
SPOKE-3-GROUP	50 Mbps	Voice Data Streaming media Video conferencing Broadcast video And so on	12 -class model

DMVPN QoS Design Steps

The steps for configuring QoS on both ends of a DMVPN tunnel are summarized as follows:

Hub Routers

1. Configure the child policy.
2. Configure the parent policy.
3. Attach the parent policy to tunnel interface.
4. Add the NHRP group membership to the interface.

Spoke Routers

1. Add the spoke to the NHRP group.
2. Add a 4-, 8-, or 12-class QoS policy to the tunnel interface.

These steps are detailed in the following sections.

Configuring the Hub Router for Per-Tunnel QoS

The first thing you need to do on the hub router is to configure the QoS policies before they can be added to the mGRE tunnel interface. Because there are three types of spokes (meaning three NHRP groups) in this design example, we first need to add the QoS classification that will be used for all of these three types of QoS model (4-, 8-, or 12-class models).

In most of the examples shown in the preceding chapters, only one of the 4-, 8-, or 12-class QoS models on any particular router or switch was used. However, with DMVPN, the situation differs somewhat. Because you can have so many different types of spokes connecting back to a single hub router, and because these may all have different QoS requirements, there is a need to set up QoS policies for all three models on one single router, which is why they are all shown together in this chapter. Therefore, on the hub router, you will have both the class and policy maps that are required to support any or all of these models. This example explores the case of a DMVPN network that has a collection of spokes that use all three standard QoS models.

The following sections show how you can build the policy maps for the various classes of service.

Configuring the Hub Router for the Four-Class QoS Model

The following example illustrates how to configure the DMVPN hub router for the 4-classs model. As with the previous examples, it is understood that the appropriate class maps have already been configured on the hub router (as defined in the Chapter 28, "WAN Aggregator (Cisco ASR 1000) QoS Design").

The first step in the configuration demonstrated in Example 37-1 is to build the child policy. As with normal QoS policy configurations, if you try to configure the parent policy map first, the router will reject the configuration because at this point the child policy map is unknown.

Example 37-1 *Configuring the Child QoS Policy Map for 4-Class Model Spokes*

```
ASR1000(config)# policy-map 4-CLASS-CHILD
 ! This policy map is a "child" of the parent policy map that is
 ! responsible for shaping all four-class spokes. When the shaper
 ! creates artificial backpressure, this policy map is invoked by
 ! the router.
ASR1000(config-pmap)# class REALTIME
ASR1000(config-pmap-c)# priority percent 33
ASR1000(config-pmap)# class CONTROL
ASR1000(config-pmap-c)# bandwidth percent 7
ASR1000(config-pmap)# class TRANSACTIONAL-DATA
ASR1000(config-pmap-c)# bandwidth percent 35
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap)# class class-default
ASR1000(config-pmap-c)# bandwidth percent 25
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect
```

Note The hub router configuration shown in this chapter was built and tested on an ASR 1000 running IOS XE 3.6, and the spoke routers are ISR 1900s. If an ISR G2 router were to be used as the hub, the configuration would be essentially the same as the ASR.

You can verify the configuration in Example 37-1 with the following commands:

- **show class-map**
- **show policy-map**

After the child policy map has been configured, you can create the upper-level parent policy map, which is primarily responsible for shaping the traffic into each VPN tunnel, as demonstrated in Example 37-2. This shaper creates the artificial backpressure that invokes the queuing model from Example 37-1.

Example 37-2 *Configuring the Parent QoS Policy Map for 4-Class Model Spokes*

```
ASR1000(config)# policy-map 4-CLASS-PARENT
ASR1000(config-pmap)# class class-default
ASR1000(config-pmap)# shape average 1500000
 ! Shaper to create queuing backpressure at 1.5 Mbps
ASR1000(config-pmap)# service-policy 4-CLASS-CHILD
 ! Nested service policy to call the "child" policy
```

You can verify the configuration in Example 37-2 with the following commands:

- **show class-map**
- **show policy-map**

After this, the parent QoS policy needs to be attached to the mGRE interface on the hub router, as demonstrated in Example 37-3.

Example 37-3 *Attaching the Parent QoS Policy Map to the mGRE Interface*

```
ASR1000(config)# interface Tunnel10
ASR1000(config-if)# ip address 10.1.1.254 255.255.255.0
 ! If using EIGRP, make sure to disable split horizon
ASR1000(config-if)# no ip split-horizon eigrp 100
ASR1000(config-if)# ip nhrp authentication hector
ASR1000(config-if)# ip nhrp map multicast dynamic
ASR1000(config-if)# ip nhrp map group SPOKE-1-GROUP service-policy output 4-CLASS-
PARENT
 ! Attach the parent service policy to the mGRE interface
 ! and associates this policy with NHRP SPOKE-1-GROUP
```

```
ASR1000(config-if)# ip nhrp network-id 12300
ASR1000(config-if)# tunnel source GigabitEthernet0/0/0
ASR1000(config-if)# tunnel mode gre multipoint
ASR1000(config-if)# tunnel key 3210
ASR1000(config-if)# tunnel protection IPsec profile MY-BIG-VPN
```

You can verify the configuration in Example 37-3 with the following commands:

- **show run interface tunnel10**
- **show ip nhrp group-map**
- **show dmvpn detail**

Note You might have noticed that this example also includes a line that disables EIGRP split horizon. Although it is your choice which routing protocol you will use inside the tunnels, thanks to its inherent stability in these kinds of environments, it is common to see Enhanced Interior Gateway Routing Protocol (EIGRP) in a hub-and-spoke topology. In fact, generally speaking, EIGRP tends to be much more stable than OSPF in hub-and-spoke designs

The nature of OSPF is that it floods updates (called link-state acknowledgements [LSAs]) every time a new route is learned. You can imagine a situation in a large hub-and-spoke VPN environment where hundreds of spoke sites are connected across the Internet. In a situation like this, if even one spoke were to lose temporary link status, the OSPF process for the whole DMVPN would generate an update LSA and flood it to every other router in the same area. If the condition of the network was poor, the situation could potentially become unmanageable, where LSAs are constantly being flooded to all the spoke routers, sending their CPUs through the roof as they try to absorb these new LSAs and compute new shortest-path first (SPF) paths.

EIGRP simply does not have this problem. Because EIGRP is not based on the link-state algorithm that Open Shortest Path First (OSPF) Protocol and Intermediate System-to-Intermediate System (IS-IS) Protocol use, it is usually much better suited to DMVPN environments and can easily scale to thousands of spokes, whereas OSPF would have difficulty scaling to anywhere near this number.

Configuring the Hub Router for the Eight-Class QoS Model

Examples 37-4 through 37-6 show how the hub router may be configured for the 8-class QoS model. As before, it is assumed the appropriate class maps have been configured.

Example 37-4 *Configuring the 8-Class Model Child QoS Policy Map*

```
ASR1000(config )# policy-map 8-CLASS-CHILD
 ! This policy map is a "child" of the parent policy map that is
 ! responsible for shaping all eight-class spokes. When the shaper
 ! creates artificial backpressure, this policy map is invoked by
 ! the router.
ASR1000(config-pmap)# class VOICE
ASR1000(config-pmap-c)# priority percent 10
ASR1000(config-pmap)# class MULTIMEDIA-CONFERENCING
ASR1000(config-pmap-c)# priority percent 23
ASR1000(config-pmap)# class NETWORK-CONTROL
ASR1000(config-pmap-c)# bandwidth percent 5
ASR1000(config-pmap)# class SIGNALING
ASR1000(config-pmap-c)# bandwidth percent 2
ASR1000(config-pmap)# class MULTIMEDIA-STREAMING
ASR1000(config-pmap-c)# bandwidth percent 10
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect dscp-based
ASR1000(config-pmap)# class TRANSACTIONAL-DATA
ASR1000(config-pmap-c)# bandwidth percent 24
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect dscp-based
ASR1000(config-pmap)# class SCAVENGER
ASR1000(config-pmap-c)# bandwidth percent 1
ASR1000(config-pmap)# class class-default
ASR1000(config-pmap-c)# bandwidth percent 25
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect
```

You can verify the configuration in Example 37-4 with the following commands:

- **show class-map**
- **show policy-map**

As before, you can now create the parent QoS policy map as demonstrated in Example 37-5.

Example 37-5 *Configuring the 8-Class Model Parent QoS Policy Map*

```
ASR1000(config)# policy-map 8-CLASS-PARENT
ASR1000(config-pmap)# class class-default
ASR1000(config-pmap-c)# shape average 10000000
 ! Shaper to create queuing backpressure at 10 Mbps
```

```
ASR1000(config-pmap-c)# service-policy 8-CLASS-CHILD
 ! Nested service policy to call the "child" policy
```

You can verify the configuration in Example 37-5 with the following commands:

- **show class-map**
- **show policy-map**

Finally, attach the parent policy map to the mGRE interface as demonstrated in Example 37-6. For simplification purposes, only the relevant QoS configuration is shown in this example.

Example 37-6 *Attaching the 8-Class Model Parent QoS Policy Map to the mGRE Interface*

```
ASR1000(config)# interface Tunnel10
 ! Only a partial configuration of the tunnel interface is shown
 ! here. The rest of the tunnel configuration is the same as the
 ! four-class model hub example shown earlier.
ASR1000(config-if)# ip address 10.1.1.254 255.255.255.0
ASR1000(config-if)# ip nhrp map group SPOKE-2-GROUP service-policy output 8-CLASS-
PARENT
 ! Attach the parent service policy to the mGRE interface
 ! and associate this policy with NHRP SPOKE-2-GROUP
```

You can verify the configuration in Example 37-6 with the following commands:

- **show run interface tunnel10**
- **show ip nhrp group-map**
- **show dmvpn detail**

Configuring the Hub Router for the Twelve-Class QoS Model

Examples 37-7 through 37-10 show how the hub router may be configured for the 12-class QoS model. As before, it is assumed that the appropriate class maps have been configured.

Example 37-7 *Configuring the 12-Class Model Child QoS Policy Map*

```
ASR1000(config)# policy-map 12-CLASS-CHILD
 ! This policy map is a "child" of the parent policy map that is
   responsible for shaping all 12-class spokes. When the shaper
   creates artificial backpressure, this policy map is invoked by
   the router.
ASR1000(config-pmap)# class VOICE
ASR1000(config-pmap-c)# priority percent 10
ASR1000(config-pmap)# class REALTIME-INTERACTIVE
ASR1000(config-pmap-c)# priority percent 13
ASR1000(config-pmap)# class BROADCAST-VIDEO
ASR1000(config-pmap-c)# priority percent 10
ASR1000(config-pmap)# class NETWORK-CONTROL
ASR1000(config-pmap-c)# bandwidth percent 2
ASR1000(config-pmap)# class SIGNALING
ASR1000(config-pmap-c)# bandwidth percent 2
ASR1000(config-pmap)# class NETWORK-MANAGEMENT
ASR1000(config-pmap-c)# bandwidth percent 3
ASR1000(config-pmap)# class MULTIMEDIA-CONFERENCING
ASR1000(config-pmap-c)# bandwidth percent 10
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect dscp-based
ASR1000(config-pmap)# class MULTIMEDIA-STREAMING
ASR1000(config-pmap-c)# bandwidth percent 10
ASR1000(config-pmap-c)# random-detect dscp-based
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap)# class TRANSACTIONAL-DATA
ASR1000(config-pmap-c)# bandwidth percent 10
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect dscp-based
ASR1000(config-pmap)# class BULK-DATA
ASR1000(config-pmap-c)# bandwidth percent 4
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect dscp-based
ASR1000(config-pmap)# class SCAVENGER
ASR1000(config-pmap-c)# bandwidth percent 1
ASR1000(config-pmap)# class class-default
ASR1000(config-pmap-c)# bandwidth percent 25
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect
```

You can verify the configuration in Example 37-7 with the following commands:

- **show class-map**
- **show policy-map**

Example 37-8 demonstrates the configuration of the parent policy map.

Example 37-8 *Configuring the 12-Class Model QoS Parent Policy Maps for Group 3 Spokes*

```
ASR1000(config)# policy-map 12-CLASS-PARENT
ASR1000(config-pmap)# class class-default
! create backpressure at 50 Mbps
ASR1000(config-pmap-c)# shape average 50000000
! use a nested service policy to call the "child" policy
ASR1000(config-pmap-c)# service-policy 12-CLASS-CHILD
```

You can verify the configuration in Example 37-8 with the following commands:

- **show class-map**
- **show policy-map**

As before, the final step is to now attach the above service policy to the mGRE tunnel interface. Example 37-9 highlights the configuration required on the DMVPN hub router to enable per-tunnel QoS for spokes in Group 3.

Example 37-9 *Attaching the 12-Class Model Parent QoS Policy Map to the mGRE Interface*

```
ASR1000(config)# interface Tunnel10
 ! Only a partial configuration of the tunnel interface is shown
 ! here. The rest of the tunnel configuration is the same as the
 ! four-class model hub example shown earlier.
ASR1000(config-if)# ip address 10.1.1.254 255.255.255.0
ASR1000(config-if)# ip nhrp map group SPOKE-3-GROUP service-policy output 12-CLASS-
PARENT
 ! Attach the parent service policy to the mGRE interface
 ! and associates this policy with NHRP SPOKE-3-GROUP
```

You can verify the configuration in Example 37-9 with the following commands:

- **show run interface tunnel10**
- **show ip nhrp group-map**
- **show dmvpn detail**

One of the unique features of DMVPN is its ability to support multiple QoS service policies on a single interface. It is an interesting feature because adding multiple service policies on a single interface is not supported in any other type of situation. In the design example that has been discussed so far there are three different kinds of spokes—spokes with 1.5 Mbps, 10 Mbps, and 50 Mbps. This doesn't require that three different mGRE interfaces be configured, rather, DMVPN allows you to configure multiple policy maps on the same interface. Example 37-10 shows how you might configure all three parent policy maps on a single mGRE interface on the hub router.

As before, only the relevant QoS configuration is shown in the following example.

Example 37-10 *Adding All Three Policy Maps to a Single mGRE Interface*

```
ASR1000(config)# interface Tunnel10
ASR1000(config-if)# ip address 10.1.1.254 255.255.255.0
ASR1000(config-if)# ip nhrp map group SPOKE-1-GROUP service-policy output 4-CLASS-PARENT
 ! Adds the SPOKE-1-GROUP spokes (4-Class model spokes)
ASR1000(config-if)# ip nhrp map group SPOKE-2-GROUP service-policy output 8-CLASS-PARENT
 ! Adds the SPOKE-2-GROUP spokes (8-Class model spokes)
ASR1000(config-if)# ip nhrp map group SPOKE-3-GROUP service-policy output 12-CLASS-PARENT
 ! Adds the SPOKE-3-GROUP spokes (12-Class model spokes)
```

You can verify the configuration in Example 37-10 with the following commands:

- **show run interface tunnel 10**
- **show ip nhrp group-map**
- **show dmvpn detail**

Configuring the Spoke Routers for Per-Tunnel QoS

The first required step to configure the necessary NHRP groups on the various spokes. Once this configuration is entered, the spoke will communicate with the hub router to tell it that it belongs to this particular NHRP group. Examples 37-11 through 37-13 show one spoke from each group with the NHRP group membership added. One critical aspect of this configuration is that the name of the NHRP group on the spoke must match the name used in the QoS service policy on the hub router. When NHRP registration takes place, the hub router stores this NHRP group name in the NHRP database, which is then in turn used to identify which service policy to apply to each spoke.

Note In the following examples, it is assumed that the appropriate class maps have already been configured for the 4-, 8-, and 12-class models that have been outlined in this book.

Example 37-11 shows how a spoke is able to announce its membership as part of Group 1. This configuration line is added to the mGRE interface under the **ip nhrp** syntax.

Example 37-11 *A Spoke with Is in Group 1 (Part of the Four-Class Model)*

```
! Spoke 1 (part of SPOKE-1-GROUP):
ISR1941_SPOKE_1(config)# interface tunnel10
ISR1941_SPOKE_1(config-if)# ip address 10.1.1.30 255.255.255.0
ISR1941_SPOKE_1(config-if)# ip nhrp authentication hector
ISR1941_SPOKE_1(config-if)# ip nhrp group SPOKE-1-GROUP
 ! Associates this spoke with NHRP SPOKE-1-GROUP
ISR1941_SPOKE_1(config-if)# ip nhrp map 10.1.1.254 172.25.1.254
ISR1941_SPOKE_1(config-if)# ip nhrp map multicast 172.25.1.254
ISR1941_SPOKE_1(config-if)# ip nhrp network-id 12300
ISR1941_SPOKE_1(config-if)# ip nhrp nhs 10.1.1.254
ISR1941_SPOKE_1(config-if)# ip tcp adjust-mss 1360
 ! Enable this command on all the spoke routers, not the hub
ISR1941_SPOKE_1(config-if)# tunnel source GigabitEthernet0/0
ISR1941_SPOKE_1(config-if)# tunnel mode gre multipoint
ISR1941_SPOKE_1(config-if)# tunnel key 3210
ISR1941_SPOKE_1(config-if)# tunnel protection IPsec profile MY-BIG-VPN
```

You can verify the configuration in Example 37-11 with the following commands:

- **show run interface tunnel 10**
- **show ip nhrp group-map**
- **show dmvpn detail**

Example 37-12 *A Spoke That Is in Group 2 (Part of the Eight-Class Model)*

```
! Spoke 2 (part of SPOKE-2-GROUP):
ISR1941_SPOKE_2(config)# interface tunnel10
ISR1941_SPOKE_2(config-if)# ip address 10.1.1.40 255.255.255.0
ISR1941_SPOKE_2(config-if)# ip nhrp authentication hector !
ISR1941_SPOKE_2(config-if)# ip nhrp group SPOKE-2-GROUP
 ! Associates this spoke with NHRP SPOKE-2-GROUP
ISR1941_SPOKE_2(config-if)# ip nhrp map 10.1.1.254 172.25.1.254
ISR1941_SPOKE_2(config-if)# ip nhrp map multicast 172.25.1.254
ISR1941_SPOKE_2(config-if)# ip nhrp network-id 12300
ISR1941_SPOKE_2(config-if)# ip nhrp nhs 10.1.1.254
ISR1941_SPOKE_2(config-if)# ip tcp adjust-mss 1360
 ! Enable this command on all the spoke routers, not the hub
ISR1941_SPOKE_2(config-if)# tunnel source GigabitEthernet0/0
ISR1941_SPOKE_2(config-if)# tunnel mode gre multipoint
ISR1941_SPOKE_2(config-if)# tunnel key 3210
ISR1941_SPOKE_2(config-if)# tunnel protection IPsec profile MY-BIG-VPN
```

You can verify the configuration in Example 37-12 with the following commands:

- **show run interface tunnel 10**
- **show ip nhrp group-map**
- **show dmvpn detail**

Example 37-13 *A Spoke with That Is in Group 3 (Part of 12-Class Model)*

```
! Spoke 3 (part of SPOKE-3-GROUP):
ISR1941_SPOKE_3(config)# interface tunnel10
ISR1941_SPOKE_3(config-if)# ip address 10.1.1.50 255.255.255.0
ISR1941_SPOKE_3(config-if)# ip nhrp authentication hector
ISR1941_SPOKE_3(config-if)# ip nhrp group SPOKE-3-GROUP
 ! Associates this spoke with NHRP SPOKE-1-GROUP
ISR1941_SPOKE_3(config-if)# ip nhrp map 10.1.1.254 172.25.1.254
ISR1941_SPOKE_3(config-if)# ip nhrp map multicast 172.25.1.254
ISR1941_SPOKE_3(config-if)# ip nhrp network-id 12300
ISR1941_SPOKE_3(config-if)# ip nhrp nhs 10.1.1.254
ISR1941_SPOKE_3(config-if)# ip tcp adjust-mss 1360
 ! Enable this command on all the spoke routers, not the hub
ISR1941_SPOKE_3(config-if)# tunnel source GigabitEthernet0/0
ISR1941_SPOKE_3(config-if)# tunnel mode gre multipoint
ISR1941_SPOKE_3(config-if)# tunnel key 3210
ISR1941_SPOKE_3(config-if)# tunnel protection IPsec profile MY-BIG-VPN
```

You can verify the configuration in Example 37-13 with the following commands:

- **show run interface tunnel10**
- **show ip nhrp group-map**
- **show dmvpn detail**
- **show policy-map multipoint**

Note in the preceding examples that the only special configuration needed on the spoke routers is to identify the NHRP group membership on each spoke. When the NHRP group membership name is configured on the spoke router, the spoke will announce its group membership to the hub to let it know what kind of QoS profile it should have applied.

Each spoke that is in one of these three groups has a similar QoS profile (such as WAN bandwidth, applications being used, and so on) and will use the same NHRP group membership name. In this example, Spoke 1 (the first one) belongs to NHRP SPOKE-1-GROUP, and Spoke 2 (the second one) belongs to NHRP SPOKE-2-GROUP, and so on. In this example, SPOKE-1-GROUP, SPOKE-2-GROUP, and SPOKE-3-GROUP all have different QoS profiles defined on the hub router.

> **Note** Although the main focus of this chapter is the QoS configuration of the hub router, and the participation of the spokes in the NHRP group membership, it also considered best practice to configure an outbound QoS policy on the spoke router's physical interface to protect high-priority traffic as it enters the WAN. This subject is discussed in greater detail in Chapter 39, "Home Office VPN QoS Case Study," which examines a remote teleworker case study.

Verifying Your DMVPN QoS Configuration

Now that service policies have been attached to the hub and applied to the mGRE interface, you can confirm that NHRP has accepted registration of the group membership from the spokes. It is critical to verify the NHRP spoke membership because this is the mechanism used to announce the spoke groups to the hub router, which are in turn used to apply the correct per-tunnel QoS policies.

As you can see from Example 37-14, the NHRP database has associated the three spokes with the correct NHRP groups so that the desired QoS policies can be applied.

Example 37-14 *Verifying That the Spokes Have Registered Their Group Membership with the Hub: show ip nhrp group-map*

```
ASR1000# show ip nhrp group-map
!
Interface: Tunnel10
 NHRP group: SPOKE-1-GROUP
  QoS policy: 4-CLASS-PARENT
  Tunnels using the QoS policy:
  Tunnel destination overlay/transport address
  10.1.1.30/172.17.10.1          ! Spoke 1
 NHRP group: SPOKE-2-GROUP
  QoS policy: 8-CLASS-PARENT
  Tunnels using the QoS policy:
  Tunnel destination overlay/transport address
  10.1.1.40/172.16.20.1          ! Spoke 2
 NHRP group: SPOKE-3-GROUP
  QoS policy: 12-CLASS-PARENT
  Tunnels using the QoS policy:
  Tunnel destination overlay/transport address
  10.1.1.50/172.16.30.1          ! Spoke 3
```

Another useful command is **show dmvpn detail**. Example 37-15 displays a portion of the output of this command pertaining to the QoS service policy application on the DMVPN. The output shows the DMVPN detail for a hub router that has a combination of the three groups attached.

Example 37-15 *Verifying the DMVPN Spoke Association:* ***show dmvpn detail***

```
ASR1000# show dmvpn detail
!
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
      N - NATed, L - Local, X - No Socket
      # Ent --> Number of NHRP entries with same NBMA peer
      NHS Status: E --> Expecting Replies, R --> Responding, W --
      UpDn Time --> Up or Down Time for a Tunnel
=================================================================
Interface Tunnel10 is up/up, Addr. is 10.1.1.254, VRF ""
   Tunnel Src./Dest. addr: 172.25.1.254/MGRE, Tunnel VRF ""
   Protocol/Transport: "multi-GRE/IP", Protect "MY-BIG-VPN"
   Interface State Control: Disabled
   nhrp event-publisher : Disabled
Type:Hub, Total NBMA Peers (v4/v6): 2
# Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb    Target Network
----- --------------- -------------- ----- -------- ----- ------
    1       172.16.10.1    10.1.1.30    UP 19:45:38    D
NHRP group: SPOKE-1-GROUP
 Output QoS service-policy applied: 4-CLASS-PARENT
    1       172.16.20.1    10.1.1.40    UP 20:51:07    D
NHRP group: SPOKE-2-GROUP
 Output QoS service-policy applied: 8-CLASS-PARENT
    1       172.16.30.1    10.1.1.50    UP 19:47:21    D
NHRP group: SPOKE-3-GROUP
 Output QoS service-policy applied: 12-CLASS-PARENT
```

As a final step, it is highly recommended to confirm your service policies to ensure that they are correctly configured and applied. When using a DMVPN, you use the **show policy-map multipoint** command. This interesting command is unique to DMVPNs. In almost every example you have seen in this book, QoS policies are verified using the command **show policy-map interface**. With DMVPN, the **interface** keyword is replaced with **multipoint**.

Example 37-16 shows a portion of the output of this command for one of the QoS groups using the eight-class model.

Example 37-16 *Verifying the Policy Map Configuration:* ***show policy-map multipoint***

```
ASR1000# show policy-map multipoint

Interface Tunnel10 <--> 30.0.0.1
Service-policy output: GROUP-2-PARENT
 Class-map: class-default (match-any)
  1182492 packets, 1485130581 bytes
   5 minute offered rate 24418000 bps, drop rate 19520000 bps
   Match: any
   Queueing
   queue limit 64 packets
   (queue depth/total drops/no-buffer drops) 0/912489/0
   (pkts output/bytes output) 307082/399820764
   shape (average) cir 10000000, bc 40000, be 40000
   target shape rate 10000000
   Service-policy : GROUP-2-CHILD
    queue stats for all priority classes:
     Queueing
     queue limit 512 packets
     (queue depth/total drops/no-buffer drops) 0/0/0
     (pkts output/bytes output) 90069/117269838
    Class-map: VOICE (match-all)
     28171 packets, 35382776 bytes
     5 minute offered rate 593000 bps, drop rate 0000 bps
     Match:  dscp ef (46)
     Priority: 10% (1000 kbps), burst 25000, b/w ex drops: 0

    Class-map: MULTIMEDIA-CONFERENCING (match-any)
     61898 packets, 77743888 bytes
     5 minute offered rate 1288000 bps, drop rate 0000 bps
     Match:  dscp af41 (34)
     Match:  dscp af42 (36)
     Match:  dscp af43 (38)
     Priority: 23% (2300 kbps), burst 57500, b/w ex drops: 0

    Class-map: NETWORK-CONTROL (match-all)
     3165 packets, 3896984 bytes
     5 minute offered rate 73000 bps, drop rate 0000 bps
     Match:  dscp cs6 (48)
     Queueing
     queue limit 64 packets
     (queue depth/total drops/no-buffer drops) 0/0/0
     (pkts output/bytes output) 3098/4033596
     bandwidth 5% (500 kbps)
```

```
Class-map: SIGNALING (match-all)
 4695 packets, 5896920 bytes
 5 minute offered rate 105000 bps, drop rate 0000 bps
 Match:  dscp cs3 (24)
 Queueing
 queue limit 64 packets
 (queue depth/total drops/no-buffer drops) 0/0/0
 (pkts output/bytes output) 4695/6112890
 bandwidth 2% (200 kbps)
Class-map: MULTIMEDIA-STREAMING (match-any)
 309878 packets, 389206768 bytes
 5 minute offe rate 6405000 bps, drop rate 5891000 bps
 Match:  dscp af31 (26)
 Match:  dscp af32 (28)
 Match:  dscp af33 (30)
 Queueing
 queue limit 64 packets
 (queue dep/total dp/no-buff drop/fldrop) 16/274845/0/0
 (pkts output/bytes output) 35033/45612966
 bandwidth 10% (1000 kbps)
 Fair-queue: per-flow queue limit 16 packets
  Exp-weight-constant: 4 (1/16)
  Mean queue depth: 15 packets
  dscp  Transmitted  Rand drop T/F drop Min Max Mark
         pkts/bytes pkts/bytes  pkts/bytes th th prob

  af31  35002/45572604 0/0         0/0     28 32  1/10
  af32     16/20832    0/0         0/0     24 32  1/10
  af33     15/19530    0/0         0/0     20 32  1/10

Class-map: TRANSACTIONAL-DATA (match-any)
      309879 packets, 389208024 bytes
      5 minute offered rate 6405000 bps, drop rate 4999000
 Match:  dscp af21 (18)
 Match:  dscp af22 (20)
 Match:  dscp af23 (22)
 Queueing
 queue limit 64 packets
 (qu depth/total drops/no-buf drops/flowdrops) 16/233198/0/0
 (pkts output/bytes output) 76681/99838662
 bandwidth 24% (2400 kbps)
  Fair-queue: per-flow queue limit 16 packets
   Exp-weight-constant: 4 (1/16)
   Mean queue depth: 14 packets
   dscp Transmitted  Random drop  Tail/Flow drop Min Max Mark
```

```
        pkts/bytes   pkts/bytes   pkts/bytes th th  prob

   af21 76628/99769656  0/0          0/0     28      32  1/10
   af22    22/28644     0/0          0/0     24      32  1/10
   af23    31/40362     0/0          0/0     20      32  1/10
Class-map: SCAVENGER (match-all)
 154939 packets, 194603384 bytes
 5 minute offered rate 3206000 bps, drop rate 3148000
 Match:  dscp cs1 (8)
 Queueing
 queue limit 64 packets
 (queue depth/total drops/no-buffer drops) 64/146681/0
 (pkts output/bytes output) 8258/10751916
 bandwidth 1% (100 kbps)
Class-map: class-default (match-any)
 309879 packets, 389206909 bytes
 5 minute offered rate 6405000 bps, drop rate 4937000
 Match: any
 Queueing
 queue limit 64 packets
 (queue depth/tot drop/no-buffer drops/fldrops) 16/230217/0/0
 (pkts output/bytes output) 79661/103718622
 bandwidth 25% (2500 kbps)
 Fair-queue: per-flow queue limit 16 packets
  Exp-weight-constant: 4 (1/16)
  Mean queue depth: 14 packets
  class Transmitted Random drop Tail/Flow drop Min Max Mark
        pkts/bytes  pkts/bytes  pkts/bytes   th  th  prob

  0     79661/103718622 0/0        0/0        16  32  1/10
  1         0/0         0/0        0/0        18  32  1/10
  2         0/0         0/0        0/0        20  32  1/10
  3         0/0         0/0        0/0        22  32  1/10
  4         0/0         0/0        0/0        24  32  1/10
  5         0/0         0/0        0/0        26  32  1/10
  6         0/0         0/0        0/0        28  32  1/10
  7         0/0         0/0        0/0        30  32  1/10
```

Per-Tunnel QoS Between Spokes

One of the most common questions that comes up when designing QoS for DMVPNs is how to apply the per-tunnel QoS feature between the spokes. For example, when one remote spoke dynamically establishes a tunnel to another spoke, how can an outbound

QoS policy be applied for that new tunnel, similar to the way that the hub router does it? Unfortunately, at this time, the per-tunnel QoS for DMVPN feature is available only on the hub router, meaning that outbound policy on a per-tunnel basis is not possible if different types of spokes are communicating with one another.

In the design example in this chapter, a spoke router does have a QoS policy applied outbound toward the DMVPN hub. However, because the spokes do not have access to the same NHRP database that the hub router does, it is not able to apply unique QoS policies for each different type of spoke. That said, if per-tunnel QoS is still required between spokes, the manual approach of using classes to identify each remote spoke must be statically configured by ACL. This approach is usually not worth the effort.

If spoke-to-spoke QoS is still considered as a mandatory requirement, another alternative is to consider FlexVPN. FlexVPN and DMVPN share many similarities, but one key difference relevant to this discussion is that FlexVPN does not support direct spoke-to-spoke communication. That is, all traffic between spokes must first transit the hub and then hairpin back to the other spoke. Although this obviously introduces a latency concern and does demand greater resources from the hub router, the advantage is that it solves the per-tunnel QoS problem between spokes. Similar to DMVPN, FlexVPN supports outbound per-tunnel QoS from the hub router toward the spokes. However, because all outbound spoke traffic must first transit the hub router, it means that only one QoS policy is ever required on the spoke router.

Summary

This chapter discussed various elements of DMVPN design, specifically how to implement per-tunnel QoS in a DMVPN environment. Specifically, this chapter discussed the advantages of DMVPN, and why it is such a popular architecture for VPNs, both big and small.

DMVPN architecture and design was briefly reviewed, and in particular the key elements that are used in configuring QoS in DMVPNs. In particular, NHRP was discussed in detail and how it is used on the spoke routers to announce their group membership to the hub router so that the correct outbound QoS policy can be applied at the hub.

The problem of QoS in DMVPN environments was discussed, and how to address the challenge of per-tunnel QoS at the hub router. The feature called per-tunnel QoS for DMVPN was introduced, and a detailed design example was given. In this design example, a single DMVPN hub with three classes of spoke was introduced. This allowed for an exploration of how to apply the 4-, 8-, and 12-class models to different spokes within a single DMVPN.

In addition, the QoS design on the spoke router was considered, although it was discussed that the per-tunnel QoS for DMVPN feature is available only on the hub router. Alternate QoS designs for spoke-to-spoke per-tunnel QoS were also discussed.

Additional Reading

DMVPN Overview: www.cisco.com/go/dmvpn

DMVPN Configuration Guide: http://www.cisco.com/en/US/partner/docs/ios-xml/ios/sec_conn_dmvpn/configuration/xe-3s/sec-conn-dmvpn-per-tunnel-qos.html

DMVPN Feature Guide: http://www.cisco.com/en/US/docs/ios/12_2t/12_2t13/feature/guide/ftgreips.html

This page intentionally left blank

Chapter 38

GET VPN QoS Design

In the preceding chapter, you became familiar with some of the important quality of service (QoS) considerations for one of the most popular virtual private network (VPN) architectures, Dynamic Multipoint VPN (DMVPN). Another popular VPN architecture that has a different, but equally powerful use case is Group Encrypted Transport, or GET VPN.

As discussed in the preceding chapter, DMVPN offers many improvements over classic IPsec VPN technology, especially when it comes to its ability to dynamically set up multipoint tunnels and to have QoS policies applied on a per-tunnel basis through the Next Hop Resolution Protocol (NHRP). GET VPN offers a different class of VPN capabilities when compared to DMVPN incorporating a new concept: encryption between IPsec endpoints without the use of tunnels. Although the packets transmitted between the group members technically use IPsec tunnel mode, the peer-to-peer relationship between members is not defined by traditional IPsec security associations (SAs).

The key advantage of DMVPN over classic IPsec VPNs is its capability to scale to a large number of spokes while at the same time reducing the management and configuration overhead. GET VPN takes scalability to a whole new level while at the same time supporting end-to-end QoS requirements, all without the use of tunnels.

When IPsec tunnels are established, the endpoints create a new connection on a peer-to-peer basis, effectively building a network overlay. The VPN overlay model that is used by traditional VPN and DMVPN models is powerful, but it also adds a great deal of configuration and management complexity and even relies on very specific, targeted QoS features, such as NHRP per-tunnel QoS to protect higher-priority traffic. By removing peer-to-peer tunnels, many of the limitations and complexity issues inherent to IPsec VPNs are instantly removed.

Why is this the case? Consider that network intelligence typically relies on each routing devices' ability to process IP packets as they move through the network. As was discussed in the previous two chapters, when transporting data over a VPN, when an original IP packet is wrapped into a new IPsec or generic routing encapsulation (GRE) packet (meaning it is tunneled), the original packet becomes invisible to the network. This can create difficulties that require the use of special tools, such as the QoS preclassify feature. When you have full control of the original IP packet header, even in the case of IPsec-encrypted VPNs (such as GET VPN), it becomes much easier to ensure the end-to-end operation of applications like voice and video that rely on key network features, such as routing, multicast, and QoS.

This chapter explores the following topics:

- GET VPN QoS overview
- GET VPN QoS configuration review
- QoS design in GET VPN environments
- Working with your service provider when deploying GET VPN

GET VPN QoS Overview

Whereas DMVPN is a great approach for hub-and-spoke VPNs over a public untrusted network, such as the Internet, GET VPN is best suited to private networks. Primarily, GET VPN is deployed in any-to-any (site-to-site) scenarios that use a private network, such as a Multiprotocol Label Switching (MPLS) network. Suppose, for example, that your company has many sites that are connected over a Layer 3 VPN MPLS service from a local provider. Although you may assume the site-to-site links are private and secure because of the nature of MPLS, your company may require that these connections over the provider's network be encrypted to ensure optimal privacy.

Although you could conceivably deploy a DMVPN within the MPLS network itself, this would be somewhat difficult to implement and manage, and the QoS design would be cumbersome. By way of comparison, if DMVPN technology were used, it would create a new overlay VPN environment within the existing MPLS VPN, whereas if GET VPN were used it would simply encrypt the packets coming and going from the customer-edge (CE) routers without the use of tunnels. Thus, one major difference is that GET VPN has no real concept of hub-and-spoke, which in itself simplifies the QoS architecture because no one major hub is aggregating all the remote sites and liable to massive oversubscription.

When evaluating whether GET VPN is the right VPN topology to deploy, it means you have an existing private network or MPLS VPN environment in place, but you simply want to encrypt traffic flowing through your network between the remote CE routers. Table 38-1 outlines some of the major differences between the DMVPN and GET VPN models.

Table 38-1 *Comparing DMVPN and GET VPN Technologies*

	DMVPN	GET VPN
Use case	Public networks (Internet).	Private networks (MPLS)
Network style	Hub-and-spoke and spoke-to spoke.	Any-to-any
Routing architecture	Routing inside GRE tunnels.	Dynamic routing of the native IPsec packets
Encryption style	Point-to-Point encryption.	Group encryption
QoS implementation	Per-tunnel QoS management through NHRP group membership. Uses a hierarchical shaper on the hub multipoint GRE (mGRE) interface.	QoS is applied at each GET VPN group member because no tunnels are used. Typically uses a hierarchical shaper on the egress interface.

The following section discuss some of the key QoS concepts related to GET VPN.

Group Domain of Interpretation

With the introduction of Group Encryption Transport, or GET, it is possible to deploy any-to-any IPsec VPNs without the use of tunnels. In both classic IPsec VPN and DMVPN technology, IPsec forms a peer relationship called a security association, or SA, between the two endpoints of the VPN. When the two endpoints begin a negotiation of the SA to agree upon the encryption and hash algorithms, they are essentially building the foundation of a point-to-point tunnel. The SA therefore is only meaningful between those two routers, and nowhere else. For example, in a DMVPN environment, when a spoke router needs to send encrypted traffic to another spoke it uses NHRP to first discover, and then automatically build a new security association to that new spoke (although DMVPN does this dynamically). The DMVPN approach relies on a separate SA, or tunnel, to each remote site; therefore, QoS must also be applied on a per SA basis for each remote location.

In the case of GET VPN, there is no concept of SAs between specific routers in the network, meaning that there are no IPsec tunnels. Instead, GET VPN leverages the concept of a group SA, which is shared by all encrypting nodes in the network. Because all the routers in the GET VPN network belong on one big group SA, there is no need to apply QoS on a per-site basis either; you simply need to configure QoS on the egress interface on each GET VPN router. Thus the whole concept of per-tunnel QoS that was used for DMVPN is irrelevant in a GET VPN design.

This technology is called the Group Domain of Interpretation (GDOI), defined in RFC 3547. Before diving deeper into how GDOI works and supports QoS in a GET VPN environment, let's begin by discussing the two types of devices that are used to build a GET VPN network.

GET VPN Building Blocks

In GET VPN deployments, two types of routers are used: the group member (GM) router, and the key server (KS) router. The GM router is the device that does the actual encryption and decryption of packets as they come and go from each site, and the KS is responsible for managing the encryption keys used by the GMs. In an MPLS environment, the GM service usually runs on the CE router, and the KS is usually run on dedicated router hardware kept either in the data center or in a secure demilitarized zone (DMZ).

While there is always at least one GM router at each remote site in the network, there is only a single (or a redundant pair) of KS routers deployed for the entire GET VPN network. The KS router has the responsibility of managing and distributing the group encryption keys that are used by the various GM routers throughout the network. The KS has the job of sending out the encryption keys and IPsec encryption policies to all authenticated and registered GM routers, and also periodically rekeys the GM routers on the network when necessary.

The KS only issues keys to GM routers that it trusts, which means that the GM routers need to have been both registered and authenticated by the KS before the group keys are issued to the GMs, thus enabling IPsec encryption services.

From a QoS perspective, the two control protocols that are being used by GET VPN are the GDOI control plane protocol, which uses UDP port 848, and ISAKMP, which uses UDP port 500. Similar to the way IOS handles other control and management protocols, such as routing protocol updates, the router by default marks the differentiated services code point (DSCP) values of both of these control protocols as CS6. In this way, both the GDOI and ISAKMP packets can be correctly processed by the per-hop behavior (PHB) QoS tools within the private network cloud. As long as the trust boundaries are set correctly throughout the network, no special remarking of these control packets is necessary.

The sizing of the GM router will depend on the expected IPsec throughput and other services running on the router, but the KS router functions purely as a control plane device, only responsible for managing the GDOI membership and encryption keys. Because the KS is not required to forward any data, it is often sized with a smaller Integrated Services Router (ISR), such as a ISR G2 2900 series. The KS router usually resides in the corporate data center or another secure place within network. Table 38-2 outlines some of the key features and QoS requirements of the GM and KS routers.

As an example of how this works, let’s say you have a private network with 20 sites that use an L3 MPLS provider for transport. If you enabled GET VPN on this network, each of these 20 sites are members of the same encryption group, and would therefore be able to send encrypted packets to any other site. As you can begin to see, this any-to-any encryption model complements very well the any-to-any routing model inherent to MPLS.

Table 38-2 *GET VPN Network Components*

Component Type	Description	QoS Requirements
Group member (GM)	■ The routers on the corporate network that do the encryption. In an MPLS network, these are the CE routers. ■ Usually deployed on an ISR or ASR router.	■ The GM receives packets with an existing DSCP marking. Re-marking on the LAN side is usually not required. ■ The GM needs to support egress class-based weighted fair queuing (CBWFQ) toward the WAN. ■ A hierarchical shaper may be required if there is a physical interface bandwidth mismatch between the GM and the offered WAN bandwidth.
Key server (KS)	■ Responsible for pushing group encryption keys out to the GMs. ■ The KS is located in a secure place in the corporate network (data center and so on). ■ An ISR router is normally used as the KS. ■ KS routers can be deployed redundantly.	■ The KS requires no special QoS configuration. ■ The KS will automatically mark all control packets (GDOI and ISAKMP) as CS6 for downstream QoS handling.

Figure 38-1 illustrates the general QoS architecture of a GET VPN environment.

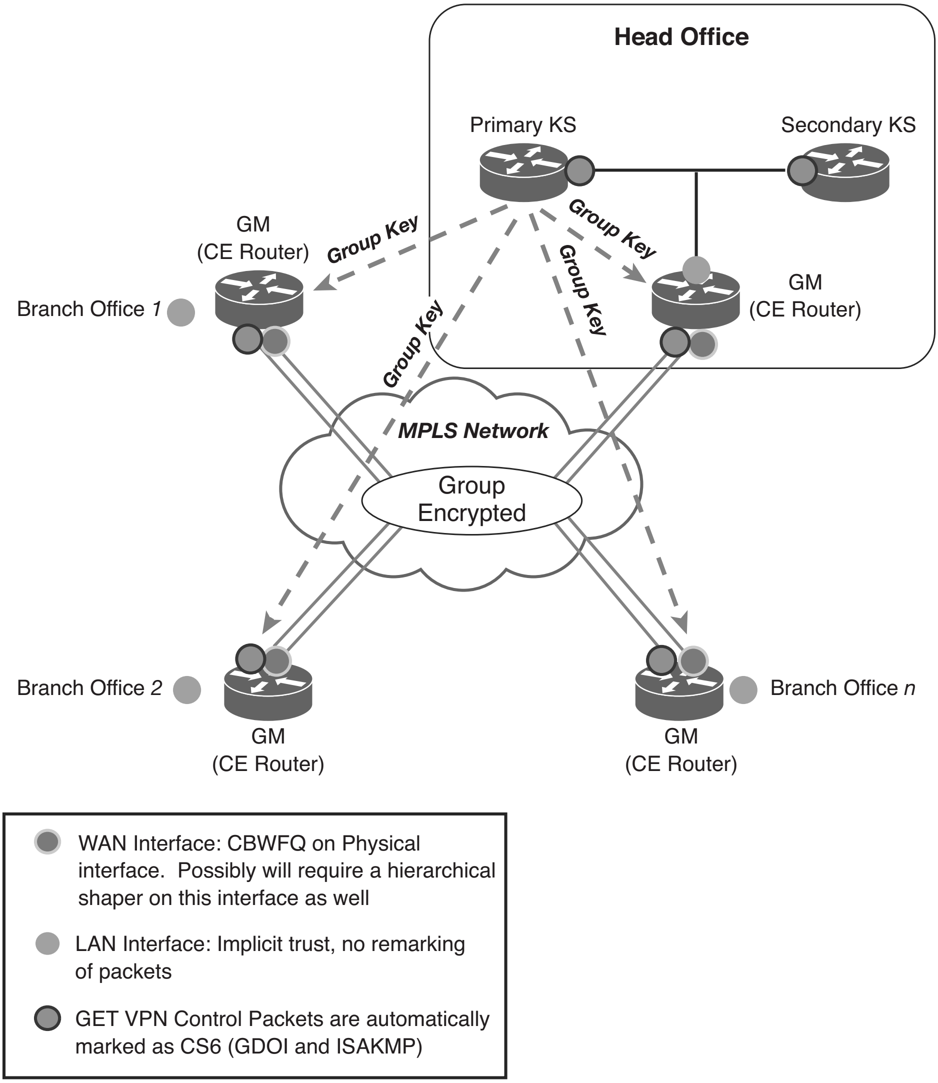

Head Office
Primary KS
Secondary KS
Group Key
Group Key
Group Key
Group Key
GM
(CE Router)
Branch Office 1
GM
(CE Router)
MPLS Network
Group
Encrypted
Branch Office 2
GM
(CE Router)
GM
(CE Router)
Branch Office n
WAN Interface: CBWFQ on Physical interface. Possibly will require a hierarchical shaper on this interface as well
LAN Interface: Implicit trust, no remarking of packets
GET VPN Control Packets are automatically marked as CS6 (GDOI and ISAKMP)

Figure 38-1 *Where to Implement QoS in a GET VPN Architecture*

IP Header Preservation

One of the design nuances of IPsec tunnel mode is that it encapsulates the original IP packet inside of a new IP packet with a new IP header, which corresponds to the source and destination IP addresses of the IPsec tunnel endpoints, not the original IP packet. This lack of IP header preservation results in a certain level of complexity when applying QoS because only the Type of Service (ToS) byte is copied to the IPsec header by default.

Of course, if you were encrypting over a public network like the Internet, it makes good sense to hide the source and destination addresses of the IP packets sent across VPN connection. However, in a private network infrastructure, you gain little advantage from this, and what's more, the extra layer of obfuscation adds an unnecessary level of complexity to the network. For example, in an MPLS network the provider-edge (PE) router normally handles IP routing within each virtual routing and forwarding (VRF). However, if you have already encrypted traffic from a branch office CE router, the PE router has no visibility into the original IP packet. Therefore, the PE can only make a routing decision based on the outside IP address information, regardless of the actual destination host. From a QoS perspective, it also makes matters more complicated because the original IP packet header is now hidden inside the IPsec tunnel mode header and classification becomes difficult.

With this in mind, the designers of GET VPN devised a new approach that would allow networks to be built where the entire original IP header could be preserved, but only the data would be encrypted. Figure 38-2 illustrates the differences between classic IPsec tunnel mode and GET VPN packet headers.

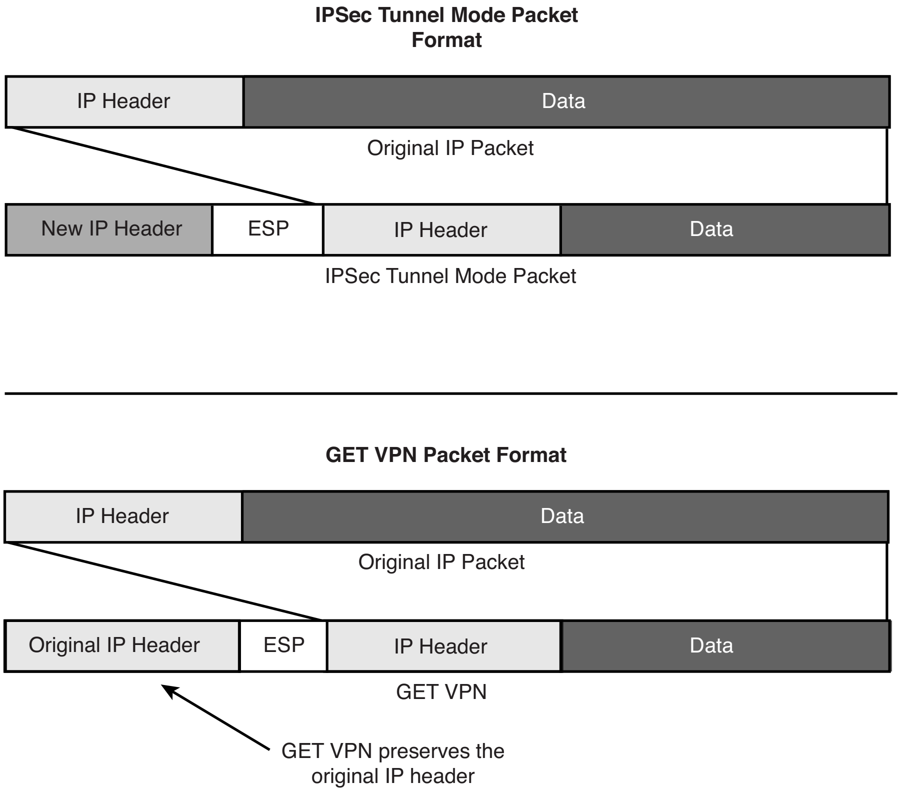


Figure 38-2 *GET VPN IP Packet Header Preservation*

As you can see from Figure 38-2, GET VPN works very differently from IPsec tunnel mode. Because of GET VPN's capability to preserve the original IP header, packet handling on the PE router is greatly simplified. For example, preserving the original IP addresses simplifies interior gateway protocol (IGP) routing decisions, allows multicast traffic to be processed unhindered (assuming the MPLS provider also supports multicast replication services of some sort), and also simplifies QoS handling.

IP header preservation has a key impact on the QoS toolset because not only the original ToS field is preserved (which is the default behavior for all IPsec technologies) but also the original IP source and destination addresses, thus offering a wider range of classification options. That said, a GET VPN packet is still considered an ESP IPsec packet, not TCP or UDP. So, if you still wanted to perform QoS classification based on the TCP or UDP port numbers, the QoS preclassify feature might still prove useful because this allows the entire TCP/IP header to be cloned and kept in router memory for classification.

GET VPN Configuration Review

GET VPN can actually be a somewhat complex and detailed subject, with many subfeatures that could literally consume an entire book on their own. In the pages of this chapter, the basic features and configuration elements of GET VPN are presented, which will allow us to then focus on the more relevant QoS design elements within a GET VPN environment.

Key Server Configuration

Example 38-1 displays the KS configuration that is used in the GET VPN QoS design examples discussed in this chapter. In the following examples discussed in this chapter, an ISR 2951 is used as the KS.

Example 38-1 *The GET VPN Key Server Configuration*

```
ISR2951-KS# show running-config
!
crypto isakmp policy 10
 encr aes 256
 authentication pre-share
!
!
crypto ipsec transform-set MYSET esp-aes esp-sha-hmac
!
crypto ipsec profile GDOI
 set security-association lifetime seconds 7200
 set transform-set MYSET
!
crypto gdoi group MY-GET-VPN
```

```
!
! The identity number must match on all GMs and the KS
identity number 101
server local
 rekey lifetime seconds 7200
 rekey retransmit 10 number 2
 rekey authentication MYPUBKEY rsa GET-VPN-KEY
 rekey transport unicast
 sa ipsec 1
  profile GDOI
  match address ipv4 CONTROL-PLANE
```

From a QoS perspective, there is really nothing to configure on the KS because it really is acting just as a server. The GET VPN GDOI control plane, which uses UDP port 848, is already marking all traffic as CS6 by default, as is ISAKMP, which uses UDP port 500. These are the only protocols that are communicated between the KS and GM routers. Beyond this, the only other important point to note is that the L2 device that connects to the KS must establish a DSCP trust relationship with the KS router and then pass all packets with an unchanged DSCP value.

It is also important to note that you will need to ensure that there is sufficient bandwidth for the KS to send rekey messages to the GMs. This varies with the size of the network and number of policies. Nevertheless, you want this to be treated as high-priority traffic throughout the network.

Once the GET VPN system is up and running, you will be able to see various group members that are registered with the KS, as demonstrated in Example 38-2.

Example 38-2 *Showing the GDOI Status on the Key Server*

```
ISR2951# show crypto gdoi ks
Total group members registered to this box: 20
Key Server Information For Group MY-GET-VPN:
    Group Name               : MY-GET-VPN
    Group Identity           : 101
    Group Members            : 20
    IPsec SA Direction       : Both
    ACL Configured           : access-list CONTROL-PLANE
```

Now that the KS is up and running, the next step is to configure the GMs.

Group Member Configuration

In comparison to the KS, the GM has only a minimal IPsec configuration. The IPsec transform sets, IPsec profile, and the encryption access control lists (ACLs) are all pushed from the KS down to the GMs. Example 38-3 illustrates how you may configure the GM for GET VPN. Note that an ISR 3925 was used to generate these configurations.

Example 38-3 *Configuring the Group Member*

```
ISR3925-GM# show running-configuration
!
crypto isakmp policy 10
 encr aes 256
 authentication pre-share
crypto isakmp key cisco123 address 0.0.0.0
!
crypto gdoi group MY-GET-VPN
 identity number 101
 server address ipv4 172.16.1.254
!
crypto map MYMAP gdoi fail-close
 match address PROTECT-KS
 activate
crypto map MYMAP 10 gdoi
 set group MY-GET-VPN
!
interface GigabitEthernet0/0
 ip address 192.168.50.2 255.255.255.0
 crypto map MYMAP
```

From the GS it is now possible to confirm the successful registration with the KS, as demonstrated in Example 38-4.

Example 38-4 *Confirming GM Registration*

```
ISR3925-GM# show crypto gdoi
GROUP INFORMATION
    Group Name               : MY-GET-VPN
    Group Identity           : 101
    Crypto Path              : ipv4
    Key Management Path      : ipv4
    Rekeys received          : 0
    IPsec SA Direction       : Both
     Group Server list       : 172.16.1.254

    Group member             : 192.168.50.2
       Version               : 1.0.1
       Registration status   : Registered
```

```
      Registered with       : 172.16.1.254
      Re-registers in       : 72 sec
      Succeeded registration: 28
      Attempted registration: 28
      Last rekey from       : 0.0.0.0
      Last rekey seq num    : 0
      Multicast rekey rcvd  : 0
      allowable rekey cipher: any
      allowable rekey hash  : any
      allowable transformtag: any ESP
   Rekeys cumulative
      Total received        : 0
      After latest register : 0
      Rekey Received        : never
ACL Downloaded From KS 172.16.1.254:
  access-list  deny pim any any
  access-list  deny udp any any port = 500
  access-list  deny udp any any port = 848
  access-list  permit ip any any
```

At this point, you now have a functioning GET VPN system. One interesting point to note is that although the KS will push out the IPsec encryption keys, encryption ACLs, and so on, one thing that is not centrally managed by the KS is the QoS configuration. The following section discusses how to layer on a QoS design to this GET VPN example.

GET VPN QoS Configuration

Thanks to the tunnel-less nature of GET VPN, the QoS configuration steps involved are in many ways much simpler than for any other type of VPN system. For example, in the case of DMVPN, QoS needs to be applied to each DMVPN spoke that terminates on the hub router. This involves hierarchical QoS policies that leverage a tunnel shaper on the parent level and various QoS tools applied in the child policy. With GET VPN, none of this is required.

The concept of GET VPN revolves around a shared IPsec group domain, meaning that there are no tunnels to speak of. Because there are no tunnels to work with in GET VPN, all that is required is to apply a service policy with the appropriate QoS policy to the physical interface that faces into the GET VPN system. As with the previous configuration examples, the QoS configuration is added to the MPLS CE, or private WAN-edge routers.

The steps for configuring QoS in a GET VPN network are summarized as follows:

GM routers:

1. Configure the QoS class maps.
2. Configure the QoS policy maps.
3. Attach a policy map to the physical WAN interface.

KS Routers require no specific QoS configuration, but the following steps should still be followed:

1. Ensure that a DSCP trust relationship exists for the KS control traffic.
2. Ensure that sufficient bandwidth is available for the KS control traffic (especially the GDOI rekeying traffic).

These steps are detailed in the following sections. The following sections examine how to apply the three different QoS models (4-class, 8-class, and 12-class models) to a GM router that is a part of a GET VPN system.

Configuring a GM with the Four-Class Model

Continuing the example from the previously configured BM router (see Example 38-3), you can now add QoS to the GM configuration. As with the previous examples in this book, it is understood that the appropriate class maps have already been configured on the hub router (as defined in the Chapter 28, "WAN Aggregator (Cisco ASR 1000) QoS Design."

The next step in this configuration is to build the policy map, as demonstrated in Example 38-5.

Note Some GET VPN environments do not necessarily require a hierarchical shaper if there is a correct bandwidth match between the CE/GM router interface and the PE router interface. However, if there is a bandwidth mismatch, a hierarchical shaper is required to provide backpressure to the CE/GM router interface to enact appropriate QoS tools. The following examples are shown without a hierarchical shaper, but one can be added if necessary using the policy maps shown below as child policies.

Example 38-5 *Configuring the Four-Class GET VPN QoS Policy Map*

```
ISR3925-GM(config)# policy-map 4-CLASS-QOS-GETVPN
 ! Creates the four-class policy map
ISR3925-GM(config-pmap)# class REALTIME
ISR3925-GM(config-pmap-c)# priority percent 33
ISR3925-GM(config-pmap)# class CONTROL
ISR3925-GM(config-pmap-c)# bandwidth percent 7
ISR3925-GM(config-pmap)# class TRANSACTIONAL-DATA
```

```
ISR3925-GM(config-pmap-c)# bandwidth percent 35
ISR3925-GM(config-pmap-c)# fair-queue
ISR3925-GM(config-pmap)# class class-default
ISR3925-GM(config-pmap-c)# bandwidth percent 25
ISR3925-GM(config-pmap-c)# fair-queue
ISR3925-GM(config-pmap-c)# random-detect
```

You can verify the configuration in Example 38-5 with the following commands:

- **show class-map**
- **show policy-map**

Finally, apply the policy map the to WAN interface, as demonstrated in Example 38-6.

Example 38-6 *Apply the Policy Map to the WAN Interface*

```
ISR3925-GM(config)# interface GigabitEthernet0/0
ISR3925-GM(config-if)# ip address 192.168.50.2 255.255.255.0
ISR3925-GM(config-if)# crypto map MYMAP
ISR3925-GM(config-if)# service-policy output 4-CLASS-QOS-GETVPN
 ! Attaches the four-class QoS policy in the outbound direction to
 ! the WAN interface
```

You can verify the configuration in Example 38-6 with the **show policy-map interface GigabitEthernet0/0** command.

Configuring a GM with the Eight-Class Model

The eight-class model follows very closely with the four-class model just presented. As before, it is assumed the appropriate class maps have been configured (see Example 38-7).

Example 38-7 *Create the Eight-Class Policy Map*

```
ISR3925-GM(config )# policy-map 8-CLASS-QOS-GETVPN
 ! Creates the eight-class policy map
ISR3925-GM(config-pmap)# class VOICE
ISR3925-GM(config-pmap-c)# priority percent 10
ISR3925-GM(config-pmap)# class MULTIMEDIA-CONFERENCING
ISR3925-GM(config-pmap-c)# priority 23
ISR3925-GM(config-pmap)# class NETWORK-CONTROL
ISR3925-GM(config-pmap-c)# bandwidth percent 5
ISR3925-GM(config-pmap)# class SIGNALING
ISR3925-GM(config-pmap-c)# bandwidth percent 2
ISR3925-GM(config-pmap)# class MULTIMEDIA-STREAMING
```

```
ISR3925-GM(config-pmap-c)# bandwidth percent 10
ISR3925-GM(config-pmap-c)# fair-queue
ISR3925-GM(config-pmap-c)# random-detect dscp-based
ISR3925-GM(config-pmap)# class TRANSACTIONAL-DATA
ISR3925-GM(config-pmap-c)# bandwidth percent 24
ISR3925-GM(config-pmap-c)# fair-queue
ISR3925-GM(config-pmap-c)# random-detect dscp-based
ISR3925-GM(config-pmap)# class SCAVENGER
ISR3925-GM(config-pmap-c)# bandwidth percent 1
ISR3925-GM(config-pmap)# class class-default
ISR3925-GM(config-pmap-c)# bandwidth percent 25
ISR3925-GM(config-pmap-c)# fair-queue
ISR3925-GM(config-pmap-c)# random-detect
```

You can verify the configuration in Example 38-7 with the following commands:

- **show class-map**
- **show policy-map**

As before, you can now attach this new policy map to the GET VPN WAN interface, as demonstrated in Example 38-8.

Example 38-8 *Configuring the Eight-Class Model QoS Policy Map*

```
ISR3925-GM(config)# interface GigabitEthernet0/0
ISR3925-GM(config-if)# ip address 192.168.50.2 255.255.255.0
ISR3925-GM(config-if)# crypto map MYMAP
ISR3925-GM(config-if)# service-policy output 8-CLASS-QOS-GETVPN
 ! Attaches the 8-Class QoS policy in the outbound direction to
 ! the WAN interface
```

You can verify the configuration in Example 38-8 with the **show policy-map interface GigabitEthernet0/0** command.

Configuring a GM with the Twelve-Class Model

The 12-class model again follows very closely with the 4- and 8-class models just presented. As before, it is assumed the appropriate class maps have been configured. Begin by creating the policy map, as demonstrated in Example 38-9.

Example 38-9 *Create the 12-Class GET VPN QoS Policy Map*

```
ISR3925-GM(config)# policy-map 12-CLASS-QOS-GETVPN
 ! Creates the 12-class policy map
```

```
ISR3925-GM(config-pmap)# class VOICE
ISR3925-GM(config-pmap-c)# priority percent 10
ISR3925-GM(config-pmap)# class REALTIME-INTERACTIVE
ISR3925-GM(config-pmap-c)# priority percent 13
ISR3925-GM(config-pmap)# class BROADCAST-VIDEO
ISR3925-GM(config-pmap-c)# priority 10
ISR3925-GM(config-pmap)# class NETWORK-CONTROL
ISR3925-GM(config-pmap-c)# bandwidth percent 2
ISR3925-GM(config-pmap)# class SIGNALING
ISR3925-GM(config-pmap-c)# bandwidth percent 2
ISR3925-GM(config-pmap)# class NETWORK-MANAGEMENT
ISR3925-GM(config-pmap-c)# bandwidth percent 3
ISR3925-GM(config-pmap)# class MULTIMEDIA-CONFERENCING
ISR3925-GM(config-pmap-c)# bandwidth percent 10
ISR3925-GM(config-pmap-c)# fair-queue
ISR3925-GM(config-pmap-c)# random-detect dscp-based
ISR3925-GM(config-pmap)# class MULTIMEDIA-STREAMING
ISR3925-GM(config-pmap-c)# bandwidth percent 10
ISR3925-GM(config-pmap-c)# random-detect dscp-based
ISR3925-GM(config-pmap-c)# fair-queue
ISR3925-GM(config-pmap)# class TRANSACTIONAL-DATA
ISR3925-GM(config-pmap-c)# bandwidth percent 10
ISR3925-GM(config-pmap-c)# fair-queue
ISR3925-GM(config-pmap-c)# random-detect dscp-based
ISR3925-GM(config-pmap)# class BULK-DATA
ISR3925-GM(config-pmap-c)# bandwidth percent 4
ISR3925-GM(config-pmap-c)# fair-queue
ISR3925-GM(config-pmap-c)# random-detect dscp-based
ISR3925-GM(config-pmap)# class SCAVENGER
ISR3925-GM(config-pmap-c)# bandwidth percent 1
ISR3925-GM(config-pmap)# class class-default
ISR3925-GM(config-pmap-c)# bandwidth percent 25

ISR3925-GM(config-pmap-c)# fair-queue
ISR3925-GM(config-pmap-c)# random-detect
```

You can verify the configuration in Example 38-9 with the following commands:

- **show class-map**
- **show policy-map**

Finally, apply the policy map the to WAN interface, as demonstrated in Example 38-10.

Example 38-10 *Apply the Policy Map to the GET VPN WAN Interface*

```
ISR3925-GM(config)# interface GigabitEthernet0/0
ISR3925-GM(config-if)# ip address 192.168.50.2 255.255.255.0
ISR3925-GM(config-if)# crypto map MYMAP
ISR3925-GM(config-if)# service-policy output 12-CLASS-QOS-GETVPN
 ! Attaches the 12-Class QoS policy in the outbound direction to
 ! the WAN interface
```

You can verify the configuration in Example 38-10 with the **show policy-map interface GigabitEthernet0/0** command.

Confirming the QoS Policy

After the QoS configuration has been created and applied to the router, the final step is to confirm that your configuration has been built as you intended. The most useful command to do this is **show policy-map interface** *x/y*. Example 38-11 shows the output of this command for the eight-class model, which was previously configured.

Example 38-11 *The Output of* ***show policy-map interface*** *x/y*

```
ISR3925-GM# show policy-map interface GigabitEthernet0/0/0
GigabitEthernet0/0/0
  Service-policy output: 8-CLASS-QOS-GETVPN
    queue stats for all priority classes:
      Queueing
      queue limit 512 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 575850/749756700
    Class-map: VOICE (match-all)
      194950 packets, 253824900 bytes
      30 second offered rate 4486000 bps, drop rate 0000 bps
      Match:  dscp ef (46)
      Priority: 10% (10000 kbps), burst bytes 250000,b/w exced drops: 0

    Class-map: MULTIMEDIA-CONFERENCING (match-any)
      380900 packets, 495931800 bytes
      30 second offered rate 8966000 bps, drop rate 0000 bps
      Match:  dscp af41 (34)
      Match:  dscp af42 (36)
      Match:  dscp af43 (38)
      Priority: 23% (23000 kbps),burst bytes 575000,b/w exceed drops: 0

    Class-map: NETWORK-CONTROL (match-all)
      5314 packets, 5186815 bytes
      30 second offered rate 101000 bps, drop rate 0000 bps
      Match:  dscp cs6 (48)
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 5263/5181877
      bandwidth 5% (5000 kbps)
    Class-map: SIGNALING (match-all)
      5905 packets, 7688310 bytes
      30 second offered rate 136000 bps, drop rate 0000 bps
      Match:  dscp cs3 (24)
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 5905/7688310
      bandwidth 2% (2000 kbps)
    Class-map: MULTIMEDIA-STREAMING (match-any)
      389905 packets, 507656310 bytes
      30 second offered rate 8973000 bps, drop rate 0000 bps
      Match:  dscp af31 (26)
      Match:  dscp af32 (28)
      Match:  dscp af33 (30)
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 8/31/0/0
      (pkts output/bytes output) 389874/507615948
      bandwidth 10% (10000 kbps)
      Fair-queue: per-flow queue limit 16 packets
        Exp-weight-constant: 4 (1/16)
        Mean queue depth: 8 packets
        dscp   Transmitted Random drop  Tail/Flow   drop Min  Max Mark
               pkts/bytes  pkts/bytes   pkts/bytes     th   th  prob

        af31 387699/504784098 0/0          0/0         28   32  1/10
        af32   1249/1626198   0/0          0/0         24   32  1/10
        af33    926/1205652   0/0          0/0         20   32  1/10
    Class-map: TRANSACTIONAL-DATA (match-any)
      389888 packets, 507634176 bytes
      30 second offered rate 8967000 bps, drop rate 0000 bps
      Match:  dscp af21 (18)
      Match:  dscp af22 (20)
      Match:  dscp af23 (22)
      Queueing
      queue limit 99 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 0/0/0/0
      (pkts output/bytes output) 389888/507634176
      bandwidth 24% (24000 kbps)
      Fair-queue: per-flow queue limit 24 packets
        Exp-weight-constant: 4 (1/16)
        Mean queue depth: 6 packets
        dscp   Transmitted Random drop  Tail/Flow   drop Min  Max Mark
               pkts/bytes  pkts/bytes   pkts/bytes     th   th  prob

        af21 388105/505312710 0/0          0/0         42   49  1/10
        af22   1047/1363194   0/0          0/0         36   49  1/10
        af23    736/958272    0/0          0/0         30   49  1/10
    Class-map: SCAVENGER (match-all)
      194950 packets, 253824900 bytes
      30 second offered rate 4486000 bps, drop rate 0000 bps
      Match:  dscp cs1 (8)
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 3/0/0
      (pkts output/bytes output) 194950/253824900
      bandwidth 1% (1000 kbps)
    Class-map: class-default (match-any)
      389905 packets, 507653837 bytes
      30 second offered rate 8973000 bps, drop rate 0000 bps
      Match: any
      Queueing
      queue limit 104 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 0/0/0/0
      (pkts output/bytes output) 389905/507653837
      bandwidth 25% (25000 kbps)
      Fair-queue: per-flow queue limit 26 packets
        Exp-weight-constant: 4 (1/16)
        Mean queue depth: 6 packets

        class  Transmitted Random drop Tail/Flow   drop Min Max Mark
               pkts/bytes     pkts/bytes  pkts/bytes       th  th  prob

        0    389905/507653837    0/0        0/0            26  52  1/10
        1         0/0            0/0        0/0            29  52  1/10
        2         0/0            0/0        0/0            32  52  1/10
        3         0/0            0/0        0/0            35  52  1/10
        4         0/0            0/0        0/0            39  52  1/10
        5         0/0            0/0        0/0            42  52  1/10
        6         0/0            0/0        0/0            45  52  1/10
        7         0/0            0/0        0/0            48  52  1/10
```

This section has demonstrated how to create, build, and apply the 4-, 8-, and 12-class QoS models that have been discussed throughout this book. One further point must be mentioned about when to use the **qos pre-classify** command in GET VPN environments. This is discussed in the following section.

How and When to Use the QoS Preclassify Feature

Design principles:

- If classification based on source or destination IP address is required, the QoS preclassify feature is not required, but is recommended nonetheless.
- If classification based on TCP or UDP port numbers is required, QoS preclassify is required.
- As a general rule, always enable the Qos preclassify feature in GET VPN environments.

As with all IOS IPsec implementations, the router always copies the DSCP markings of the original IP packet to the outer IPsec header. This is also true of GET VPN. However, in the case of GET VPN, the entire original IP packet header is preserved, meaning that if QoS classification needs to be done based on the source or destination IP address, this is available to you without the need for the QoS preclassify feature. Although this does offer you more options for classification, it does not help much if you want to classify packets on the UDP or TCP port number.

Even though GET VPN preserves the original IP packet header, the protocol type of the packet is still type 50 (ESP). However, TCP is identified as protocol type 6, and UDP is identified as protocol type 17, meaning that classification based on the UDP or TCP port numbers is impossible without a little help from the QoS preclassify feature. As was described in some detail earlier in Chapter 36, "IPsec VPN QoS Considerations and Recommendations," the QoS preclassify feature allows the router to make a copy, or clone, of the original TCP/IP header to be used later for QoS classification. When this feature is used on the crypto map, it is possible to enable QoS classification based on the TCP or UDP port numbers.

Example 38-12 illustrates how to enable this feature on a GM router.

Example 38-12 *Configuring the QoS Preclassify Feature on a GM Router*

```
ISR3925-GM(config)# crypto map MYMAP 10 gdoi
ISR3925-GM(config-crypto-map)# qos pre-classify
```

As a general rule of thumb, it is always a good idea to configure the QoS preclassify feature, even if you do not intend to classify packets on the TDP or UDP port numbers. Because this feature has very little performance impact on the router, there is really no downside to enabling it on the GS routers. An important point to note is that even though you may use **qos pre-classify** on the encrypting router, because of the per-hop behavior of QoS, you lose the ability to classify based on TCP and UDP ports at all intermediate routers (such as in the MPLS cloud).

A Case for Combining GET VPN and DMVPN

It might seem like a strange marriage, but there is actually a very strong use case for designing a VPN network that uses both GET VPN and DMVPN at the same time. From a network responsiveness perspective, DMVPN has one key drawback when spoke-to-spoke communication occurs. By design, it usually take a second or two for the dynamic tunnel to come up and be useful. Recall that with DMVPN the spokes always have a permanent IPsec connection to the hub but that spoke-to-spoke tunnels are on demand and are dynamic. As it turns out, the delay in setting up the spoke-to-spoke tunnel is not caused by NHRP, nor by the packetization of the GRE tunnel, but rather by the exchange of ISAKMP messaging and the establishment of the IPsec SA between the two spoke routers.

This slight delay can actually be quite noticeable for users. For example, when making a VoIP phone call from one spoke to another, this small delay in setting up the dynamic encrypted VPN tunnel can result in a pause in the call signaling between the source and destination, to the point that the user perceives that the network is slow because the far-end phone does not ring as fast as it should.

This is where these two VPN topologies comes together. If DMVPN were used solely for setting up GRE tunnels, but IPsec were left out of the picture, the establishment of the VPN tunnels between spokes improves significantly. In fact, there is almost no delay in GRE-only spoke-to-spoke communications. The only challenge with this approach is that it does not provide any encryption. To overcome this problem, GET VPN can be configured to run inside of the GRE tunnels. In this way, the spoke or hub data packets are always encrypted with the group SA, meaning that no setup time is required. The encrypted packets are simply transported over the DMVPN GRE tunnels. Thus, the DMVPN spoke routers also become GET VPN GM routers, which encrypt the data and then funnel it into the correct GRE tunnel.

Obviously, this approach is quite different from the normal DMVPN design. In traditional DMVPN, IPsec is used to encrypt the entire GRE packet with little regard for the data flowing inside of the GRE tunnel. However, when using GET VPN, it is not the GRE tunnel itself that is being encrypted, but rather the data packets that go through the GRE tunnel. Actually, the DMVPN component of this solution allows the remote spokes to appear like they are connected over a private carrier network. However, in this case, it is the GRE tunnels spawned by the DMVPN design that allows the cloud in the middle to appear as a private network instead of a public one.

The net result of this design is that spoke-to-spoke communication with high-speed tunnel establishment is supported while still providing the appropriate level of security through IPsec encryption. The overall quality of the user experience is thus improved.

Working with Your Service Provider When Deploying GET VPN

Design principles:

- Ensure that there is consistent end-to-end DSCP handling throughout the MPLS WAN network

QoS works on a per-hop basis, meaning that QoS policies must be applied at each node within the network. The QoS tools presented in this chapter have focused on how to deal with congestion at the GM router, as packets leave the WAN interface. To enable a true end-to-end QoS policy, the MPLS provider should also implement congruent QoS policies throughout their network to match what has been deployed in your network.

In many cases, it can be difficult to match your company's QoS policy with what has been offered from your service provider, but in many cases a compromise must be met and a hybrid of one of these three models might be used.

The general recommendation is to map the applications to the appropriate DSCP value such that the queuing policy is consistent from beginning to end. For example, in an MPLS environment, there should be consistent DSCP handling from the CE to PE to P to PE to CE.

Summary

This chapter focused on the design of QoS in GET VPN (also known as tunnel-less VPN) networks.. Unlike DMVPN, GET VPN is best suited to private network topologies, such as MPLS networks. GET VPN relies on the concept of group encryption, meaning that no point-to-point SAs are required. Instead, GET VPN uses a group SA to allow any-to-any encryption.

GET VPN also has the added advantage that the original IP header is preserved through the encryption process, which makes it easier to classify QoS based on not just the DSCP or ToS values, but it also allows you to classify based on the source and destination IP addresses. If other classification is required, such as TCP or UDP port numbers, the QoS preclassify feature should be used in the crypto map.

Three examples were given in this chapter outlining how to implement QoS on a GET VPN GM router. These were for the 4-, 8-, and 12-class models for QoS. Because GET VPN does not rely on tunnels, the service policies for any QoS model is simply applied to the WAN interface without the need for a parent shaping policy.

A real-world use case was examined where DMVPN and GET VPN topologies could be combined to improve the overall user experience when using spoke-to-spoke communications over a public network. In this example, DMVPN was used to provide the GRE tunnel connections between spokes, while GET VPN was used for the encryption of traffic inside these tunnels.

Finally, it was recommended to work closely with your service provider to clearly understand what QoS models they have implemented and how their QoS marking scheme could potentially effect your traffic as it transits their network.

Additional Reading

GET VPN Cisco main page: http://www.cisco.com/go/getvpn

GET VPN Design and Implementation Guide: http://www.cisco.com/en/US/prod/collateral/vpndevc/ps6525/ps9370/ps7180/GETVPN_DIG_version_1_0_External.pdf

RFC 3547, GDOI: http://www.ietf.org/rfc/rfc3547.txt

Chapter 39

Home Office VPN QoS Case Study

With the continued success of Tifosi Software's cloud computing service offering, the company has decided to open a new call center system to help improve overall customer satisfaction and to also reduce costs. In the past, Tifosi had outsourced call center operations to a reputable company in Asia, but with new technologies becoming available, CIO Rod Bratton felt that there was a strong business case to use internal call center agents in a work-at-home scenario. In total, they would like to have 200 call center agents, with 100 of them in North America, 50 in Asia, and 50 in Europe, using a "follow-the-sun" model. The intention is that all call center agents will work from their home offices to help further reduce costs.

Building the Technical Solution

Although Rod is confident in the overall business case of bringing the call center operations in-house and allowing the agents to work from home, he was unsure how the solution could be supported technically. The key concern is how to keep a consistent customer experience if they were to use work-at-home agents who connect over the public Internet. In particular, Rod is concerned that callers might be affected by annoying things like poor voice or video quality, thus resulting in an overall negative impression of Tifosi's ability to support their customers.

Rod discussed this concern with Tifosi's senior network architect, Tom Spaghetti, and asked him to come up with a solution that would support the key requirement for this proposal. Because the Tifosi's call center agents will be connecting through local Internet connections, Tom suggested that a Cisco DMVPN solution would likely be the best approach, as it would be easy to manage, allow for a plug-and-play deployment of the home office routers, and it supports per-tunnel QoS at the headend.

Tom's team recommended that Tifosi follow closely Cisco's Virtual Office (CVO) deployment model, where a small IOS router could be deployed at the remote worker's home and be daisy-chained behind the router provided by the local ISP, as illustrated in Figure

39-1. The team recommended using a redundant pair of ASR 1002 routers at the corporate head office and a single ISR 881 router for each home call center agent.

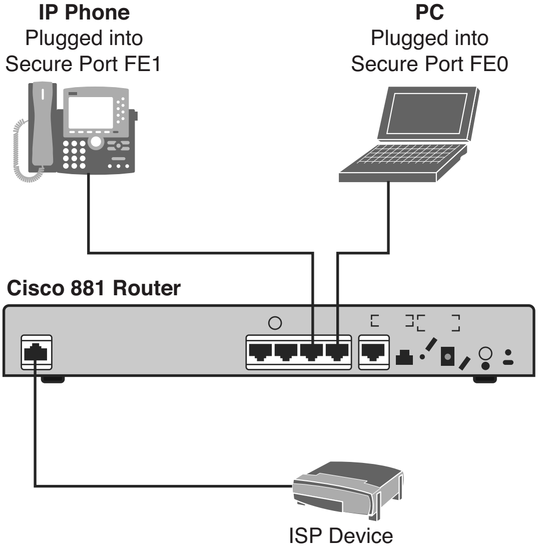


Figure 39-1 *Tifosi's Proposed Virtual Office Home Setup*

The QoS Application Requirements

To offer a full-service experience to Tifosi's customers, it is agreed that the home office solution will require support for the following applications:

- Cisco Agent Desktop (CAD)
- Cisco IP phone using an iLBC codec
- WebEx with video capabilities
- Cisco TelePresense video conferencing
- Cisco Jabber for internal chat and IP softphone

To support these various voice, video, and data communications requirements, it is agreed that the minimum bandwidth for each home agent should be no less than 10 Mbps in both the upstream and downstream directions. It is also recognized that although the majority of ISPs will not be able to support any kind of QoS on the Internet, to handle possible congestion issues in the Tifosi routers themselves it is be necessary to implement a QoS policy both on the headend router (facing into the DMVPN network) and on the

881 spoke routers (facing toward the ISP provided router at the home). Because home agents are going to be using a variety of different multimedia applications, it is recommended to proceed with an eight-class QoS model, both on the home routers and on the DMVPN hub router. This has two key benefits:

- It maintains a similar QoS policy to what is being used by the rest of the Tifosi network.
- This model supports their key application requirements for the remote call center agents.

To summarize, the eight-class model is set up as follows, both on the headend router and the ISR 881 at the agent's home office:

- **Voice:** Marked EF and provisioned with an EF PHB with 10 percent strict-priority queuing.
- **Real-Time Interactive:** (To support on-demand TelePresense video) Marked CS4, yet provisioned with an EF PHB with 23 percent strict-priority queuing.
- **Signaling:** (To support EIGRP routing used on the DMVPN tunnels) Marked CS3 and provisioned with 2 percent guaranteed bandwidth.
- **Multimedia Conferencing:** (To support Cisco Jabber, WebEx and other video-conference needs) Marked AF41 and treated with an AF PHB, provisioned with guaranteed bandwidth allocation of 10 percent, with DSCP-based WRED enabled.
- **Transactional Data:** (Used primarily for Cisco Agent Desktop) Marked AF21 and treated with an AF PHB, provisioned with guaranteed bandwidth allocation of 25 percent, with DSCP-based WRED enabled.
- **Bulk Data:** Marked AF11 and treated with an AF PHB, provisioned with guaranteed bandwidth allocation of 4 percent, with DSCP-based WRED enabled.
- **Scavenger:** Marked CS1 and treated with a bandwidth-constrained queue provisioned at 1 percent.
- **Best Effort:** All remaining applications continue to receive a default service that would (collectively) amount to no less than a quarter of any given link's bandwidth.

The QoS Configuration

Although it is assumed that the ISPs themselves are not be able to provide any level of QoS over the Internet, each router in this network needs to support congestion management towards the Internet. The configuration for the DMVPN network is configured both at the headend ASR 1002 router and at the ISR 881 router at the call center agent's home office.

Headend Router Configuration

The headend router QoS configuration follows very closely with the solution that was outlined in Chapter 37, “DMVPN QoS Design.” This solution relies on the per-tunnel (per-SA) QoS mechanism for each remote site. To do this, a parent policy shaper is applied to the mGRE tunnel interface such that when congestion is signaled, the eight-class QoS policy is used. In this case, there is only one type of user, so the eight-class model is the only one that will be implemented. In this case, the aggregate WAN bandwidth at the headend is modeled at 1 Gbps, which is shared with the data center. It is expected that this aggregate bandwidth is sufficient to support the call center home agents.

The first step is to create the class maps based on the eight-class model shown previously, as demonstrated in Example 39-1.

Example 39-1 *Define the Class Maps (Default Is Already Defined)*

```
ASR1000(config)# class-map match-all VOICE
ASR1000(config-cmap)# match dscp ef
ASR1000(config-cmap)# class-map match-all REALTIME-INTERACTIVE
ASR1000(config-cmap)# match dscp cs4
ASR1000(config-cmap)# class-map match-any SIGNALING
ASR1000(config-cmap)# match dscp cs3
ASR1000(config-cmap)# class-map match-all MULTIMEDIA-CONFERENCING
ASR1000(config-cmap)# match dscp af41
ASR1000(config-cmap)# class-map match-any TRANSACTIONAL-DATA
ASR1000(config-cmap)# match dscp af21
ASR1000(config-cmap)# class-map match-any BULK-DATA
ASR1000(config-cmap)# match dscp af11
ASR1000(config-cmap)# class-map match-any SCAVANGER
ASR1000(config-cmap)# match dscp cs1
 ! Defines the 8 classes on the headend
```

Now that the class maps have been created, you need to create the matching child policy that will use these classes, as demonstrated in Example 39-2. Note that the child policy needs to be configured before the parent policy; otherwise, the router will reject the unconfigured child policy.

Example 39-2 *Create the Child Policy Map*

```
ASR1000(config)# policy-map 8-CLASS-CHILD
ASR1000(config-pmap)# class VOICE
ASR1000(config-pmap-c)# priority percent 10
 ! Assign voice 10 % in strict-priority queue
ASR1000(config-pmap-c)# class REALTIME-INTERACTIVE
```

```
ASR1000(config-pmap-c)# priority 23
 ! Assign voice 23 % in strict-priority queue
ASR1000(config-pmap-c)# class SIGNALING
ASR1000(config-pmap-c)# bandwidth percent 2
 ! Signaling is given 2%
ASR1000(config-pmap-c)# class MULTIMEDIA-CONFERENCING
ASR1000(config-pmap-c)# bandwidth percent 10
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect dscp-based
 ! Multimedia conf is given 10% WFQ with WRED based on DSCP
ASR1000(config-pmap-c)# class TRANSACTIONAL-DATA
ASR1000(config-pmap-c)# bandwidth percent 25
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect dscp-based
 ! Transactional data is given 25% WFQ with WRED based on DSCP
ASR1000(config-pmap-c)# class BULK-DATA
ASR1000(config-pmap-c)# bandwidth percent 4
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect dscp-based
 ! Bulk data is only given 4% WFQ with WRED
ASR1000(config-pmap-c)# class SCAVENGER
ASR1000(config-pmap-c)# bandwidth percent 1
 ! Scavenger is given only 1% of available BW
ASR1000(config-pmap-c)# class class-default
ASR1000(config-pmap-c)# bandwidth percent 25
ASR1000(config-pmap-c)# fair-queue
ASR1000(config-pmap-c)# random-detect
 ! Best effort in the default class is given 25% WFQ with WRED
```

The third step is to create the parent policy map that is key to generating artificial backpressure for each spoke in the DMVPN network, as demonstrated in Example 39-3.

Example 39-3 *Create the Parent Policy Maps*

```
ASR1000(config)# policy-map 8-CLASS-PARENT
ASR1000(config-pmap)# class class-default
ASR1000(config-pmap-c)# shape average 10000000
 ! Create backpressure at 10 Mbps downstream to each agent
ASR1000(config-pmap-c)# service-policy 8-CLASS-CHILD
  ! Use a nested service policy to call the "child" policy
```

Once the policy and class maps have been created, it is important to verify that they have been configured as expected. This can be done with the following commands:

- **show class-map**
- **show policy-map**

The final step is to apply the parent policy map to the mGRE interface, as demonstrated in Example 39-4

Example 39-4 *Apply the Policy Map to the mGRE Interface*

```
ASR1000(config)# interface Tunnel10
ASR1000(config-if)# ip address 10.1.1.254 255.255.255.0
 !
<snip>
 !
ASR1000(config-if)# ip nhrp map group GROUP1 service-policy output 8-CLASS-PARENT
 ! Associate the policy map to the NHRP Group
ASR1000(config-if)# ip nhrp network-id 12300
ASR1000(config-if)# tunnel source GigabitEthernet0/0/0
ASR1000(config-if)# tunnel mode gre multipoint
ASR1000(config-if)# tunnel key 3210
ASR1000(config-if)# tunnel protection ipsec profile TIFOSI
```

You can verify the configuration in Example 39-4 with the **show policy-map interface** command. Note that in this scenario there is no outbound QoS policy attached to the physical interface connected to the WAN. The ASR 1002 used in this case is connected at a physical speed of 1 Gbps and is also serviced by Tifosi's ISP at 1 Gbps, meaning that no bandwidth mismatch is present. It is noted that a QoS policy may be used on the physical interface in the future to address any oversubscription issues if they arise, but the initial QoS implementation does not include this in the design.

Home Office Router (Spoke) Configuration

As you will have noted, the procedure just outlined follows exactly the process that was demonstrated in Chapter 37, for configuring the headend DMVPN router. We will now look at how to configure QoS on the spoke router.

The main challenge at the branch office is that there is almost always going to be a mismatch of speeds between the home office router and the bandwidth available to access the Internet. For example, the ISR 881 router has a 100-Mbps physical interface connection to the ISP provided router; however, Tifosi has only purchased a 10-Mbps connection from the ISP. It is clear from this mismatch that the ISR 881 could potentially send more than 10 Mbps up to the Internet and not realize that the upstream router has encountered congestion (especially when doing video), and thus seriously hamper the quality of the applications and the user experience. So although it might not be possible to have any special QoS handling over the Internet, Tifosi can at least ensure that the home users do not trip over themselves as they run out the front door!

It is important to remember that in IOS a router never starts using the QoS congestion management toolset until it detects that the interface has encountered congestion. For example, if the interface is running at 100 Mbps, but the router is only sending something smaller, say 15 Mbps, QoS is not invoked because the router does not see any need for QoS; it simply doesn't see any congestion because no packets are being dropped. In this case, packets will be dropped only where congestion is actually encountered, which is at connection between the ISP-provided router and the Internet. You might think that an easy way to solve this issue is to just reduce the bandwidth on the WAN interface of the Cisco 881 router using the **bandwidth** *xxx* command on the interface. Unfortunately, this command doesn't actually change the bandwidth of the interface, and does nothing to solve your problem. (The **bandwidth** statement on the interface is primarily used to influence routing protocol path calculations.) The solution here is to use a hierarchical shaper, similar to what was used at the headend router. Figure 39-2 illustrates this issue.

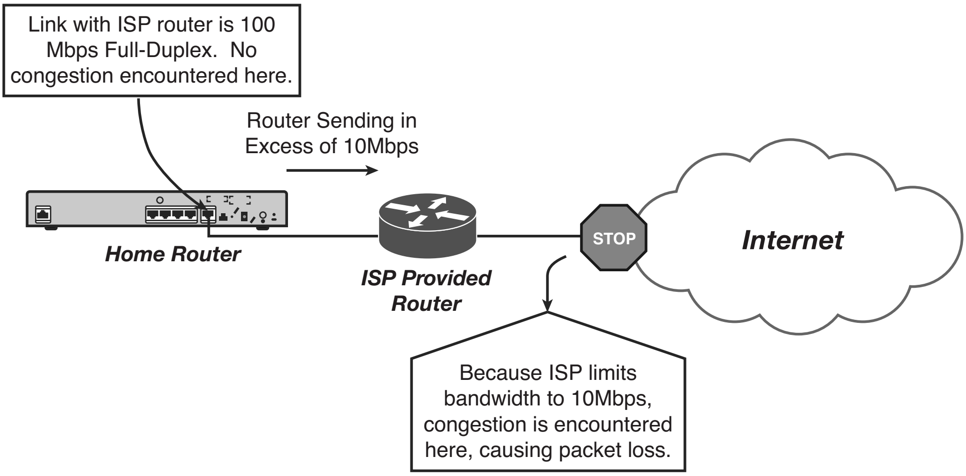

Link with ISP router is 100 Mbps Full-Duplex. No congestion encountered here.
Router Sending in Excess of 10Mbps
Home Router
ISP Provided Router
STOP
Internet
Because ISP limits bandwidth to 10Mbps, congestion is encountered here, causing packet loss.

Figure 39-2 *Dealing with Bandwidth Constraints in Home Networks*

To begin with, configure the same class and policy maps that were used on the headend router. The configuration in Example 39-5 should be enabled on each spoke router. Again, the eight-class model is used at the branch office router.

Example 39-5 *Create the Class Maps According to the Eight-Class Model Previously Defined*

```
ISR881(config)# class-map match-all VOICE
ISR881(config-cmap)# match dscp ef
ISR881(config-cmap)# class-map match-all REALTIME-INTERACTIVE
ISR881(config-cmap)# match dscp cs4
ISR881(config-cmap)# class-map match-any SIGNALING
ISR881(config-cmap)# match dscp cs3
```

```
ISR881(config-cmap)# class-map match-all MULTIMEDIA-CONFERENCING
ISR881(config-cmap)# match dscp af41
ISR881(config-cmap)# class-map match-any TRANSACTIONAL-DATA
ISR881(config-cmap)# match dscp af21
ISR881(config-cmap)# class-map match-any BULK-DATA
ISR881(config-cmap)# match dscp af11
ISR881(config-cmap)# class-map match-any SCAVANGER
ISR881(config-cmap)# match dscp cs1
 ! Defines the eight classes on each spoke router
```

As before, the next step is to configure the child policy map that is used to provide CBWFQ to the different applications once congestion is detected from the parent shaper. Example 39-6 demonstrates this configuration.

Example 39-6 *Create the Child Policy Map According to the Eight-Class Model Previously Defined*

```
ISR881(config)# policy-map 8-CLASS-CHILD
ISR881(config-pmap)# class VOICE
ISR881(config-pmap-c)# priority percent 10
 ! Assign voice 10 % in strict-priority queue
ISR881(config-pmap-c)# class REALTIME-INTERACTIVE
ISR881(config-pmap-c)# priority 23
 ! Assign voice 23 % in strict-priority queue
ISR881(config-pmap-c)# class SIGNALING
ISR881(config-pmap-c)# bandwidth percent 2
 ! Signaling is given 2%
ISR881(config-pmap-c)# class MULTIMEDIA-CONFERENCING
ISR881(config-pmap-c)# bandwidth percent 10
ISR881(config-pmap-c)# fair-queue
ISR881(config-pmap-c)# random-detect dscp-based
 ! Multimedia conf is given 10% WFQ with WRED based on DSCP
ISR881(config-pmap-c)# class TRANSACTIONAL-DATA
ISR881(config-pmap-c)# bandwidth percent 25
ISR881(config-pmap-c)# fair-queue
ISR881(config-pmap-c)# random-detect dscp-based
 ! Transactional data is given 25% WFQ with WRED based on DSCP
ISR881(config-pmap-c)# class BULK-DATA
ISR881(config-pmap-c)# bandwidth percent 4
ISR881(config-pmap-c)# fair-queue
ISR881(config-pmap-c)# random-detect dscp-based
 ! Bulk data is only given 4% WFQ with WRED
ISR881(config-pmap-c)# class SCAVENGER
ISR881(config-pmap-c)# bandwidth percent 1
```

```
 ! Scavenger is given only 1% of available BW
ISR881(config-pmap-c)# class class-default
ISR881(config-pmap-c)# bandwidth percent 25
ISR881(config-pmap-c)# fair-queue
ISR881(config-pmap-c)# random-detect
 ! Best effort in the default class is given 25% WFQ with WRED
```

Now that the class maps and parent policy map have been created, configure the hierarchical parent shaper that enforce the artificial backpressure at 10 Mbps to ensure that congestion is not encountered on the link to the ISP, as demonstrated in Example 39-7.

Example 39-7 *Create the Parent Policy Map*

```
ISR881(config)# policy-map 8-CLASS-PARENT
ISR881(config-pmap)# class class-default
 ! create backpressure at 10 Mbps downstream to each agent
ISR881(config-pmap-c)# shape average 10000000
 ! use a nested service policy to call the "child" policy
ISR881(config-pmap-c)# service-policy 8-CLASS-CHILD
```

After the policy and class maps have all been created, it is important to verify that they have been configured as expected. You can do so with the following commands:

- **show class-map**
- **show policy-map**

After the parent policy has been verified, apply it to the WAN interface on the home office router, as demonstrated in Example 39-8. Note that here you do not need to apply it to the DMVPN tunnel interface, but rather to the physical interface. In this case, the most important thing is not the exceed traffic inside the tunnel, but rather the total amount of data leaving the physical interface. With the automatic function of IOS to copy the DSCP value to the outer IP header, the policy map on the physical interface is able to correctly classify and queue traffic as necessary.

As a final step, it is important to add the NHRP group name for this spoke type to the mGRE tunnel interface so that the hub router knows what QoS policy to apply in the downstream direction as traffic flows from hub to spoke. In the case of this spoke, demonstrated in Example 39-8, the spoke name GROUP1 is used. This matches the NHRP name configured on the hub router QoS service policy.

Example 39-8 *Apply the Policy Map to the Physical Interface*

```
ISR881(config)# interface FastEthernet 0/0
ISR881(config-if)# ip address 192.168.1.44 255.255.255.0
```

```
ISR881(config-if)# service-policy output 8-CLASS-PARENT
 ! This section attaches the outbound service policy to the physical interface
 !
ISR881(config)# interface tunnel 1
ISR881(config-if)# ip address 10.1.1.22 255.255.255.0
ISR881(config-if)# ip nhrp group GROUP1
 ! This section is used to tell the hub router the NHRP group membership of this
 ! spoke.
```

Now that the service policy has been added to the tunnel interface of the home office router, confirm the configuration is correct by using the **show policy-map interface x/y** command.

Summary

Through the use of a case study, this chapter discussed key design criteria for applying QoS to a home VPN router, in particular a corporate home teleworker that is working as a call center agent. This chapter expanded on the DMVPN example discussed in Chapter 37 and demonstrated how to correctly implement the eight-class QoS model both at the hub and at the spoke routers.

In the case of the home teleworker, a new challenge was addressed where a speed mismatch exists between the spoke router and the offered bandwidth from the local ISP. This chapter discussed the method of using a hierarchical shaper to create artificial backpressure on the physical interface so that congestion could be correctly handled at the ISR router. This approach helps to prevent packet loss at the uplink to the Internet at the ISP-provided router, although it is not able to help with packet loss on the Internet itself.

Additional Reading

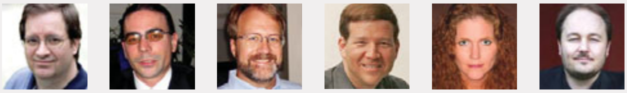
CVO main page: http://www.cisco.com/go/cvo

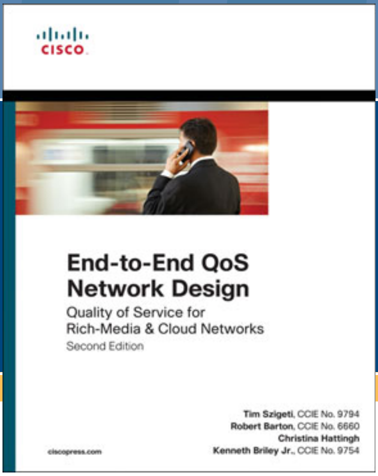
CVO Solutions Guide: http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns430/guide_c07-683001.html

CVO Deployment Guide: http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns430/ns855/deployment_guide_c22-493157.html

Index

Implementing Cisco Unified Communications Voice over IP and QoS (CVOICE)
Cisco Unity Connection
Official Cert Guide
CCNP Security SECURE 642-637
IPv6 for Enterprise Networks
Configuring Cisco Unified Communications Manager and Unity Connection





CISCO

End-to-End QoS Network Design

Quality of Service for Rich-Media & Cloud Networks

Second Edition

Tim Szigeti, CCIE No. 9794
Robert Barton, CCIE No. 6660
Christina Hattingh
Kenneth Briley Jr., CCIE No. 9754

ciscopress.com

Safari® Books Online

FREE Online Edition

Your purchase of ***End-to-End QoS Network Design*** includes access to a free online edition for 45 days through the **Safari Books Online** subscription service. Nearly every Cisco Press book is available online through **Safari Books Online**, along with thousands of books and videos from publishers such as Addison-Wesley Professional, Exam Cram, IBM Press, O'Reilly Media, Prentice Hall, Que, Sams, and VMware Press.

Safari Books Online is a digital library providing searchable, on-demand access to thousands of technology, digital media, and professional development books and videos from leading publishers. With one monthly or yearly subscription price, you get unlimited access to learning tools and information on topics including mobile app and software development, tips and tricks on using your favorite gadgets, networking, project management, graphic design, and much more.

Activate your FREE Online Edition at informit.com/safarifree

STEP 1: Enter the coupon code: KCMPXAA.

STEP 2: New Safari users, complete the brief registration form. Safari subscribers, just log in.

If you have difficulty registering on Safari or accessing the online edition, please e-mail customer-service@safaribooksonline.com

Addison Wesley · Adobe Press · ALPHA · Cisco Press · FT Press FINANCIAL TIMES · IBM Press · Microsoft Press · New Riders · O'REILLY

Peachpit Press · PRENTICE HALL · que · Redbooks · SAMS · sas Publishing · vmware PRESS · WILEY · wrox

Appendix A

AutoQoS for Medianet

As of August 2010, an updated version of AutoQoS was released for the Catalyst 2960-G/S, 3560-G/E/X, and 3750-G/E/X family of switches (with IOS Release 12.2(55)SE). This release was directly based on the recommendations put forward in the Enterprise QoS SRND, Release 4.0 to support rich-media applications; in fact, the global configuration command for this version of AutoQoS is **auto qos srnd4**.

In addition, in October 2012 a corresponding (Modular QoS command-line interface [MQC]-based) version of AutoQoS SRND4 was released for the Catalyst 4500 series switches as part of Cisco IOS 15.1(1)SG and IOS XE 3.3.0SG.

This appendix discusses the functionality and options of this latest AutoQoS feature, in addition to the details the QoS configurations that it automatically generates.

Note Because of the differences between Multi-Layer Switch (MLS)-QoS and MQC-based QoS, not all AutoQoS models and features are identically supported on the Catalyst 2K/3K as compared to the Catalyst 4K.

Note Some configuration variations may exist between the recommendations in the QoS SRND 4.0 (and this book) as compared with the AutoQoS-generated commands; these variations are relatively minor and are primarily the result of the product teams engineering preferences/constraints.

AutoQoS SRND4 Models for Cisco Catalyst 2960/3560/3750 Series Switches

AutoQoS-SRND4, which can be shortened to AutoQoS for the sake of simplicity, presents the network administrator with four main ingress QoS policy options in interface configuration mode:

- **auto qos trust** {**cos** | **dscp**}: This option configures the port to statically trust either class of service (CoS) or differentiated services code point (DSCP). If neither CoS nor DSCP are explicitly specified, the **auto qos trust** command will configure, by default, CoS trust on Layer 2 switch ports and DSCP trust on Layer 3 routed interfaces.
- **auto qos video** [**cts** | **ip-camera**]: This new option provides automatic configuration support for both Cisco TelePresence Systems (via the **cts** keyword) in addition to IP video-surveillance cameras (via the **ip-camera** keyword).
- **auto qos classify** {**police**}: This option provides a generic template that can classify and mark up to six classes of Medianet traffic and can optionally provision data plane policing/scavenger-class QoS policy elements for these traffic classes (via the optional **police** keyword).
- **auto qos voip** [**cisco-phone** | **cisco-softphone** | **trust**]: This option provides not only legacy support for Auto QoS VoIP IP telephony deployments but also expands on these models to include provisioning for additional classes of rich-media applications and to include data plane policing/scavenger-class QoS policy elements to protect and secure these applications.

Each ingress option is automatically complemented by a complete set of ingress and egress queuing configurations, complete with both CoS- and DSCP-to-queue mappings, as shown in Figure A-1.

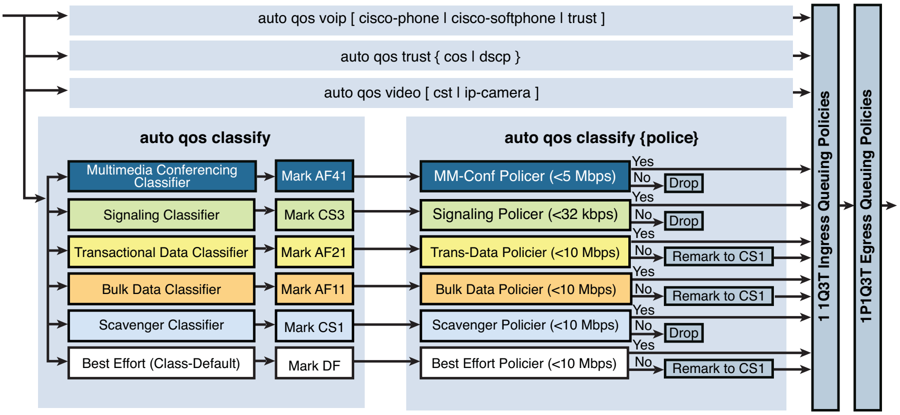

auto qos voip [cisco-phone | cisco-softphone | trust]
auto qos trust { cos | dscp }
auto qos video [cst | ip-camera]
auto qos classify
Multimedia Conferencing Classifier
Mark AF41
Signaling Classifier
Mark CS3
Transactional Data Classifier
Mark AF21
Bulk Data Classifier
Mark AF11
Scavenger Classifier
Mark CS1
Best Effort (Class-Default)
Mark DF
auto qos classify {police}
MM-Conf Policer (<5 Mbps)
Yes
No
Drop
Signaling Policer (<32 kbps)
Yes
No
Drop
Trans-Data Policier (<10 Mbps)
Yes
No
Remark to CS1
Bulk Data Policier (<10 Mbps)
Yes
No
Remark to CS1
Scavenger Policier (<10 Mbps)
Yes
No
Drop
Best Effort Policier (<10 Mbps)
Yes
No
Remark to CS1
1 1Q3T Ingress Queuing Policies
1P1Q3T Egress Queuing Policies

Figure A-1 *AutoQoS SRND4 Models*

The complete configurations provisioned by each of these new AutoQoS model options, along with the complete ingress and egress queuing configurations, are detailed in the following sections. For the sake of logical development, however, **auto qos voip** model is discussed last, because it combines several policy elements from other AutoQoS model options.

AutoQoS Trust Models

The **auto qos trust** interface command configures static trust policies on the ports or interfaces that it is configured on. If the port is operating as a Layer 2 switch port, (by default) CoS trust is configured; whereas, if the port is operating as a Layer 3 routed interface, (by default) DSCP trust is configured.

Example A-1 shows the configuration of AutoQoS trust on a Layer 2 switch port.

Example A-1 *AutoQoS Trust Applied on a Layer 2 Switch Port*

```
! This section configures autoqos trust on a L2 switch port
C3750(config)# interface GigabitEthernet1/0/1
C3750(config-if)# description L2-ACCESS-PORT
C3750(config-if)# switchport access vlan 10
C3750(config-if)# switchport voice vlan 110
C3750(config-if)# spanning-tree portfast
C3750(config-if)# auto qos trust
 ! Autoconfigures static trust policy
 ! (+ ingress and egress queuing policies)
```

You can verify the effect of this **auto qos trust** policy on a Layer 2 switch port by the **show run interface** command, as shown in Example A-2.

Example A-2 *AutoQoS Trust Applied on a Layer 2 Switch Port Verification:* ***show run interface***

```
C3750# show run interface GigabitEthernet1/0/1
Building configuration...
Current configuration : 251 bytes
!
interface GigabitEthernet1/0/1
 description L2-ACCESS-PORT
 switchport access vlan 10
 switchport voice vlan 110
 srr-queue bandwidth share 1 30 35 5
 queue-set 2
 priority-queue out
 mls qos trust cos
```

```
  ! AutoQoS has configured the port to static CoS-trust
 auto qos trust
 spanning-tree portfast
end
C3750#
```

AutoQoS Video Models

Besides supporting IP telephony devices such as Cisco IP phones and softphones (via AutoQoS-VoIP), AutoQoS now also supports video devices, such as Cisco TelePresence Systems (CTS) and IP video-surveillance cameras, both of which support conditional trust via CDP-negotiation.

Cisco TelePresence Systems (CTS) can mark their video flow and their audio flows with to CoS 4 and DSCP CS4. In addition, any voice traffic originating from the Cisco 7975G IP Phone, which is an integral part of the CTS, is marked to CoS 5 and DSCP EF. Furthermore, any signaling traffic—whether for the CTS or the IP phone—is marked CoS 3 and DSCP CS3.

Similar to **auto qos trust** behavior, **auto qos video cts** dynamically extends CoS trust to CTS systems connecting to Layer 2 switch ports (by default) and dynamically extends DSCP trust to CTS systems connecting to Layer 3 routed interfaces (by default).

CTS systems are usually connected to Layer 2 switch ports, however, as shown in Example A-3.

Example A-3 *AutoQoS Video CTS Configuration on a Layer 2 Switch Port*

```
C3750(config)# interface GigabitEthernet1/0/1
C3750(config-if)# description L2-ACCESS-PORT-TO-CTS
C3750(config-if)# switchport access vlan 10
C3750(config-if)# switchport voice vlan 110
C3750(config-if)# spanning-tree portfast
C3750(config-if)# auto qos video cts
  ! Autoconfigures conditional-trust policy for CTS
  ! (+ ingress and egress queuing policies)
```

Nonetheless, should an administrator choose to trust DSCP instead of CoS, he can still do so while using the **auto qos video cts** command, simply by manually adding an **mls qos trust dscp** interface command to the configuration.

This design option demonstrates a simple, yet powerful point: AutoQoS configurations may be modified and tailored to specific administrative needs or preferences. In other words, deploying AutoQoS is not an "all-or-nothing" option, but rather one that may be viewed as a generic template on which custom-tailored designs may be overlaid. Even with a moderate amount of manual configuration, AutoQoS can still significantly expedite Medianet QoS deployments and greatly reduce manual configuration errors in the process.

Unlike CTS devices, IP video-surveillance cameras are only required to mark their video (and if supported, audio) flows at Layer 3 (typically to DSCP CS5/40). This allows for more flexible deployment models because these cameras do not therefore have to be deployed in dedicated VLANs connecting to the access switch via an 802.1Q trunk. Therefore, the **auto qos video ip-camera** interface command dynamically extends DSCP trust to such devices, after these have successfully identified themselves to the switch via CDP. DSCP trust is dynamically extended whether the port is configured as a Layer 2 switch port or as a Layer 3 routed interface, as shown in Example A-4.

Example A-4 *AutoQoS Video IP Camera Configuration on a Layer 2 Switch Port*

```
C3750(config)# interface GigabitEthernet1/0/1
C3750(config-if)# description L2-ACCESS-PORT-TO-IPVS-CAMERA
C3750(config-if)# switchport access vlan 10
C3750(config-if)# switchport voice vlan 110
C3750(config-if)# spanning-tree portfast
C3750(config-if)# auto qos video ip-camera
 ! Auto-configures conditional-trust policy for IPVS
 ! (+ ingress and egress queuing policies)
```

You can verify the effect of this **auto qos video ip-camera** policy on a Layer 2 switch port by the **show run interface** command, as shown in Example A-5.

Example A-5 *AutoQoS Video IP Camera Applied on a Layer 2 Switch Port Verification: **show run interface***

```
C3750# show run interface GigabitEthernet 1/0/1
Building configuration...
Current configuration : 309 bytes
!
interface GigabitEthernet1/0/1
 description L2-ACCESS-PORT-TO-IPVS-CAMERA
 switchport access vlan 10

 srr-queue bandwidth share 1 30 35 5
 queue-set 2
 priority-queue out
 mls qos trust device ip-camera
  ! AutoQoS conditional-trust policy for ip-camera devices
 mls qos trust dscp
```

```
  ! AutoQoS has configured DSCP trust to be dynamically extended
 auto qos video ip-camera
 spanning-tree portfast
end
```

In a similar vein to the CTS (DSCP trust) example, should an administrator want to extend CoS trust instead of DSCP trust to IPVS cameras, he could add **mls qos trust cos** to the **auto qos video ip-camera** interface configuration.

AutoQoS Classify and Police Models

The AutoQoS classify and police models provide a generic template to support additional rich-media and data applications, providing a classification (and optional policing) model for these. These models are most suitable for switch ports connecting to PC endpoint devices.

Six application classes (multimedia Conferencing, Signaling, Transactional Data, Bulk Data, Scavenger, and Best-Effort) are automatically defined via class maps. Each class map references an associated extended IP access list. These IP access lists define the TCP and UDP port numbers of the given class of applications are based on sample ports. *However, it cannot be overemphasized that these are just generic application examples for these classes and the administrator can add/change/delete the access list entries to match on their specific applications.*

Example A-6 shows the application of the **auto qos classify** command on a Layer 2 switch port.

Example A-6 *AutoQoS Classify Configuration on a Layer 2 Switch Port*

```
C3750(config)# interface GigabitEthernet1/0/1
C3750(config-if)# description L2-ACCESS-PORT-TO-PC
C3750(config-if)# switchport access vlan 10
C3750(config-if)# spanning-tree portfast
C3750(config-if)# auto qos classify
 ! Autoconfigures classify policy
 ! (+ ingress and egress queuing policies)
```

You can verify the effect of this **auto qos classify** policy on a Layer 2 switch port by the **show run** command, as shown in Example A-7.

Example A-7 *AutoQoS Classify Configuration on a Layer 2 Switch Port Verification: show run*

```
C3750# show run
Building configuration...
```

```
<snip>
! This section defines the class maps for AutoQoS-Classify
! Each class map is associated with an extended IP access list
class-map match-all AUTOQOS_MULTIENHANCED_CONF_CLASS
 match access-group name AUTOQOS-ACL-MULTIENHANCED-CONF
class-map match-all AUTOQOS_DEFAULT_CLASS
 match access-group name AUTOQOS-ACL-DEFAULT
class-map match-all AUTOQOS_TRANSACTION_CLASS
 match access-group name AUTOQOS-ACL-TRANSACTIONAL-DATA
class-map match-all AUTOQOS_SIGNALING_CLASS
 match access-group name AUTOQOS-ACL-SIGNALING
class-map match-all AUTOQOS_BULK_DATA_CLASS
 match access-group name AUTOQOS-ACL-BULK-DATA
class-map match-all AUTOQOS_SCAVANGER_CLASS
 match access-group name AUTOQOS-ACL-SCAVANGER
!
! This section defines the policy map for AutoQoS-Classify
policy-map AUTOQOS-SRND4-CLASSIFY-POLICY
 class AUTOQOS_MULTIENHANCED_CONF_CLASS
  set dscp af41
  ! Marks multimedia conferencing traffic to AF41
 class AUTOQOS_BULK_DATA_CLASS
  set dscp af11
  ! Marks bulk data traffic to AF11
 class AUTOQOS_TRANSACTION_CLASS
  set dscp af21
  ! Marks transactional data traffic to AF21
 class AUTOQOS_SCAVANGER_CLASS
  set dscp cs1
  ! Marks scavenger traffic to CS1
 class AUTOQOS_SIGNALING_CLASS
  set dscp cs3
  ! Marks signaling traffic to CS3
 class AUTOQOS_DEFAULT_CLASS
  set dscp default
  ! An explicit default class marks best effort traffic to DF
!
<snip>
! This section applies the AutoQoS-Classify policy map to the interface
interface GigabitEthernet1/0/1
 description L2-ACCESS-PORT-TO-PC
 switchport access vlan 10

 srr-queue bandwidth share 1 30 35 5
 queue-set 2
 priority-queue out
 auto qos classify
 spanning-tree portfast
 service-policy input AUTOQOS-SRND4-CLASSIFY-POLICY
  ! Attaches the AutoQoS-Classify service policy to the interface
!
<snip>
! This section defines the extended IP access lists for AutoQoS-Classify
ip access-list extended AUTOQOS-ACL-BULK-DATA
 permit tcp any any eq 22
 permit tcp any any eq 465
 permit tcp any any eq 143
 permit tcp any any eq 993
 permit tcp any any eq 995
 permit tcp any any eq 1914
 permit tcp any any eq ftp
 permit tcp any any eq ftp-data
 permit tcp any any eq smtp
 permit tcp any any eq pop3
ip access-list extended AUTOQOS-ACL-DEFAULT
 permit ip any any
ip access-list extended AUTOQOS-ACL-MULTIENHANCED-CONF
 permit udp any any range 16384 32767
ip access-list extended AUTOQOS-ACL-SCAVANGER
 permit tcp any any range 2300 2400
 permit udp any any range 2300 2400
 permit tcp any any range 6881 6999
 permit tcp any any range 28800 29100
 permit tcp any any eq 1214
 permit udp any any eq 1214
 permit tcp any any eq 3689
 permit udp any any eq 3689
 permit tcp any any eq 11999
ip access-list extended AUTOQOS-ACL-SIGNALING
 permit tcp any any range 2000 2002
 permit tcp any any range 5060 5061
 permit udp any any range 5060 5061
ip access-list extended AUTOQOS-ACL-TRANSACTIONAL-DATA
 permit tcp any any eq 443
 permit tcp any any eq 1521
 permit udp any any eq 1521
 permit tcp any any eq 1526
 permit udp any any eq 1526
 permit tcp any any eq 1575
 permit udp any any eq 1575
 permit tcp any any eq 1630
 permit udp any any eq 1630
!
<snip>
```

As you can see from the configuration output in Example A-7, the **auto qos classify** command generates class maps, associated extended IP access lists, and a policy map that is attached to the interface (along with input and output queuing policies, which are discussed in detail a following section). Again, note that the IP access list entries shown here are based on sample *ports and are just generic application examples for these classes. You can add/change/delete the access list entries to match on your specific applications.*

In addition, should the administrator want to enable data plane policing/Scavenger class QoS policies on these application classes, he may do so by including the option keyword **police** in conjunction with the **auto qos classify** interface command, as shown in Example A-8.

Example A-8 *AutoQoS Classify and Police Configuration on a Layer 2 Switch Port*

```
C3750(config)# interface GigabitEthernet1/0/1
C3750(config-if)# description L2-ACCESS-PORT-TO-PC
C3750(config-if)# switchport access vlan 10
C3750(config-if)# spanning-tree portfast
C3750(config-if)# auto qos classify police
 ! Auto-configures classify & police policy
 ! (+ ingress and egress queuing policies)
```

You can verify the effect of this **auto qos classify police** policy on a Layer 2 switch port by the **show run** command, as shown in Example A-9.

Note For the sake of brevity and to minimize redundancy, the class maps and extended IP access lists (which are identical to those shown in Example A-7) are not repeated in future examples.

Example A-9 *AutoQoS Classify and Police Configuration on a Layer 2 Switch Port Verification: show run*

```
C3750# show run
Building configuration...
<snip>
!
! This section configures the global policed-DSCP markdown map
```

```
mls qos map policed-dscp  0 10 18 to 8
 ! DSCP 0 (DF), 10 (AF11) and 18 (AF21) are marked down to 8 (CS1)
 ! if found to be in excess of their (respective) policing rates
!
<snip>
! This section defines the policy-map for AutoQoS-Classify-Police
policy-map AUTOQOS-SRND4-CLASSIFY-POLICE-POLICY
 class AUTOQOS_MULTIENHANCED_CONF_CLASS
  set dscp af41
  police 5000000 8000 exceed-action drop
  ! Multimedia-conf is marked AF41 and policed to drop at 5 Mbps
 class AUTOQOS_BULK_DATA_CLASS
  set dscp af11
  police 10000000 8000 exceed-action policed-dscp-transmit
  ! Bulk-data is marked AF11 and policed to re-mark (to CS1) at 10 Mbps
 class AUTOQOS_TRANSACTION_CLASS
  set dscp af21
  police 10000000 8000 exceed-action policed-dscp-transmit
  ! Trans-data is marked AF21 and policed to re-mark (to CS1) at 10 Mbps
 class AUTOQOS_SCAVANGER_CLASS
  set dscp cs1
  police 10000000 8000 exceed-action drop
  ! Scavenger traffic is marked CS1 and policed to drop at 10 Mbps
 class AUTOQOS_SIGNALING_CLASS
  set dscp cs3
  police 32000 8000 exceed-action drop
  ! Signaling is marked CS3 and policed to drop at 32 Kbps
 class AUTOQOS_DEFAULT_CLASS
  set dscp default
  police 10000000 8000 exceed-action policed-dscp-transmit
  ! An explicit default class marks all other IP traffic to DF
  ! and polices all other IP traffic to re-mark (to CS1) at 10 Mbps
!
<snip>
! This section applies the AutoQoS-Classify-Police policy map to the interface
interface GigabitEthernet1/0/1
 description L2-ACCESS-PORT-TO-PC
 switchport access vlan 10
 switchport voice vlan 110
 srr-queue bandwidth share 1 30 35 5
 queue-set 2
 priority-queue out
 auto qos classify police
 spanning-tree portfast
 service-policy input AUTOQOS-SRND4-CLASSIFY-POLICE-POLICY
```

```
  ! Attaches the AutoQoS-Classify service policy to the interface
!
<snip>
```

As you can see from the configuration output in Example A-9, the two principle changes in the configuration attributable to the **police** keyword used in conjunction with the **auto qos classify** command are as follows:

- A globally defined **policed-dscp** map to mark down DF (0), AF11 (10), and AF21 (18) to CS1 (8)—if found to be exceeding their respective policing rates.
- An amended policy map that polices multimedia conferencing traffic (to drop if exceeding 5 Mbps), bulk data (to re-mark if exceeding 10 Mbps), transactional data (to re-mark if exceeding 10 Mbps), scavenger (to drop if exceeding 10 Mbps), signaling (to drop if exceeding 32 Kbps), and best-effort traffic (to re-mark if exceeding 10 Mbps).

AutoQoS VoIP Models

As with legacy AutoQoS-VoIP, there are three deployment options for AutoQoS (SRND4) VoIP: **trust**, **cisco-phone**, and **cisco-softphone**. Figure A-2 illustrates these updated **auto qos voip** deployment options—complete with ingress and egress queuing configurations.

An important point to be noted is that because the SRND4 versions of **auto qos voip** expand functionality beyond the original AutoQoS-VoIP feature, you must indicate which version of this AutoQoS-VoIP is desired. By default, simply entering **auto qos voip** interface configuration commands will invoke legacy AutoQoS-VoIP configurations. However, if you first enter **auto qos srnd4** in the global configuration command *before* applying these **auto qos voip** interface configuration commands, the SRND4 versions of **auto qos voip** will be applied.

Each of these **auto qos voip** deployment options is detailed in turn.

AutoQoS VoIP Trust Model

The first deployment option of **auto qos voip** is the **trust** option, which is effectively a legacy deployment option (because this functionality has been relegated by the previously discussed **auto qos trust** option). Like **auto qos trust**, **auto qos voip trust** configures static CoS trust on Layer 2 switch ports and static DSCP trust on Layer 3 routed interfaces. However, unlike **auto qos trust**, there is no additional **cos** or **dscp** keyword option to override these default trust settings (but this may be manually overridden with an explicitly defined **mls qos trust** [**cos** | **dscp**] interface configuration command).

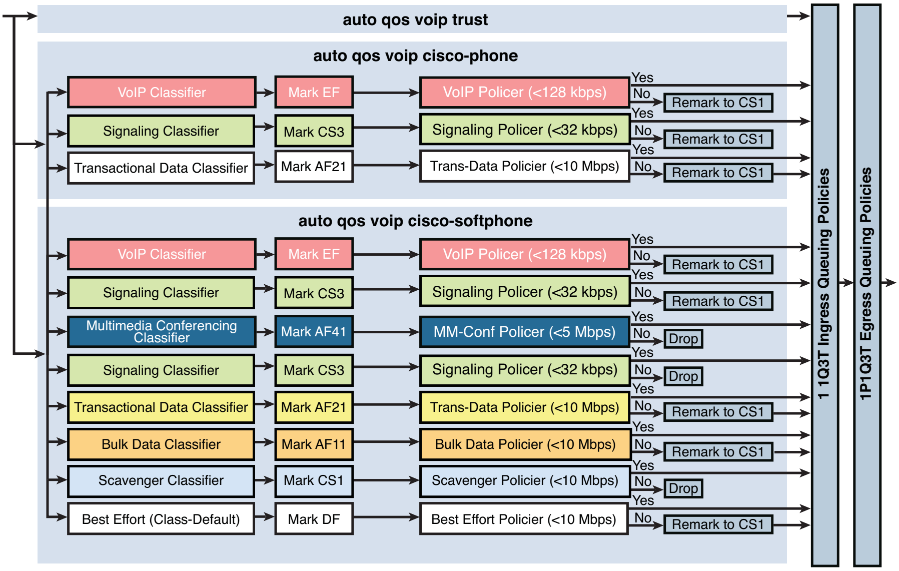

auto qos voip trust
auto qos voip cisco-phone
VoIP Classifier
Mark EF
VoIP Policer (<128 kbps)
Signaling Classifier
Mark CS3
Signaling Policer (<32 kbps)
Transactional Data Classifier
Mark AF21
Trans-Data Policier (<10 Mbps)
Yes
No
Remark to CS1
auto qos voip cisco-softphone
Multimedia Conferencing Classifier
Mark AF41
MM-Conf Policer (<5 Mbps)
Drop
Bulk Data Classifier
Mark AF11
Bulk Data Policier (<10 Mbps)
Scavenger Classifier
Mark CS1
Scavenger Policier (<10 Mbps)
Best Effort (Class-Default)
Mark DF
Best Effort Policier (<10 Mbps)
1 1Q3T Ingress Queuing Policies
1P1Q3T Egress Queuing Policies

Figure A-2 *AutoQoS SRND4 VoIP Models*

AutoQoS VoIP Cisco Phone Model

A second deployment option offered by the (SRND4) **auto qos voip** feature is to use the **cisco-phone** keyword. As previously mentioned, the administrator must first enter **auto qos srnd4** in the global configuration before entering **auto qos voip cisco-phone** on a specific interfaces. When **auto qos voip cisco-phone** is configured on a Layer 2 switch port, it dynamically extends trust CoS to Cisco IP phones; when configured on Layer 3 routed interfaces, it dynamically extends trust DSCP to Cisco IP phones. In addition, this command configures data plane policing/Scavenger class QoS policies on voice, signaling, and best effort traffic, as shown in Example A-10 and Example A-11.

Example A-10 *AutoQoS VoIP Cisco Phone (SRND4) Applied on a Layer 2 Switch Port*

```
! This section specifies that SRND4 version of AutoQoS is to be enabled
C3750(config)# auto qos srnd4
 ! Globally defines the current version of AutoQoS to be SRND4
! This section applies AutoQoS (SRND4) to a layer 2 switch port
C3750(config)# interface GigabitEthernet1/0/1
C3750(config-if)# description L2-ACCESS-PORT
C3750(config-if)# switchport access vlan 10
C3750(config-if)# switchport voice vlan 110
C3750(config-if)# spanning-tree portfast
C3750(config-if)# auto qos voip cisco-phone
 ! Auto-configures conditional-trust + marking + policing for IP Phones
 ! (+ ingress and egress queuing policies)
```

You can verify the effect of this **auto qos voip cisco-phone** policy on a Layer 2 switch port by the **show run** command, as shown in Example A-11.

Example A-11 *AutoQoS VoIP Cisco Phone (SRND4) Applied on a Layer 2 Switch Port Verification: **show run***

```
C3750# show run
Building configuration...
<snip>
! This section confirms the AutoQoS version currently enabled
auto qos srnd4
!
! This section defines the AutoQoS-VoIP-Cisco-Phone (SRND4) Class-Maps
class-map match-all AUTOQOS_VOIP_DATA_CLASS
 match ip dscp ef
 ! Voice is matched on DSCP EF
class-map match-all AUTOQOS_DEFAULT_CLASS
 match access-group name AUTOQOS-ACL-DEFAULT
 ! An explicit default class matches all other traffic via IP ACL
class-map match-all AUTOQOS_VOIP_SIGNAL_CLASS
 match ip dscp cs3
 ! Signaling traffic is matched on CS3
!
! This section defines the AutoQoS-VoIP-Cisco-Phone (SRND4) Policy-Map
policy-map AUTOQOS-SRND4-CISCOPHONE-POLICY
 class AUTOQOS_VOIP_DATA_CLASS
  set dscp ef
  police 128000 8000 exceed-action policed-dscp-transmit
  ! Voice is marked to DSCP EF and re-marked if exceeding 128 Kbps
 class AUTOQOS_VOIP_SIGNAL_CLASS
  set dscp cs3
  police 32000 8000 exceed-action policed-dscp-transmit
  ! Signaling is marked to DSCP CS3 and policed if exceeding 32 Kbps
 class AUTOQOS_DEFAULT_CLASS
  set dscp default
  police 10000000 8000 exceed-action policed-dscp-transmit
  ! An explicit default class marks all other IP traffic to DF
  ! and polices all other IP traffic to re-mark (to CS1) at 10 Mbps
!
! This section attaches the AutoQoS-VoIP-Cisco-Phone (SRND4) policy map to the
interface
interface GigabitEthernet1/0/1
 description L2-ACCESS-PORT
 switchport access vlan 10
 switchport voice vlan 110
 srr-queue bandwidth share 1 30 35 5
 queue-set 2
 priority-queue out
 mls qos trust device cisco-phone
  ! AutoQoS has configured a conditional-trust policy for IP Phones
 mls qos trust cos
  ! AutoQoS has configured CoS-trust to be dynamically extended
auto qos voip cisco-phone
spanning-tree portfast
 service-policy input AUTOQOS-SRND4-CISCOPHONE-POLICY
  ! Attaches the AutoQoS-VoIP-Cisco-Phone Policy-Map to the interface
!
<snip>
! This section defines the explicit-default extended IP ACL
ip access-list extended AUTOQOS-ACL-DEFAULT
 permit ip any any
!
```

Example A-11 shows that the applied version of **auto qos voip** is **srnd4** and, therefore, voice is policed to re-mark if exceeding 128 Kbps, signaling is policed to re-mark if exceeding 32 Kbps, and best effort traffic is policed to re-mark to scavenger if exceeding 10 Mbps.

AutoQoS VoIP Cisco-Softphone Model

A third deployment option offered by the (SRND4) **auto qos voip** feature is to use the **cisco-softphone** keyword. As previously mentioned, the administrator must first enter **auto qos srnd4** in the global configuration before entering **auto qos voip cisco-softphone** on specific interfaces.

In addition to the Voice and Signaling classes, six additional application classes (multimedia Conferencing, Signaling, Transactional Data, Bulk Data, Scavenger and Best Effort) are automatically defined via class maps. Each class map references an associated extended IP access list. These IP access lists define the TCP and UDP port numbers of the given class of applications, based on the sample ports. *However, it cannot be overemphasized that these are just generic application examples for these classes and the administrator can add/change/delete the access list entries to match on their specific applications.*

Example A-12 shows the application of **auto qos voip cisco-softphone** on a Layer 2 switch port interface.

Example A-12 *AutoQoS VoIP Cisco Softphone (SRND4) Applied on a Layer 2 Switch Port*

```
! This section specifies that SRND4 version of AutoQoS is to be enabled
C3750(config)# auto qos srnd4
```

```
 ! Globally defines the current version of AutoQoS to be SRND4
! This section applies AutoQoS (SRND4) to a layer 2 switch port
C3750(config)# interface GigabitEthernet1/0/1
C3750(config-if)# description L2-ACCESS-PORT
C3750(config-if)# switchport access vlan 10
C3750(config-if)# switchport voice vlan 110
C3750(config-if)# spanning-tree portfast
C3750(config-if)# auto qos voip cisco-softphone
 ! Autoconfigures conditional-trust + marking +policers for softphones
 ! (+ ingress and egress queuing policies)
```

You can verify the effect of this **auto qos voip cisco-softphone** policy on a Layer 2 switch port with the **show run** command, as shown in Example A-13.

Note For the sake of brevity and to minimize redundancy, the class maps and extended IP access lists (which are identical to those shown in Example A-7) are not repeated here.

Example A-13 *AutoQoS VoIP Cisco Softphone (SRND4) Applied on a Layer 2 Switch Port Verification: **show run***

```
C3750# show run
Building configuration...
<snip>
! This section confirms the AutoQoS version currently enabled
auto qos srnd4
!
<snip>
! This section defines the AutoQoS-VoIP-Cisco-SoftPhone policy map
policy-map AUTOQOS-SRND4-SOFTPHONE-POLICY
 class AUTOQOS_VOIP_DATA_CLASS
  set dscp ef
  police 128000 8000 exceed-action policed-dscp-transmit
  ! Voice is marked to DSCP EF and re-marked if exceeding 128 Kbps
 class AUTOQOS_VOIP_SIGNAL_CLASS
  set dscp cs3
  police 32000 8000 exceed-action policed-dscp-transmit
  ! Signaling is marked to DSCP CS3 and re-marked if exceeding 32 Kbps
 class AUTOQOS_MULTIENHANCED_CONF_CLASS
  set dscp af41
  police 5000000 8000 exceed-action drop
  ! MM-Conf is marked to DSCP AF41 and re-marked if exceeding 5 Mbps
 class AUTOQOS_BULK_DATA_CLASS
  set dscp af11
  police 10000000 8000 exceed-action policed-dscp-transmit
  ! Bulk Data is marked to DSCP AF11 and re-marked if exceeding 10 Mbps
 class AUTOQOS_TRANSACTION_CLASS
  set dscp af21
  police 10000000 8000 exceed-action policed-dscp-transmit
  ! Trans-Data is marked to DSCP AF21 and re-marked if exceeding 10 Mbps
 class AUTOQOS_SCAVANGER_CLASS
  set dscp cs1
  police 10000000 8000 exceed-action drop
  ! Scavenger is marked to DSCP CS1 and re-marked if exceeding 10 Mbps
 class AUTOQOS_SIGNALING_CLASS
  set dscp cs3
  police 32000 8000 exceed-action drop
  ! Signaling is marked to DSCP CS3 and dropped if exceeding 32 Kbps
 class AUTOQOS_DEFAULT_CLASS
  set dscp default
  ! An explicit default class marks all other IP traffic to DF
!
<snip>
! This section attaches the AutoQoS-VoIP-Cisco-SoftPhone (SRND4) policy map to the
interface
interface GigabitEthernet1/0/1
 description L2-ACCESS-PORT
 switchport access vlan 10
 switchport voice vlan 110
 srr-queue bandwidth share 1 30 35 5
 queue-set 2
 priority-queue out
 auto qos voip cisco-softphone
 spanning-tree portfast
 service-policy input AUTOQOS-SRND4-SOFTPHONE-POLICY
  ! Attaches the AutoQoS-VoIP-Cisco-SoftPhone Policy to the interface
!
```

AutoQoS 1P1Q3T Ingress Queuing Models

Example A-14 shows the AutoQoS SRND4 ingress queuing model configuration. These ingress queuing policies are automatically configured along with any other AutoQoS SRND4 QoS model.

Example A-14 *AutoQoS (SRND4) 1P1Q3T Ingress Queuing Verification:* ***show run***

```
C3750# show run
Building configuration...
```

```
<snip>
! This section displays (non-default) input queue parameters
mls qos srr-queue input bandwidth 70 30
 ! Q1 is assigned 70% BW via SRR shared weights
 ! Q2 SRR shared weight is ignored (as it has been configured as a PQ)
mls qos srr-queue input threshold 1 80 90
 ! Q1 thresholds are configured at 80% (Q1T1) and 90% (Q1T2)
 ! Q1T3 is implicitly set at 100% (the tail of the queue)
 ! Q2 thresholds are all set (by default) to 100% (the tail of Q2)
mls qos srr-queue input priority-queue 2 bandwidth 30
 ! Q2 is enabled as a strict-priority ingress queue with 30% BW
! This section displays (non-default) ingress CoS-to-Queue mappings
mls qos srr-queue input cos-map queue 1 threshold 2 3
 ! CoS value 3 is mapped to ingress Q1T2
mls qos srr-queue input cos-map queue 1 threshold 3 6 7
 ! CoS values 6 and 7 are mapped to ingress Q1T3
mls qos srr-queue input cos-map queue 2 threshold 1 4
! CoS values 4 is mapped to ingress Q2 (the PQ)
! This section displays (non-default) ingress DSCP-to-Queue mappings
mls qos srr-queue input dscp-map queue 1 threshold 2 24
 ! DSCP CS3 is mapped to ingress Q1T2
mls qos srr-queue input dscp-map queue 1 threshold 3 48 49 50 51 52 53 54 55
 ! DSCP CS6 (48) and non-standard DSCPs 49-55 are mapped to Q1T3
mls qos srr-queue input dscp-map queue 1 threshold 3 56 57 58 59 60 61 62 63
 ! DSCP CS7 (56) and non-standard DSCPs 57-63 are mapped to Q1T3
mls qos srr-queue input dscp-map queue 2 threshold 3 32 33 40 41 42 43 44 45
 ! DSCP CS4 (32), CS5 (40) and non-standard DSCPs 33-45 are mapped to Q2T3
mls qos srr-queue input dscp-map queue 2 threshold 3 46 47
 ! DSCP EF (46) and non-standard DSCP 47 are mapped to Q2T3
```

Note Ingress queuing is not supported on the Cisco Catalyst 2960-S.

AutoQoS 1P3Q3T Egress Queuing Models

Example A-15 shows the AutoQoS SRND4 egress queuing model configuration. These egress queuing policies are automatically configured along with any other AutoQoS SRND4 QoS model.

Example A-15 *AutoQoS (SRND4) 1P3Q3T Egress Queuing Verification:* ***show run***

```
C3750# show run
Building configuration...
```

```
<snip>
! This section displays (non-default) egress CoS-to-Queue mappings
mls qos srr-queue output cos-map queue 1 threshold 3 4 5
 ! CoS 4 and 5 are mapped to egress Q1T3 (the tail of the PQ)
mls qos srr-queue output cos-map queue 2 threshold 1 2
 ! CoS 2 is mapped to egress Q2T1
mls qos srr-queue output cos-map queue 2 threshold 2 3
 ! CoS 3 is mapped to egress Q2T2
mls qos srr-queue output cos-map queue 2 threshold 3 6 7
 ! CoS 6 and 7 are mapped to Q2T3
mls qos srr-queue output cos-map queue 3 threshold 3 0
 ! CoS 0 is mapped to Q3T3 (the tail of the default queue)
mls qos srr-queue output cos-map queue 4 threshold 3 1
 ! CoS 1 is mapped to Q4T3 (tail of the less-than-best-effort queue)
! This section displays (non-default) egress DSCP-to-Queue mappings
mls qos srr-queue output dscp-map queue 1 threshold 3 32 33 40 41 42 43 44 45
 ! Maps CS4 (32) and DSCPs 33-45 to Q1T3 (the tail of the PQ)
mls qos srr-queue output dscp-map queue 1 threshold 3 46 47
 ! Maps EF (46) and non-standard DSCP 47 to Q1T3 (the tail of the PQ)
mls qos srr-queue output dscp-map queue 2 threshold 1 16 17 18 19 20 21 22 23
 ! Maps CS2 (16) and AF2 (18/20/22) and DSCPs 17-23 to Q2T1
mls qos srr-queue output dscp-map queue 2 threshold 1 26 27 28 29 30 31 34 35
 ! Maps AF3 (26/28/30) and AF41 (34) and DSCPs 27-35 to Q2T1
mls qos srr-queue output dscp-map queue 2 threshold 1 36 37 38 39
 ! Maps AF42 (36) and AF43 (38) and DSCPs 37-39 to Q2T1
mls qos srr-queue output dscp-map queue 2 threshold 2 24
 ! Maps CS3 (24) to Q2T2
mls qos srr-queue output dscp-map queue 2 threshold 3 48 49 50 51 52 53 54 55
 ! Maps CS6 (48) and non-standard DSCPs 49-55 to Q2T3
mls qos srr-queue output dscp-map queue 2 threshold 3 56 57 58 59 60 61 62 63
 ! Maps CS7 (56) and non-standard DSCPs 57-63 to Q2T3
mls qos srr-queue output dscp-map queue 3 threshold 3 0 1 2 3 4 5 6 7
 ! Maps DF (0) and DSCPs 1-7 to Q3T3 (tail of best-effort queue)
mls qos srr-queue output dscp-map queue 4 threshold 1 8 9 11 13 15
 ! Maps CS1 and non-standard DSCPs 9-15 to Q4T1
mls qos srr-queue output dscp-map queue 4 threshold 2 10 12 14
 ! Maps AF1 (10/12/14) to Q4T2

! This section displays (non-default) egress queue parameters
mls qos queue-set output 1 threshold 1 100 100 50 200
 ! Q1T1 is set to 100%; Q1T2 is set to 100%;
 ! Q1 (PQ) Reserve Threshold is set to 100%;
 ! Q1 (PQ) Maximum (Overflow) Threshold is set to 200%
mls qos queue-set output 1 threshold 2 125 125 100 400
 ! Q2T1 is set to 125%; Q2T2 is set to 125%;
 ! Q2 Reserve Threshold is set to 100%;
 ! Q2 Maximum (Overflow) Threshold is set to 400%
mls qos queue-set output 1 threshold 3 100 100 100 400
 ! Q3T1 is set to 100%, Q2T2 is set to 100%
 ! Q3 Reserve Threshold is set to 100%;
 ! Q3 Maximum (Overflow) Threshold is set to 400%
mls qos queue-set output 1 threshold 4 60 150 50 200
 ! Q4T1 is set to 60%; Q4T2 is set to 150%
 ! Q4 Reserve Threshold is set to 50%;
 ! Q4 Maximum (Overflow) Threshold is set to 200%
mls qos queue-set output 1 buffers 15 25 40 20
 ! Allocates 15% of buffers to Q1; 25% to Q2; 40% to Q3 and 20% to Q4
<snip>
! This section displays (non-default) interface egress queuing settings
interface GigabitEthernet1/0/1
 description L2-ACCESS-PORT
 switchport access vlan 10
 switchport voice vlan 110
 srr-queue bandwidth share 1 30 35 5
  ! The SRR sharing weights are set to allocate 30% BW to Q2
  ! 35% BW to Q3 and 5% BW to Q4
  ! Q1 SRR sharing weight is ignored, as it will be configured as a PQ
 queue-set 2
  ! The interface(s) is assigned to queue-set 1
 priority-queue out
  ! Q1 is enabled as a strict priority queue
 mls qos trust cos
 auto qos trust
 spanning-tree portfast
!
```

AutoQoS SRND4 Models for Cisco Catalyst 4500 Series Switches

AutoQoS Version 4 (equivalent to AutoQoS SRND4 on the Catalyst 2K/3K series platforms) automatically defines eight policy maps: Three provide backward compatibility with AutoQoS-VoIP, and an additional five support AutoQoS Version 4. These policy maps are as follows:

- AutoQos-VoIP-Input-Cos-Policy
- AutoQos-VoIP-Input-Dscp-Policy
- AutoQos-VoIP-Output-Policy

- AutoQos-4.0-Input-Policy
- AutoQos-4.0-Classify-Input-Policy
- AutoQos-4.0-Cisco-Phone-Input-Policy
- AutoQos-4.0-Cisco-Softphone-Input-Policy
- AutoQos-4.0-Output-Policy

Each of these policy-maps is detailed in turn.

AutoQos-VoIP-Input-Cos-Policy

As shown in Example A-16, the AutoQos-VoIP-Input-Cos-Policy matches VoIP traffic on CoS 5 and 3 for media and signaling (respectively) and associates these with QoS groups 46 and 24 (respectively) for egress queuing (via the AutoQos-VoIP-Output-Policy).

Example A-16 *C4500 AutoQoS: AutoQos-VoIP-Input-Cos-Policy*

```
C4500# show run
Building configuration...
<snip>
 ! This section defines the AutoQos-VoIP-Input-Cos-Policy class maps
class-map match-all AutoQos-VoIP-Bearer-Cos
  match cos  5
 ! VoIP media traffic is matched on CoS 5
class-map match-all AutoQos-VoIP-Control-Cos
  match cos  3
 ! VoIP signaling traffic is matched on CoS 3

 ! This section defines the AutoQos-VoIP-Input-Cos-Policy map
policy-map AutoQos-VoIP-Input-Cos-Policy
  class AutoQos-VoIP-Bearer-Cos
   set qos-group 46
 ! VoIP media is associated with QoS-Group 46 (for queuing)
  class AutoQos-VoIP-Control-Cos
   set qos-group 24
 ! VoIP signaling is associated with QoS-Group 24 (for queuing)
```

AutoQos-VoIP-Input-Dscp-Policy

Similarly, as shown in Example A-17, the AutoQos-VoIP-Input-Dscp-Policy matches VoIP traffic on DSCP EF for media. However, it matches signaling traffic on both CS3 and AF31 (a legacy marking value for signaling). VoIP media is associated with QoS group

46, and the signaling classes are associated with QoS groups 24 and 26 (respectively) for egress queuing (via the AutoQos-VoIP-Output-Policy).

Example A-17 *C4500 AutoQoS: AutoQos-VoIP-Input-Dscp-Policy*

```
C4500# show run
Building configuration...
<snip>
 ! This section defines the AutoQos-VoIP-Input-Dscp-Policy class maps
class-map match-all AutoQos-VoIP-Bearer-Dscp
  match  dscp ef
 ! VoIP media traffic is matched on DSCP EF
class-map match-all AutoQos-VoIP-Control-Dscp26
  match  dscp af31
 ! VoIP signaling traffic is matched on DSCP AF31 (legacy marking)
class-map match-all AutoQos-VoIP-Control-Dscp24
  match  dscp cs3
 ! VoIP signaling traffic is matched on CS3

 ! This section defines the AutoQos-VoIP-Input-Dscp-Policy map
policy-map AutoQos-VoIP-Input-Dscp-Policy
  class AutoQos-VoIP-Bearer-Dscp
   set qos-group 46
 ! VoIP media is associated with QoS-Group 46 (for queuing)
  class AutoQos-VoIP-Control-Dscp26
   set qos-group 26
 ! VoIP signaling (AF31) is associated with QoS-Group 26 (for queuing)
  class AutoQos-VoIP-Control-Dscp24
   set qos-group 24
 ! VoIP signaling (CS3) is associated with QoS-Group 24 (for queuing)
```

AutoQos-VoIP-Output-Policy

As the previously defined AutoQos-VoIP-Input-Cos-Policy and AutoQos-VoIP-Input-Dscp-Policy have mapped VoIP media and signaling to respective QoS groups, these QoS groups can be mapped directly to egress queues by the AutoQos-VoIP-Output-Policy, as shown in Example A-18.

Example A-18 *C4500 AutoQoS: AutoQos-VoIP-Output-Policy*

```
C4500# show run
Building configuration...
<snip>
 ! This section defines the AutoQos-VoIP-Output-Policy class maps
```

```
class-map match-all AutoQos-VoIP-Bearer-QosGroup
  match qos-group 46
 ! VoIP media traffic is matched on QoS-group 46 for egress queuing
class-map match-all AutoQos-VoIP-Control-QosGroup26
  match qos-group 26
 ! VoIP signaling (AF31) is matched on QoS-group 26 for egress queuing
class-map match-all AutoQos-VoIP-Control-QosGroup24
  match qos-group 24
 ! VoIP signaling (CS3) is matched on QoS-group 24 for egress queuing

! This section defines the AutoQos-VoIP-Output-Policy map
policy-map AutoQos-VoIP-Output-Policy
  class AutoQos-VoIP-Bearer-QosGroup
   set dscp ef
   set cos 5
   priority
   police cir percent 33
 ! VoIP media is marked DSCP EF and CoS 5
 ! and is provisioned with a strict priority service
 ! but  limited to 33% BW
  class AutoQos-VoIP-Control-QosGroup26
   set dscp af31
   set cos 3
   bandwidth remaining percent 5
 ! VoIP signaling (AF31) is marked DSCP AF31 and CoS 3
 ! and is provisioned with a guaranteed-bandwidth service
  class AutoQos-VoIP-Control-QosGroup24
   set dscp cs3
   set cos 3
   bandwidth remaining percent 5
 ! VoIP signaling (CS3) is marked DSCP CS3 and CoS 3
 ! and is provisioned with a guaranteed-bandwidth service
  class class-default
   dbl
 ! DBL is enabled on the default class
```

AutoQos-4.0-Input-Policy

The AutoQos-4.0-Input-Policy, shown in Example A-19, is intended for ports that are connected to *trusted* endpoints, and therefore bases classification on CoS/DSCP markings. These markings are then mapped to corresponding QoS groups for egress queue mapping (via the AutoQos-4.0-Output-Policy).

Example A-19 *C4500 AutoQoS: AutoQos-4.0-Input-Policy*

```
C4500# show run
Building configuration...
<snip>
 ! This section defines the AutoQos-4.0-Input-Policy class maps
class-map match-any AutoQos-4.0-VoIP
  match dscp ef
  match cos 5
 ! VoIP is matched on DSCP EF and CoS 5
class-map match-all AutoQos-4.0-Broadcast-Vid
  match dscp cs5
 ! Broadcast Video is matched on DSCP CS5
class-map match-all AutoQos-4.0-Realtime-Interact
  match dscp cs4
 ! Realtime Interactive is matched on DSCP CS4
class-map match-all AutoQos-4.0-Network-Ctrl
  match dscp cs7
 ! Network Control is matched on DSCP CS7
class-map match-all AutoQos-4.0-Internetwork-Ctrl
  match dscp cs6
 ! Internetwork Control is matched on DSCP CS6
class-map match-any AutoQos-4.0-Signaling
  match dscp cs3
  match cos 3
 ! Signaling is matched on DSCP CS3 and CoS 3
class-map match-all AutoQos-4.0-Network-Mgmt
  match dscp cs2
 ! Network management is matched on DSCP CS2
class-map match-any AutoQos-4.0-Multimedia-Conf
  match dscp af41
  match dscp af42
  match dscp af43
 ! Multimedia Conferencing is matched on AF4
class-map match-any AutoQos-4.0-Multimedia-Stream
  match dscp af31
  match dscp af32
  match dscp af33
 ! Multimedia Streaming is matched on AF3
class-map match-any AutoQos-4.0-Transaction-Data
  match dscp af21
  match dscp af22
  match dscp af23
 ! Transactional Data is matched on AF2
class-map match-any AutoQos-4.0-Bulk-Data
```

```
  match dscp af11
  match dscp af12
  match dscp af13
 ! Bulk Data is matched on AF1
class-map match-all AutoQos-4.0-Scavenger
  match dscp cs1
 ! Scavenger is matched on DSCP CS1

! This section defines the AutoQos-4.0-Input-Policy map
policy-map AutoQos-4.0-Input-Policy
  class AutoQos-4.0-VoIP
   set qos-group 32
 ! VoIP media is associated with QoS-Group 32 (for queuing)
  class AutoQos-4.0-Broadcast-Vid
   set qos-group 32
 ! Broadcast Video is associated with QoS-Group 32 (for queuing)
  class AutoQos-4.0-Realtime-Interact
   set qos-group 32
 ! Realtime Interactive is associated with QoS-Group 32 (for queuing)
  class AutoQos-4.0-Network-Ctrl
   set qos-group 16
 ! Network Control is associated with QoS-Group 16 (for queuing)
  class AutoQos-4.0-Internetwork-Ctrl
   set qos-group 16
 ! Internetwork Control is associated with QoS-Group 16 (for queuing)
  class AutoQos-4.0-Signaling
   set qos-group 16
 ! Signaling is associated with QoS-Group 16 (for queuing)
  class AutoQos-4.0-Network-Mgmt
   set qos-group 16
 ! Network Management is associated with QoS-Group 16 (for queuing)
  class AutoQos-4.0-Multimedia-Conf
   set qos-group 34
 ! Multimedia Conf is associated with QoS-Group 34 (for queuing)
  class AutoQos-4.0-Multimedia-Stream
   set qos-group 26
 ! Multimedia Streaming is associated with QoS-Group 26 (for queuing)
  class AutoQos-4.0-Transaction-Data
   set qos-group 18
 ! Transactional Data is associated with QoS-Group 18 (for queuing)
  class AutoQos-4.0-Bulk-Data
   set qos-group 10
 ! Bulk Data is associated with QoS-Group 10 (for queuing)
  class AutoQos-4.0-Scavenger
   set qos-group 8
 ! Scavenger is associated with QoS-Group 8 (for queuing)
```

AutoQos-4.0-Classify-Input-Policy

In contrast to the previous model, the AutoQos-4.0-Classify-Input-Policy, shown in Example A-20, is intended for ports that are connected to *untrusted* endpoints, and therefore bases classification on extended IP ACLs for marking and queuing (via the AutoQos-4.0-Output-Policy).

Example A-20 *C4500 AutoQoS: AutoQos-4.0-Classify-Input-Policy*

```
C4500# show run
Building configuration...
<snip>
 ! This section defines AutoQos-4.0-Classify-Input-Policy class maps
 ! Each class map is associated with a respective extended ACL
 ! <Extended ACLs are not shown for the sake of brevity>
class-map match-all AutoQos-4.0-Multimedia-Conf-Classify
  match access-group name AutoQos-4.0-ACL-Multimedia-Conf
class-map match-all AutoQos-4.0-Signaling-Classify
  match access-group name AutoQos-4.0-ACL-Signaling
class-map match-all AutoQos-4.0-Transaction-Classify
  match access-group name AutoQos-4.0-ACL-Transactional-Data
class-map match-all AutoQos-4.0-Bulk-Data-Classify
  match access-group name AutoQos-4.0-ACL-Bulk-Data
class-map match-all AutoQos-4.0-Scavenger-Classify
  match access-group name AutoQos-4.0-ACL-Scavenger
class-map match-all AutoQos-4.0-Default-Classify
  match access-group name AutoQos-4.0-ACL-Default

 ! This section defines the AutoQos-4.0-Classify-Input-Policy map
policy-map AutoQos-4.0-Classify-Input-Policy
  class AutoQos-4.0-Multimedia-Conf-Classify
   set dscp af41
   set cos 4
   set qos-group 34
 ! Multimedia Conf is marked DSCP AF41 and CoS 4
 ! and is associated with QoS-Group 34 (for queuing)
  class AutoQos-4.0-Signaling-Classify
   set dscp cs3
   set cos 3
   set qos-group 16
 ! Signaling is marked DSCP CS3 and CoS 3
 ! and is associated with QoS-Group 16 (for queuing)
  class AutoQos-4.0-Transaction-Classify
   set dscp af21
   set cos 2
   set qos-group 18
 ! Transactional Data is marked DSCP AF21 and CoS 2
 ! and is associated with QoS-Group 18 (for queuing)
  class AutoQos-4.0-Bulk-Data-Classify
   set dscp af11
   set cos 1
   set qos-group 10
 ! Bulk Data is marked DSCP AF11 and CoS 1
 ! and is associated with QoS-Group 10 (for queuing)
  class AutoQos-4.0-Scavenger-Classify
   set dscp cs1
   set cos 1
   set qos-group 8
 ! Scavenger is marked DSCP CS1 and CoS 1
 ! and is associated with QoS-Group 8 (for queuing)
  class AutoQos-4.0-Default-Classify
   set dscp default
   set cos 0
 ! Best Effort is marked DSCP DF and CoS 0
```

AutoQos-4.0-Cisco-Phone-Input-Policy

In the AutoQos-4.0-Cisco-Phone-Input-Policy, VoIP and signaling traffic (matched by CoS 5 and CoS 3, respectively) are marked to DSCP EF and CS3 (respectively) and associated with QoS groups 32 and 16 (respectively). In addition, VoIP is policed to 128 Kbps and signaling is policed to 32 Kbps; both traffic classes will be re-marked to CS1 (scavenger) if exceeding their respective policing rates. Finally, a data-plane policing policy is applied to the default class to re-mark endpoint-generated flows as scavenger if above 10 Mbps. The AutoQos-4.0-Cisco-Phone-Input-Policy is shown in Example A-21.

Example A-21 *C4500 AutoQoS: AutoQos-4.0-Cisco-Phone-Input-Policy*

```
C4500# show run
Building configuration...
<snip>
! <class maps are omitted for brevity and to minimize redundancy>
! This section defines the AutoQos-4.0-Cisco-Phone-Input-Policy map
policy-map AutoQos-4.0-Cisco-Phone-Input-Policy
  class AutoQos-4.0-VoIP-Data-Cos
   set dscp ef
   set qos-group 32
   police cir 128000 bc 8000
     conform-action transmit
     exceed-action set-dscp-transmit cs1
     exceed-action set-cos-transmit 1
```

```
 ! VoIP is marked DSCP EF & associated with QoS-group 32 (for queuing)
 ! and is policed to 128 Kbps and re-marked as scavenger if exceeding
 class AutoQos-4.0-VoIP-Signal-Cos
   set dscp cs3
   set qos-group 16
   police cir 32000 bc 8000
     conform-action transmit
     exceed-action set-dscp-transmit cs1
     exceed-action set-cos-transmit 1
 ! Signaling is marked DSCP CS3 & associated with QoS group 16
 ! and is policed to 32kbps and re-marked as scavenger if exceeding
  class AutoQos-4.0-Default-Classify
   set dscp default
   set cos 0
   police cir 10000000 bc 8000
     conform-action transmit
     exceed-action set-dscp-transmit cs1
     exceed-action set-cos-transmit 1
 ! Best Effort is marked DSCP DF & CoS 0
 ! and is policed to 10 Mbps and re-marked as scavenger if exceeding
```

AutoQos-4.0-Cisco-Softphone-Input-Policy

The AutoQos-4.0-Cisco-Softphone-Input-Policy expands on the previous model to include data plane policing on all traffic classes, and is shown in Example A-22.

Example A-22 *C4500 AutoQoS: AutoQos-4.0-Cisco-Softphone-Input-Policy*

```
C4500# show run
Building configuration...
<snip>
! <class maps are omitted for brevity and to minimize redundancy>
! This section defines the AutoQos-4.0-Cisco-Softphone-Input-Policy map
policy-map AutoQos-4.0-Cisco-Softphone-Input-Policy
  class AutoQos-4.0-VoIP-Data
   set dscp ef
   set cos 5
   set qos-group 32
   police cir 128000 bc 8000
     conform-action transmit
     exceed-action set-dscp-transmit cs1
     exceed-action set-cos-transmit 1
 ! VoIP is marked DSCP EF & CoS 5
 ! and is associated with QoS-group 32 (for queuing)
! and is policed to 128 Kbps and re-marked as scavenger if exceeding
 class AutoQos-4.0-VoIP-Signal
  set dscp cs3
  set cos 3
  set qos-group 16
  police cir 32000 bc 8000
    conform-action transmit
    exceed-action set-dscp-transmit cs1
    exceed-action set-cos-transmit 1
! Signaling is marked DSCP CS3 & CoS 3
! and is associated with QoS group 16 (for queuing)
! and is policed to 32 Kbps and re-marked as scavenger if exceeding
 class AutoQos-4.0-Multimedia-Conf-Classify
  set dscp af41
  set cos 4
  set qos-group 34
  police cir 5000000 bc 8000
    conform-action transmit
    exceed-action drop
! Multimedia conferencing is marked DSCP AF41 & CoS 4
! and is associated with QoS group 34 (for queuing)
! and is policed to 5 Mbps and re-marked as scavenger if exceeding
 class AutoQos-4.0-Signaling-Classify
  set dscp cs3
  set cos 3
  set qos-group 16
  police cir 32000 bc 8000
    conform-action transmit
    exceed-action drop
! Signaling is marked DSCP CS3 & CoS 3
! and is associated with QoS group 16 (for queuing)
! and is policed to 32 Kbps and re-marked as scavenger if exceeding
 class AutoQos-4.0-Transaction-Classify
  set dscp af21
  set cos 2
  set qos-group 18
  police cir 10000000 bc 8000
    conform-action transmit
    exceed-action set-dscp-transmit cs1
    exceed-action set-cos-transmit 1
! Transactional data is marked DSCP AF21 & CoS 2
! and is associated with QoS group 18 (for queuing)
! and is policed to 10 Mbps and re-marked as scavenger if exceeding
 class AutoQos-4.0-Bulk-Data-Classify
  set dscp af11
```

```
   set cos 1
   set qos-group 10
   police cir 10000000 bc 8000
     conform-action transmit
     exceed-action set-dscp-transmit cs1
     exceed-action set-cos-transmit 1
 ! Transactional data is marked DSCP AF11 & CoS 1
 ! and is associated with QoS group 10 (for queuing)
 ! and is policed to 10 Mbps and re-marked as Scavenger if exceeding
  class AutoQos-4.0-Scavenger-Classify
   set dscp cs1
   set cos 1
   set qos-group 8
   police cir 10000000 bc 8000
     conform-action transmit
     exceed-action drop
 ! Scavenger is marked DSCP CS1 & CoS 1
 ! and is associated with QoS group 8 (for queuing)
 ! and is policed to 10 Mbps and re-marked as Scavenger if exceeding
  class AutoQos-4.0-Default-Classify
   set dscp default
   set cos 0
 ! Best effort is marked DSCP DF & CoS 0
```

AutoQos-4.0-Output-Policy

Because the previously defined AutoQoS-4.0 input policies have mapped application classes to their respective QoS-groups, these QoS groups can be mapped directly to egress queues by the AutoQos-4.0-Output-Policy, as shown in Example A-23.

Example A-23 *C4500 AutoQoS: AutoQos-4.0-Output-Policy*

```
C4500# show run
Building configuration...
<snip>
! <class maps are omitted for brevity and to minimize redundancy>
! This section defines the AutoQos-4.0-Output-Policy map
policy-map AutoQos-4.0-Output-Policy
  class AutoQos-4.0-Scavenger-Queue
   bandwidth remaining percent 1
 ! Scavenger traffic is constrained to 1% BWR
  class AutoQos-4.0-Priority-Queue
   priority
   police cir percent 30 bc 33 ms
```

```
    conform-action transmit
    exceed-action drop
! VoIP traffic is given strict-priority service
! but is limited to 33%
 class AutoQos-4.0-Control-Mgmt-Queue
  bandwidth remaining percent 10
! Control and Management traffic is given 10% BWR
 class AutoQos-4.0-Multimedia-Conf-Queue
  bandwidth remaining percent 10
! Multimedia conferencing traffic is given 10% BWR
 class AutoQos-4.0-Multimedia-Stream-Queue
  bandwidth remaining percent 10
! Multimedia streaming traffic is given 10% BWR
 class AutoQos-4.0-Trans-Data-Queue
  bandwidth remaining percent 10
  dbl
! Transactional data traffic is given 10% BWR with DBL
 class AutoQos-4.0-Bulk-Data-Queue
  bandwidth remaining percent 4
  dbl
! Bulk data traffic is given 5% BWR with DBL
 class class-default
  bandwidth remaining percent 25
  dbl
! Best effort traffic is given 25% BWR with DBL
```

Additional Reading

Cisco Enterprise Medianet Campus QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html

Cisco Medianet Campus AutoQoS At-A-Glance: http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/autoqosmediacampus.pdf

Cisco Catalyst 3750 AutoQoS Configuration Guide: http://www.cisco.com/en/US/docs/switches/lan/catalyst3750x_3560x/software/release/15.0_2_se/configuration/guide/swqos.html#wp1231112

Cisco Catalyst 4500 AutoQoS Configuration Guide: http://www.cisco.com/en/US/docs/switches/lan/catalyst4500/15.1/XE_330SG/configuration/guide/qos_mrg.html#wp1563359

Appendix B

Control Plane Policing

Control plane policing (abbreviated as CPP for Cisco IOS routers and as CoPP for Cisco IOS switches) is an application of quality of service (QoS) technologies in a security context that is available on switches and routers running Cisco IOS that allows the configuration of QoS policies that rate limit the traffic handled by the main CPU of the network device. This protects the control plane of the switch from direct denial-of-service (DoS) attacks, reconnaissance activity, and other unexpected flooding of the control plane.

CPP/CoPP protects IOS-based routers and switches by allowing the definition and enforcement of QoS policies that regulate the traffic processed by the main switch CPU (route or switch processor). With CPP/CoPP, these QoS policies are configured to permit, block, or rate limit the packets handled by the main CPU.

A router or switch can be logically divided into four functional components or planes:

- Data plane
- Management plane
- Control plane
- Services plane

The vast majority of traffic travels through the router via the data plane. However, the route/switch processor (which will hereafter be abbreviated as RP, for route processor) must handle certain packets, such as routing updates, keepalives, and network management. This is often referred to as control and management plane traffic.

Because the RP is critical to network operations, any service disruption to it or the control and management planes can result in business-impacting network outages. A DoS attack targeting the RP, which can be perpetrated either inadvertently or maliciously, usually involves high rates of traffic that needs to be punted up to the CPU (from local routing cache or distributed routing engines) that results in excessive CPU utilization on the RP itself. This may be the case with packets destined to nonexistent networks or other routing failures. This type of attack, which can be devastating to network stability and availability, may display the following symptoms:

- High route processor CPU utilization (near 100 percent).
- Loss of line protocol keepalives and routing protocol updates, leading to route flaps and major network transitions.
- Interactive sessions via the command-line interface (CLI) are slow or completely unresponsive due to high CPU utilization.
- RP resource exhaustion, meaning that resources such as memory and buffers are unavailable for legitimate IP data packets.
- Packet queue backup, which leads to indiscriminate drops (or drops due to lack of buffer resources) of other incoming packets.

CPP/CoPP addresses the need to protect the control and management planes, ensuring routing stability, availability, and packet delivery. It uses a dedicated control plane configuration via the Modular QoS command-line interface (MQC) to provide filtering and rate limiting capabilities for control plane packets.

Packets handled by the main CPU, referred to as control plane traffic, typically include the following:

- Routing protocols.
- Packets destined to the local IP address of the router.
- Packets from network management protocols, such as Simple Network Management Protocol (SNMP).
- Interactive access protocols, such as Secure Shell (SSH) and Telnet.
- Other protocols, such as Internet Control Message Protocol (ICMP), or IP options, might also require handling by the switch CPU.
- Layer 2 packets such as bridge protocol data unit (BPDU), Cisco Discovery Protocol (CDP), DOT1X, and so on.
- Layer 3 protocols such as authentication, authorization, and accounting (AAA), syslog, Network Time Protocol (NTP), Internet Security Association and Key Management Protocol (ISAKMP), Resource Reservation Protocol (RSVP), and so on.

CPP/CoPP leverages the MQC for its QoS policy configuration. MQC allows the classification of traffic into classes and lets you define and apply distinct QoS policies to separately rate limit the traffic in each class. MQC lets you divide the traffic destined to the CPU into multiple classes based on different criteria. For example, four traffic classes could be defined based on relative importance:

- Critical
- Normal

- Undesirable
- Default

After you define the traffic classes, you can define and enforce a QoS policy for each class according to importance. The QoS policies in each class can be configured to permit all packets, drop all packets, or drop only those packets exceeding a specific rate limit.

Note The number of control plane classes is not limited to four, but should be chosen based on local network requirements, security policies, and a thorough analysis of the baseline traffic.

Note It is also important to keep in mind that CoPP/CPP does not protect the switch/router against itself. For example, in some situations SNMP traps or NetFlow exports generated by the device may be excessive and could have detrimental effects on the CPU, the same way a DoS attack might. Therefore, it is beneficial for an administrator to keep this caveat in mind when deploying such management policies.

Defining Control Plane Policing Traffic Classes

Developing a CPP policy starts with the classification of the control plane traffic. To that end, the control plane traffic needs to be first identified and separated into different class maps.

This section presents a classification template that can be used as a model when implementing CPP on IOS routers in addition to when deploying CoPP on Cisco Catalyst switches. This template presents a realistic classification, where traffic is grouped based on its relative importance and protocol type. The template uses eight different classes, which provide a high level of granularity and make it suitable for real-world environments.

Note Even though you can use this template as a reference, the actual number and type of classes needed for a given network can differ and should be selected based on local requirements, security policies, and a thorough analysis of baseline traffic.

This CPP/CoPP template defines these eight traffic classes:

- **Border Gateway Protocol (BGP):** This class defines traffic that is crucial to maintaining neighbor relationships for BGP routing protocol, such as BGP keepalives and routing updates. Maintaining BGP routing protocol is crucial to maintaining connectivity within a network or to an Internet service provider (ISP). Sites that are not running BGP would not use this class.

- **Interior Gateway Protocol (IGP):** This class defines traffic that is crucial to maintaining IGP routing protocols, such as Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), and Routing Information Protocol (RIP). Maintaining IGP routing protocols is crucial to maintaining connectivity within a network.
- **Interactive Management:** This class defines interactive traffic that is required for day-to-day network operations. This class would include light volume traffic used for remote network access and management. For example, Telnet, SSH, NTP, SNMP, and Terminal Access Controller Access Control System (TACACS).
- **File Management:** This class defines high-volume traffic used for software image and configuration maintenance. This class would include traffic generated for remote file transfer; for example, Trivial File Transfer Protocol (TFTP) and File Transfer Protocol (FTP).
- **Monitoring/Reporting:** This class defines traffic used for monitoring a router. This kind of traffic should be permitted but should never be allowed to pose a risk to the router. With CPP/CoPP, this traffic can be permitted but limited to a low rate. Examples include packets generated by ICMP echo requests (ping and traceroute) in addition to traffic generated by Cisco IOS IP Service Level Agreements (IP SLAs) to generate ICMP with different differentiated services code point (DSCP) settings to report on response times within different QoS data classes.
- **Critical Applications:** This class defines application traffic that is crucial to a specific network. The protocols that might be included in this class include generic routing encapsulation (GRE), Hot Standby Router Protocol (HSRP), Virtual Router Redundancy Protocol (VRRP), Gateway Load Balancing Protocol (GLBP), Session Initiation Protocol (SIP), Data Link Switching (DLSw), Dynamic Host Configuration Protocol (DHCP), Multicast Source Discovery Protocol (MSDP), Internet Group Management Protocol (IGMP), Protocol Independent Multicast (PIM), multicast traffic, and IPsec.
- **Undesirable:** This explicitly identifies unwanted or malicious traffic that should be dropped and denied access to the RP. For example, this class could contain packets from a well-known worm. This class is particularly useful when specific traffic destined to the router should always be denied rather than be placed into a default category. Explicitly denying traffic allows you to collect rough statistics on this traffic using show commands and thereby offers some insight into the rate of denied traffic. Access control list entries (ACEs) used for classifying undesirable traffic may be added and modified as new undesirable applications appear on the network, and therefore you can use these ACEs as a reaction tool.
- **Default:** This class defines all remaining traffic destined to the RP that does not match any other class. MQC provides the default class so that you can specify how

to treat traffic that is not explicitly associated with any other user-defined classes. It is desirable to give such traffic access to the RP, but at a highly reduced rate. With a default classification in place, statistics can be monitored to determine the rate of otherwise unidentified traffic destined to the control plane. After this traffic is identified, further analysis can be performed to classify it. If needed, the other CPP/CoPP policy entries can be updated to account for this traffic.

Deploying Control Plane Policing Policies

Because CPP/CoPP filters traffic, it is critical to gain an adequate level of understanding about the legitimate traffic destined to the RP prior to deployment. CPP/CoPP policies built without proper understanding of the protocols, devices, or required traffic rates involved can block critical traffic, which has the potential of creating a DoS condition. Determining the exact traffic profile needed to build the CPP/CoPP policies might be difficult in some networks.

The following steps follow a conservative methodology that facilitates the process of designing and deploying CPP/CoPP. This methodology uses iterative access control list (ACL) configurations to help identify and to incrementally filter traffic.

To deploy CPP/CoPP, you should perform the six steps that follow.

Step 1: Determine the Classification Scheme for Your Network

Identify the known protocols that access the RP and divide them into categories using the most useful criteria for your specific network. As an example of classification, the eight categories template presented earlier in this section (BGP, IGP, Interactive Management, File Management, Reporting, Critical Applications, Undesirable, and Default) use a combination of relative importance and traffic type. Select a scheme suited to your specific network, which might require a larger or smaller number of classes.

Step 2: Define Classification Access Lists

Configure each ACL to permit all known protocols in its class that require access to the RP. At this point, each ACL entry should have both source and destination addresses set to **any**. In addition, the ACL for the default class should be configured with a single entry, **permit ip any any**. This matches traffic not explicitly permitted by entries in the other ACLs. After the ACLs have been configured, create a class map for each class defined in Step 1, including one for the default class. Then assign each ACL to its corresponding class map.

Note In this step, you should create a separate class map for the default class, instead of using the class default available in some platforms. Creating a separate class map and assigning a **permit ip any any** ACL allows you to identify traffic not yet classified as part of another class.

Each class map should then be associated with a policy map that permits all traffic, regardless of classification. The policy for each class should be set as conform-action transmit exceed-action transmit.

Step 3: Review the Identified Traffic and Adjust the Classification.

Ideally, the classification performed in Step 1 identified all required traffic destined to the router. However, realistically, not all required traffic is identified before deployment, and the **permit ip any any** entry in the default class ACL logs a number of packet matches. Some form of analysis is required to determine the exact nature of the unclassified packets. For example, you can use the **show access-lists** command to see the entries in the ACLs that are in use and to identify any additional traffic sent to the RP. However, to analyze the unclassified traffic, you can use one of these techniques:

- General ACL classification based on traffic characterization and tracing by Cisco routers

Note For more information on this technique, see http://www.cisco.com/en/US/tech/tk59/technologies_tech_note09186a0080149ad6.shtml.

- Packet analyzers

When traffic has been properly identified, adjust the class configuration accordingly. Remove the ACL entries for those protocols that are not used. Add a **permit ip any any** entry for each protocol just identified.

Step 4: Restrict a Macro Range of Source Addresses

Refine the classification ACLs by only allowing the full range of the allocated CIDR block to be permitted as the source address. For example, if the network has been allocated 172.68.0.0/16, permit source addresses from 172.68.0.0/16 where applicable.

This step provides data points for devices or users from outside the CIDR block that might be accessing the equipment. An external BGP (eBGP) peer requires an exception because the permitted source addresses for the session lies outside the CIDR block. This phase might be left on for a few days to collect data for the next phase of narrowing the ACL entries.

Step 5: Narrow the ACL Permit Statements to Authorized Source Addresses

Increasingly limit the source address in the classification ACLs to permit only sources that communicate with the RP. For example, only known network management stations should be permitted to access the SNMP ports on a router.

Step 6: Refine CPP/CoPP Policies by Implementing Rate Limiting

Use the **show policy-map control-plane** command to collect data about the actual policies in place. Analyze the packet count and rate information and develop a rate-limiting policy accordingly. At this point, you might decide to remove the class map and ACL used for the classification of default traffic. If so, you should also replace the previously defined policy for the default class by the class default policy.

Note Table AB-1 shows a set of tested and validated CPP/CoPP rates. Note that the values presented here are solely for illustration purposes, as every environment has different baselines.

Note At the time of this writing, the Catalyst 4500 supports policing rates only in bits per second (bps), but the Catalyst 6500 and Cisco IOS support both bps and packets per second (pps) rate limits. When configuring CPP, rate limiting using a pps rate is generally preferred because it is the per-packet processing that more significantly impacts CPU utilization than the bps rate.

Table B-1 summarizes control plane policing rate examples along with recommended conforming and exceeding actions.

Table B-1 *Control Plane Policing Rate Limits and Actions Examples*

Traffic Class	Rate (pps)	Rate (bps)	Conform Action	Exceed Action
Border Gateway Protocol	500	4,000,000	Transmit	Drop
Interior Gateway Protocol	50	300,000	Transmit	Drop
Interactive Management	100	500,000	Transmit	Drop
File Management	500	6,000,000	Transmit	Drop

Traffic Class	Rate (pps)	Rate (bps)	Conform Action	Exceed Action
Monitoring	125	900,000	Transmit	Drop
Critical Applications	125	900,000	Transmit	Drop
Undesirable	10[1]	32 Kbps[1]	Drop	Drop
Default	100	500,000	Transmit	Drop

[1] The policing rate for the Undesirable CPP class is effectively 0—regardless of whether pps or bps rates are used and also regardless of what values these rates are set to—since both the conforming and exceeding actions for this class are set to drop. However, the policer still performs a metering operation of the rates of these flows which is often useful for management purposes.

Cisco Catalyst 3850 Control Plane Policing

The Catalyst 3850 switch supports control plane policing, but at the time of this writing, this feature is not configurable.

Cisco Catalyst 4500 Control Plane Policing

On Cisco IOS Catalyst switches, CoPP comes into play right after the switching or the routing decision and before traffic is forwarded to the control plane. When CoPP is enabled, the sequence of events (at a high level) is as follows:

1. A packet enters the switch configured with CoPP on the ingress port.
2. The port performs any applicable input port and QoS services.
3. The packet gets forwarded to the switch CPU.
4. The switch CPU makes a routing or a switching decision, determining whether the packet is destined for the control plane.
5. Packets destined for the control plane are processed by CoPP and are dropped or delivered to the control plane according to each traffic class policy. Packets that have other destinations are forwarded normally.

The Catalyst 4500 and Catalyst 6500 series switches implement CoPP similarly. However, CoPP has been enhanced on both platforms to leverage the benefits of their hardware architectures, and as a result each platform provides unique features. Therefore, the CoPP implementations on Catalyst 4500 and Catalyst 6500 series switches are discussed in platform-specific detail in their respective sections within this appendix.

The Catalyst 4500 series switches support CoPP in hardware in a centralized, nondistributed fashion. CoPP policies are centrally configured under the control plane configuration mode and then enforced in hardware by the classification ternary content-address-able memory (TCAM) and QoS policers of the Supervisor Engine. Figure B-1 shows this CoPP model.

The Catalyst 4500 implementation of CoPP uses MQC to define traffic classification criteria and to specify the configurable policy actions for the classified traffic. MQC uses class maps to define packets for a particular traffic class. After you have classified the traffic, you can create policy maps to enforce policy actions for the identified traffic. The control plane global configuration command allows the CoPP service policy to be directly attached to the control plane.

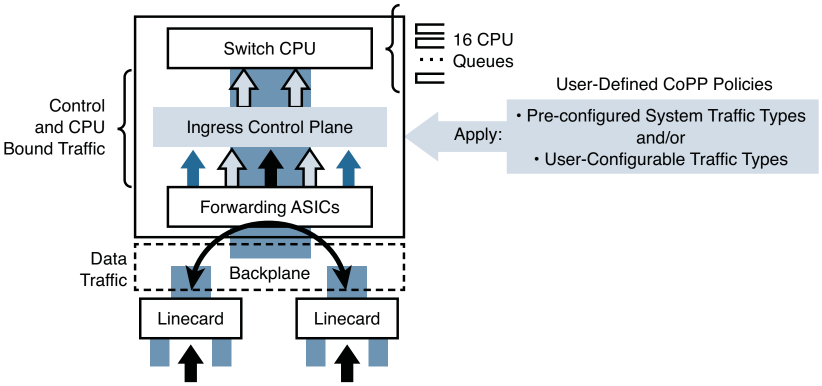

Switch CPU
16 CPU Queues
User-Defined CoPP Policies
Control and CPU Bound Traffic
Ingress Control Plane
Apply:
• Pre-configured System Traffic Types
and/or
• User-Configurable Traffic Types
Forwarding ASICs
Data Traffic
Backplane
Linecard
Linecard

Figure B-1 *Catalyst 4500 Control Plane Policing Implementation*

Catalyst 4500 CoPP supports the definition of non-IP traffic classes in addition to IP traffic classes. With this, instead of using the default class for handling all non-IP traffic, you can define separate policies for non-IP traffic. This results in better and more granular control over non-IP protocols, such as Address Resolution Protocol (ARP), Internetwork Packet Exchange (IPX), bridge protocol data units (BPDUs), Cisco Discovery Protocol (CDP), and Secure Socket Tunneling Protocol (SSTP).

One particular characteristic of (Multi-Layer Switch [MLS]-based) Catalyst 4500 CoPP is that the CoPP policy must be named system-cpp-policy. In fact, on these systems, the system-cpp-policy is the only policy map that can be attached to the control plane.

> **Note** This restriction has been removed on MQC-based Catalyst 4500 series switches, beginning with the Supervisor 6-E. However, to maintain backward compatibility, the system-cpp-policy name has been used in this configuration example.

To facilitate the configuration of system-cpp-policy, Catalyst 4500's CoPP provides a global macro function (called system-cpp) that automatically generates and applies CoPP policies to the control plane. The resulting configuration uses a collection of system-defined class maps for common Layer 3 and Layer 2 control plane traffic. The names of all system-defined CoPP class maps and their matching ACLs contain the prefix system-cpp. By default, no action is specified on any of the system predefined traffic classes. Table B-2 lists the predefined system ACLs.

Table B-2 *Catalyst 4500 System Predefined CoPP ACLs*

Predefined Named ACL	Description
system-cpp-dot1x	MAC DA = 0180.C200.0003
system-cpp-lldp	MAC DA=0180.c200.000E
system-cpp-mcast-cfm	MAC DA=0100.0ccc.ccc0 - 0100.0ccc.ccc7
system-cpp-ucast-cfm	MAC DA=0100.0ccc.ccc0
system-cpp-bpdu-range	MAC DA = 0180.C200.0000 - 0180.C200.000F
system-cpp-cdp	MAC DA = 0100.0CCC.CCCC (UDLD/DTP/VTP/Pagp)
system-cpp-sstp	MAC DA = 0100.0CCC.CCCD
system-cpp-cgmp	MAC DA = 01-00-0C-DD-DD-DD
system-cpp-ospf	IP Protocol = OSPF, IP DA matches 224.0.0.0/24
system-cpp-igmp	IP Protocol = IGMP, IP DA matches 224.0.0.0/3
system-cpp-pim	IP Protocol = PIM, IP DA matches 224.0.0.0/24
system-cpp-all-systems-on-subnet	IP DA = 224.0.0.1
system-cpp-all-routers-on-subnet	IP DA = 224.0.0.2
system-cpp-ripv2	IP DA = 224.0.0.9
system-cpp-ip-mcast-linklocal	IP DA = 224.0.0.0/24
system-cpp-dhcp-cs	IP Protocol = UDP, L4SrcPort = 68, L4DstPort = 67
system-cpp-dhcp-sc	IP Protocol = UDP, L4SrcPort = 67, L4DstPort = 68
system-cpp-dhcp-ss	IP Protocol = UDP, L4SrcPort = 67, L4DstPort = 67

In addition to the predefined classes, you can configure your own class maps matching other control plane traffic. To take effect, these user-defined class maps need to be added to the system-cpp-policy policy map.

CoPP can be deployed on the Catalyst 4500 in one of two main ways:

- You can use the global macro **macro global apply system-cpp** to preconfigure CoPP access lists, class maps, and a system-cpp-policy policy map (with no class actions configured); an administrator can then modify this template and tune it to suit specific environments.

- The CoPP policy can be generated manually (as shown in the Example B-1).

In Example B-1, CoPP has been deployed manually (to keep the policy as consistent as possible between the other CoPP/CPP configuration examples), inline with the previously defined recommendations.

Note It is recommended to include a CPP or CoPP prefix in the ACL and class map names to prevent any potential classification errors for similarly named ACLs and class maps used in data plane policies.

Example B-1 *Catalyst 4500 Control Plane Policing Configuration Example*

```
! This section defines the access lists for the CoPP traffic classes
C4500-E(config)# ip access-list extended COPP-BGP
C4500-E(config-ext-nacl)# remark BGP
C4500-E(config-ext-nacl)# permit tcp host 192.168.1.1 host 10.1.1.1 eq bgp
C4500-E(config-ext-nacl)# permit tcp host 192.168.1.1 eq bgp host 10.1.1.1

C4500-E(config)# ip access-list extended COPP-IGP
C4500-E(config-ext-nacl)# remark IGP (OSPF)
C4500-E(config-ext-nacl)# permit ospf any host 224.0.0.5
C4500-E(config-ext-nacl)# permit ospf any host 224.0.0.6
C4500-E(config-ext-nacl)# permit ospf any any

C4500-E(config)# ip access-list extended COPP-INTERACTIVE-MANAGEMENT
C4500-E(config-ext-nacl)# remark TACACS (return traffic)
C4500-E(config-ext-nacl)# permit tcp host 10.2.1.1 host 10.1.1.1 established
C4500-E(config-ext-nacl)# remark SSH
C4500-E(config-ext-nacl)# permit tcp 10.2.1.0 0.0.0.255 host 10.1.1.1 eq 22
C4500-E(config-ext-nacl)# remark SNMP
C4500-E(config-ext-nacl)# permit udp host 10.2.2.2 host 10.1.1.1 eq snmp
C4500-E(config-ext-nacl)# remark NTP
C4500-E(config-ext-nacl)# permit udp host 10.2.2.3 host 10.1.1.1 eq ntp
C4500-E(config)# ip access-list extended COPP-FILE-MANAGEMENT
C4500-E(config-ext-nacl)# remark (initiated) FTP (active and passive)
C4500-E(config-ext-nacl)# permit tcp 10.2.1.0 0.0.0.255 eq 21 host 10.1.1.1 gt 1023
established
C4500-E(config-ext-nacl)# permit tcp 10.2.1.0 0.0.0.255 eq 20 host 10.1.1.1 gt 1023
C4500-E(config-ext-nacl)# permit tcp 10.2.1.0 0.0.0.255 gt 1023 host 10.1.1.1 gt
1023 established
C4500-E(config-ext-nacl)# remark (initiated) TFTP
C4500-E(config-ext-nacl)# permit udp 10.2.1.0 0.0.0.255 gt 1023 host 10.1.1.1 gt
1023
```

```
C4500-E(config)# ip access-list extended COPP-MONITORING
C4500-E(config-ext-nacl)# remark PING-ECHO
C4500-E(config-ext-nacl)# permit icmp any any echo
C4500-E(config-ext-nacl)# remark PING-ECHO-REPLY
C4500-E(config-ext-nacl)# permit icmp any any echo-reply
C4500-E(config-ext-nacl)# remark TRACEROUTE
C4500-E(config-ext-nacl)# permit icmp any any ttl-exceeded
C4500-E(config-ext-nacl)# permit icmp any any port-unreachable

C4500-E(config)# ip access-list extended COPP-CRITICAL-APPLICATIONS
C4500-E(config-ext-nacl)# remark HSRP
C4500-E(config-ext-nacl)# permit ip any host 224.0.0.2
C4500-E(config-ext-nacl)# remark DHCP
C4500-E(config-ext-nacl)# permit udp host 0.0.0.0 host 255.255.255.255 eq bootps
C4500-E(config-ext-nacl)# permit udp host 10.2.2.8 eq bootps any eq bootps

C4500-E(config)# ip access-list extended COPP-UNDESIRABLE
C4500-E(config-ext-nacl)# remark UNDESIRABLE TRAFFIC
C4500-E(config-ext-nacl)# permit udp any any eq 1434



 ! This section defines the CoPP policy class maps
C4500-E(config)# class-map match-all COPP-BGP
C4500-E(config-cmap)# match access-group name COPP-BGP
 ! Associates COPP-BGP ACL with class map
C4500-E(config)# class-map match-all COPP-IGP
C4500-E(config-cmap)# match access-group name COPP-IGP
 ! Associates COPP-IGP ACL with class map
C4500-E(config)# class-map match-all COPP-INTERACTIVE-MANAGEMENT
C4500-E(config-cmap)# match access-group name COPP-INTERACTIVE-MANAGEMENT
  ! Associates COPP-INTERACTIVE-MANAGEMENT ACL with class map
C4500-E(config)# class-map match-all COPP-FILE-MANAGEMENT
C4500-E(config-cmap)# match access-group name COPP-FILE-MANAGEMENT
  ! Associates COPP-FILE-MANAGEMENT with class map
C4500-E(config)# class-map match-all COPP-MONITORING
C4500-E(config-cmap)# match access-group name COPP-MONITORING
  ! Associates COPP-MONITORING ACL with class map
C4500-E(config)# class-map match-all COPP-CRITICAL-APPLICATIONS
C4500-E(config-cmap)# match access-group name COPP-CRITICAL-APPLICATIONS
  ! Associates COPP-CRITICAL-APPLICATIONS ACL with class map
C4500-E(config)# class-map match-all COPP-UNDESIRABLE
C4500-E(config-cmap)# match access-group name COPP-UNDESIRABLE
  ! Associates COPP-UNDESIRABLE ACL with class map
 ! This section defines the CoPP policy
C4500-E(config-cmap)# policy-map system-cpp-policy
C4500-E(config-pmap)# class COPP-BGP
C4500-E(config-pmap-c)# police cir 4000000 bc 400000 be 400000
C4500-E(config-pmap-c-police)# conform-action transmit
C4500-E(config-pmap-c-police)# exceed-action drop
 ! Polices BGP to 4 Mbps
C4500-E(config-pmap)# class COPP-IGP
C4500-E(config-pmap-c)# police cir 300000 bc 3000 be 3000
C4500-E(config-pmap-c-police)# conform-action transmit
C4500-E(config-pmap-c-police)# exceed-action drop
 ! Polices IGP to 300 Kbps
C4500-E(config-pmap)# class COPP-INTERACTIVE-MANAGEMENT
C4500-E(config-pmap-c)# police cir 500000 bc 5000 be 5000
C4500-E(config-pmap-c-police)# conform-action transmit
C4500-E(config-pmap-c-police)# exceed-action drop
 ! Polices Interactive Management to 500 Kbps
C4500-E(config-pmap)# class COPP-FILE-MANAGEMENT
C4500-E(config-pmap-c)# police cir 6000000 bc 60000 be 60000
C4500-E(config-pmap-c-police)# conform-action transmit
C4500-E(config-pmap-c-police)# exceed-action drop
 ! Polices File Management to 6 Mbps
C4500-E(config-pmap)# class COPP-MONITORING
C4500-E(config-pmap-c)# police cir 900000 bc 9000 be 9000
C4500-E(config-pmap-c-police)# conform-action transmit
C4500-E(config-pmap-c-police)# exceed-action drop
 ! Polices Monitoring to 900 Kbps
C4500-E(config-pmap)# class COPP-CRITICAL-APPLICATIONS
C4500-E(config-pmap-c)# police cir 900000 bc 9000 be 9000
C4500-E(config-pmap-c-police)# conform-action transmit
C4500-E(config-pmap-c-police)# exceed-action drop
 ! Polices Critical Applications to 900 Kbps
C4500-E(config-pmap)# class COPP-UNDESIRABLE
C4500-E(config-pmap-c)# police cir 32000 bc 3000 be 3000
C4500-E(config-pmap-c-police)# conform-action drop
C4500-E(config-pmap-c-police)# exceed-action drop
 ! Polices all Undesirable traffic (conform action is drop)
C4500-E(config-pmap)# class class-default
C4500-E(config-pmap-c)# police cir 500000 bc 5000 be 5000
C4500-E(config-pmap-c-police)# conform-action transmit
C4500-E(config-pmap-c-police)# exceed-action drop
 ! Polices all other Control Plane traffic to 500  Kbps
 ! This section attaches the CoPP policy to the control plane
C4500-E(config)#control-plane
C4500-E(config-cp)# service-policy input system-cpp-policy
! Attaches CoPP policy to control plane
```

You can verify the configuration in Example B-1 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map control-plane**

Cisco Catalyst 6500 Control Plane Policing

As previously stated, the Catalyst 4500 and Catalyst 6500 series switches implement CoPP similarly. However, CoPP has been enhanced on both platforms to leverage the benefits of their hardware architectures, and as a result each platform provides unique features.

In the Catalyst 6500 series switches, CoPP takes advantage of the processing power present on linecards by implementing a distributed CoPP model. In this platform, the class QoS policies are centrally configured under the control plane configuration mode. When configured, these policies are first applied at the route processor (Multilayer Switch Feature Card [MSFC]) level, and then they get automatically pushed to the Policy Feature Card (PFC) and each Distributed Forwarding Card (DFC). Figure B-2 illustrates this CoPP model.

CoPP is enabled by default on the Catalyst 6500. These default settings provide adequate protection to the control plane in most deployment scenarios. To create manual CoPP policies, the default CoPP policy needs to be disabled. To disable the default CoPP configuration, enter the **no service-policy input policy-default-autocopp** control plane configuration mode command.

Example B-2 shows the corresponding CoPP configuration for the Catalyst 6500 series switches, based on the recommendations previously defined.

Note As previously mentioned, Cisco 6500 CoPP allows the ability to configure rate limits based on either bits per second or packets per second. When configuring CPP, rate limiting using a pps rate is preferred because it is the per-packet processing that more significantly impacts CPU utilization than the bps rate. The pps rates in the following example are based on Table B-1.

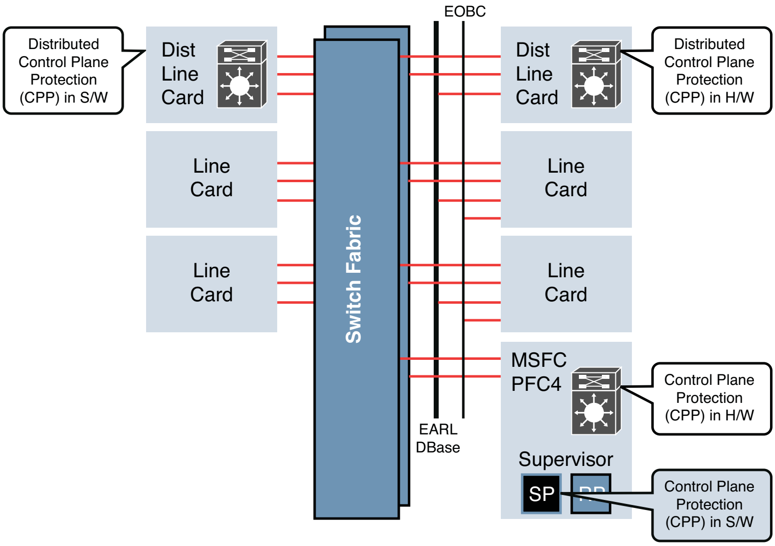

Distributed Control Plane Protection (CPP) in S/W
Dist Line Card
Line Card
Line Card
Switch Fabric
EOBC
Dist Line Card
Distributed Control Plane Protection (CPP) in H/W
Line Card
Line Card
MSFC PFC4
Control Plane Protection (CPP) in H/W
EARL DBase
Supervisor
SP
RP
Control Plane Protection (CPP) in S/W

Figure B-2 *Catalyst 6500 Control Plane Policing Implementation*

Note To minimize redundancy, ACLs and class maps—which are identical to the previous example—are not repeated here.

Example B-2 *Catalyst 6500 Control Plane Policing Configuration Example*

```
! This section defines the CoPP policy
C6500-E(config)# policy-map COPP-POLICY
C6500-E(config-pmap)# class COPP-ACL-BGP
C6500-E(config-pmap-c)# police rate 500 pps burst 50
C6500-E(config-pmap-c-police)# conform-action transmit
C6500-E(config-pmap-c-police)# exceed-action drop
 ! Polices BGP to 4 Mbps
C6500-E(config-pmap)# class COPP-ACL-IGP
C6500-E(config-pmap-c)# police rate 50 pps burst 5
C6500-E(config-pmap-c-police)# conform-action transmit
C6500-E(config-pmap-c-police)# exceed-action drop
 ! Polices IGP to 300 Kbps
C6500-E(config-pmap)# class COPP-ACL-INTERACTIVE-MANAGEMENT
C6500-E(config-pmap-c)# police rate 100 pps burst 10
C6500-E(config-pmap-c-police)# conform-action transmit
C6500-E(config-pmap-c-police)# exceed-action drop
 ! Polices Interactive Management to 500 Kbps
C6500-E(config-pmap)# class COPP-ACL-FILE-MANAGEMENT
```

```
C6500-E(config-pmap-c)# police rate 500 pps burst 50
C6500-E(config-pmap-c-police)# conform-action transmit
C6500-E(config-pmap-c-police)# exceed-action drop
 ! Polices File Management to 6 Mbps
C6500-E(config-pmap)# class COPP-ACL-MONITORING
C6500-E(config-pmap-c)# police rate 125 pps burst 12
C6500-E(config-pmap-c-police)# conform-action transmit
C6500-E(config-pmap-c-police)# exceed-action drop
 ! Polices Monitoring to 900 Kbps
C6500-E(config-pmap)# class COPP-ACL-CRITICAL-APPLICATIONS
C6500-E(config-pmap-c)# police rate 125 pps burst 12
C6500-E(config-pmap-c-police)# conform-action transmit
C6500-E(config-pmap-c-police)# exceed-action drop
 ! Polices Critical Applications to 900 Kbps
C6500-E(config-pmap-)# class COPP-ACL-UNDESIRABLE
C6500-E(config-pmap-c)# police rate 10 pps burst 1
C6500-E(config-pmap-c-police)# conform-action drop
C6500-E(config-pmap-c-police)# exceed-action drop
 ! Polices all Undesirable traffic (conform action is drop)
C6500-E(config-pmap)# class class-default
C6500-E(config-pmap-c)# police rate 100 pps burst 10
C6500-E(config-pmap-c-police)# conform-action transmit
C6500-E(config-pmap-c-police)# exceed-action drop
 ! Polices all other Control Plane traffic to 500 Kbps

 ! This section attaches the CoPP policy to the control plane
C6500-E(config)#control-plane
C6500-E(config-cp)# service-policy input COPP-POLICY
 ! Attaches CoPP policy to control plane
```

You can verify the configuration in Example B-2 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map control-plane**

Cisco IOS Control Plane Policing (for ASR and ISR Routers)

In Cisco IOS, CPP is implemented in software and comes into play right after the routing decision and before traffic is forwarded to the control plane, as shown in Figure B-3. In addition, Cisco IOS CPP allows for a CPP policy to be applied in the input/output directions.

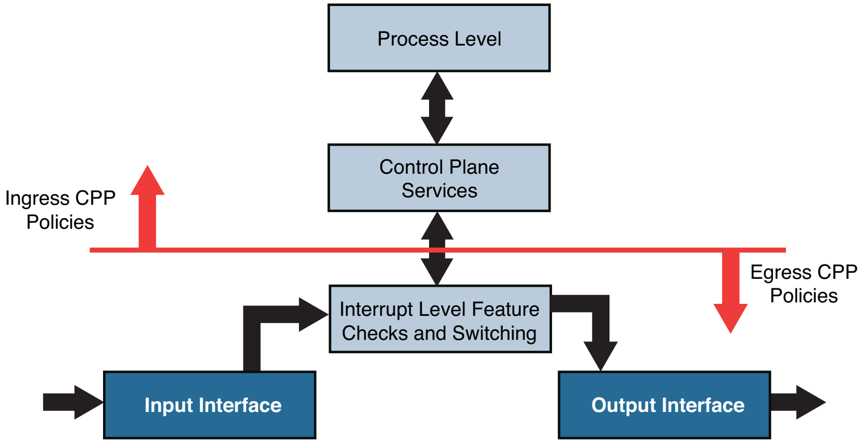


Figure B-3 *Cisco IOS Control Plane Policing Operation*

When CPP is enabled, the sequence of events (at a high level) is as follows:

1. A packet enters the router (configured with CPP) on the ingress interface.
2. The interface performs any applicable input port and QoS services.
3. The packet gets forwarded to the router CPU.
4. The CPU makes a routing or switching decision, determining whether the packet is destined for the control plane.
5. Packets destined for the control plane are processed by CPP and are dropped or delivered to the control plane according to each traffic class policy. Packets that have other destinations are forwarded normally.
6. (Optional) Packets originating from the control plane may also be subject to outbound CPP policies, to prevent the router from being utilized as a source of network reconnaissance or DoS attack.

The corresponding CPP configuration for the Cisco IOS routers, based on the recommendations previously defined, is shown in Example B-3.

Note As previously mentioned, Cisco IOS CPP allows the ability to configure rate limits based on either bits per second or packets per second. When configuring CPP, rate limiting using a pps rate is preferred because it is the per-packet processing that more significantly impacts CPU utilization than the bps rate. The pps rates in the following example are based on Table B-1.

Note To minimize redundancy, ACLs and class maps—which are nearly identical to those in Example B-1—are not repeated here. The only difference is that the prefix CPP is used in ACL and class map and policy map names in this example, as compared to CoPP in the previous examples.

Example B-3 *Cisco IOS Control Plane Policing Configuration Example*

```
! This section defines the CPP-Policy policy map
Router(config)# policy-map CPP-POLICY
Router(config-pmap)# class CPP-ACL-BGP
Router(config-pmap-c)# police rate 500 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices BGP control plane traffic to 500 pps
Router(config-pmap)# class CPP-ACL-IGP
Router(config-pmap-c)# police rate 50 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices IGP control plane traffic to 50 pps
Router(config-pmap)# class CPP-ACL-INTERACTIVE-MANAGEMENT
Router(config-pmap-c)# police rate 100 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices Management control plane traffic to 100 pps
Router(config-pmap)# class CPP-ACL-FILE-MANAGEMENT
Router(config-pmap-c)# police rate 500 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices File-Mgmt control plane traffic to 500 pps
Router(config-pmap)# class CPP-ACL-MONITORING
Router(config-pmap-c)# police rate 125 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices Monitoring control plane traffic to 125 pps
Router(config-pmap)# class CPP-ACL-CRITICAL-APPLICATIONS
Router(config-pmap-c)# police rate 125 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices Critical Applications control plane traffic to 125 pps
Router(config-pmap)# class CPP-ACL-UNDESIRABLE
Router(config-pmap-c)# police rate 10 pps
Router(config-pmap-c-police)# conform-action drop
Router(config-pmap-c-police)# exceed-action drop
```

```
 ! Polices Undesirable control plane traffic to (effectively) 0 pps
 ! As both the conform and exceed action are set to drop
Router(config-pmap)# class class-default
Router(config-pmap-c)# police rate 100 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices all other control plane traffic to 100 pps

! This section applies the CPP policy to the control plane
! The CPP policy is applied in both directions
Router(config)# control-plane
Router(config-cp)# service-policy input CPP-POLICY
 ! Attaches the CPP-POLICY to the control plane in the input direction
Router(config-cp)# service-policy output CPP-POLICY
 ! Attaches the CPP-POLICY to the control plane in the output direction
```

You can verify the configuration in Example B-3 with the following commands:

- **show class-map**
- **show policy-map**
- **show policy-map control-plane** { **input** | **output** }

Additional Reading

Cisco Enterprise Medianet Campus QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSCampus_40.html

Cisco Enterprise Medianet WAN QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html

Cisco White Paper: Deploying Control Plane Policing: http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6586/ps6642/prod_white_paper0900aecd804fa16a.html

Characterizing and Tracing Packet Floods Using Cisco Routers: http://www.cisco.com/en/US/tech/tk59/technologies_tech_note09186a0080149ad6.shtml

Cisco Catalyst 4500 Control Plane Policing Configuration Guide: http://www.cisco.com/en/US/docs/switches/lan/catalyst4500/12.2/15.02SG/configuration/guide/cntl_pln.html

Cisco Catalyst 6500 Control Plane Policing Configuration Guide: http://www.cisco.com/en/US/docs/switches/lan/catalyst6500/ios/15.0SY/configuration/guide/control_plane_policing_copp.html

Cisco IOS Control Plane Policing Configuration Guide: http://www.cisco.com/en/US/docs/ios-xml/ios/qos_plcshp/configuration/15-0m/qos-plcshp-ctrl-pln-plc.html