IPv6 Fundamentals

A Straightforward Approach to Understanding IPv6

Second Edition

ciscopress.com

Rick Graziani

IPv6 Fundamentals: A Straightforward Approach to Understanding IPv6

Second Edition

Rick Graziani

Cisco Press

800 East 96th Street

Indianapolis, IN 46240

IPv6 Fundamentals: A Straightforward Approach to Understanding IPv6, Second Edition

Rick Graziani

Warning and Disclaimer

This book is designed to provide information about IPv6 (Internet Protocol version 6). 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 author, 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.

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 email at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Trademark Acknowledgments

Special Sales

Editor-in-Chief: Mark Taub

Product Line Manager: Brett Bartow

Business Operation Manager, Cisco Press: Ronald Fligge

Executive Editor: Mary Beth Ray

Managing Editor: Sandra Schroeder

Development Editor: Ellie Bru

Project Editor: Mandie Frank

Copy Editor: Kitty Wilson

Technical Editors: Jim Bailey, Tim Martin

Editorial Assistant: Vanessa Evans

Cover Designer: Chuti Prasertsith

Composition: codeMantra

Indexer: Cheryl Lenser

Proofreader: Larry Sulky

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 Author

Rick Graziani has been an instructor of computer networking and computer science courses at Cabrillo College in Aptos, California since 1994. Rick also teaches networking courses in the Computer Engineering department at the University of California, Santa Cruz and is on the Curriculum Engineering team for Cisco Networking Academy. Prior to teaching, he worked in the information technology field for Santa Cruz Operation, Tandem Computers, Lockheed Missiles and Space Company, and served five years in the U.S. Coast Guard. When he is not working, he is most likely surfing at one of his favorite Santa Cruz surf breaks or hanging out with his dog, Luigi. You are welcome to use his instructional resources on his Cabrillo College website, www.cabrillo.edu/~rgraziani, for IPv6, CCNA, or CCNP information. You can email graziani@cabrillo.edu to obtain the username and password for all his materials.

Rick and Luigi

About the Technical Reviewers

Jim Bailey, CCIE No. 5275 (Routing & Switching; Service Provider) and CCDE No. 20090008, is a Solution Architect at Cisco Systems with more than 25 years of experience in networking. As part of the Cisco Advanced Services team, he works on the architecture, design, and implementation of networks for enterprise, service provider, and government customers. He has focused on IPv6 integration into those networks for more than 12 years. He has presented *IPv6 Planning, Deployment and Operational Considerations* at Cisco Live conferences. He has served as a technical reviewer for the *IPv6 for Enterprise Networks* and *IPv6 Design and Deployment LiveLessons* published by Cisco Press.

Tim Martin, CCIE No. 2020, is a dynamic presenter and a member of the Cisco Live Distinguished Speaker hall of fame. Tim frequently speaks at Cisco Live events in both the United States and Europe. Tim has been in the inter-networking industry for more than 34 years. Cisco Press recently published a title of his work *IPv6 Design & Deployment Live Lessons* (ISBN 9780134655512), a 6-hour video series that provides guidance on IPv6 enterprise design. In his current role at Cisco, he is a Solutions Architect focused on the U.S. public sector market. Tim achieved the distinction of a multi-protocol CCIE No. 2020 in June 1996 and has also attained the Gold Certified Engineer status from the IPv6 Forum. He has participated in numerous industry events related to IPv6 and contributes to the IETF IPv6 subcommittees. Tim is also a member of many different IPv6-related task forces, including the FEDv6TF, NAv6TF, TXv6TF, and RMv6TF.

Dedication

To brothers Frank and Mark. You are not only my brothers but you are my best friends. I love you both. Also, to all of my current and former students. I am humbled by the opportunity to teach such wonderful people. You make my job fun, and you are the reason I love to go to work every day.

Acknowledgments

First of all, I would like to thank my family their love and support. Family is the best.

I would like to thank all my friends and colleagues for their assistance. Mark Boolootian, Dave Barnett, and Jim Warner, thanks for the many years of discussing technologies and answering questions. We've drawn a lot of topologies on a lot of napkins over the years.

The technical editors Jim Bailey and Tim Martin at Cisco Systems deserve much more credit than the brief mention as technical reviewers for this book. They make me look a lot smarter than I am. They did an incredibly thorough job making sure that this book is as accurate and as up to date as possible. Their expertise and experience were invaluable in helping me author this book. They are both the unsung heroes of this project. Thank you for your dedication and your commitment.

I owe a great deal of gratitude to Gerlinde Brady, Mike Matera, and Rich Simms at Cabrillo College for their friendship and support. You made sure that the CS/CIS departments at Cabrillo College continued to run smoothly while I was engaged in the writing process. I feel very fortunate to work with all of you and all of our other friends in the CS/CIS department. Thank you David Hovey and Ahmad Allulu for lab support. Thank you Brad Smith, Patrick Mantey, J.J. Garcia-Luna Aceves, and Katia Obraczka for the privilege of teaching in the Computer Science and Engineering department at the University of California, Santa Cruz. Teaching students and working with faculty at UCSC have made this a much better book.

Writing this book has been one of many privileges I have received due to the honor of working with Dennis Frezzo, Jeremy Creech, Telethia Willis, and many others who work for the Cisco Networking Academy. Thank you all for the proud opportunity to be part of a program that has changed the lives of thousands of students around the world. More than colleagues, you are all friends, for which I am grateful.

Thank you, Pat Farley, for making sure I still get in my surf sessions. Now that the book is done, you'll see me in the lineup much more often. Thank you, Teri Graziani, for always being there and taking care of things while I was busy writing this book. I appreciate it more than you know.

Special thanks to Mary Beth Ray, executive editor for Cisco Press and a friend. Thank you for your patience and understanding through this long process. You always have that calm assurance and guidance.

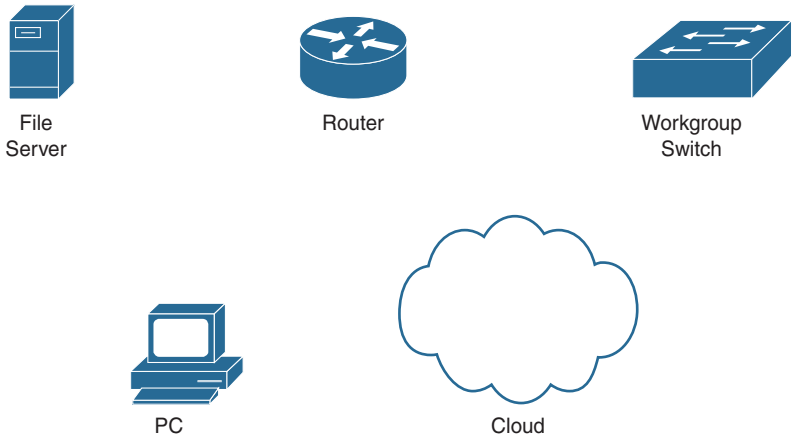
Thank you, Ellie Bru at Cisco Press, for working with me on a daily basis—weekdays and weekends—editing, formatting, and orchestrating the entire process. You were a pleasure to work with, and I am very grateful for all the hard work you put into this book. To Mandie Frank, Dhayanidhi, Kitty Wilson, and Larry Sulky, thank you for making me look like a better writer than I really am. And to everyone else at Cisco Press, I am extremely grateful for everything you have done. I am constantly amazed at the level of cooperation and teamwork required to produce a technical book. I am very thankful for all your help.

Finally, thank you to all my friends in Rome and Frascati, Italy, where I spent many months writing this book. Thank you, Giuseppe Cinque, for your many years of friendship and for convincing me to spend my sabbatical in Italy writing this book. Thank you, Mama and Papa Cinque, for making me part of your Positano family. Thank you, Levi Adam, Fidele Lassandro, Antonio Brancaccia, Daniel and Andrea and everyone at Tusculum Sport Center, Fermate N'Attimo, everyone at Il Borgo Verde, the Molinari family at Antico Forno, and everyone at Etabli for your friendship. And a special thank you to Alice, Mauro, Loredana, Marco, and Timmy Chialastri, for making Rome a second home. Your kindness made the time I spent in Rome some of the most enjoyable months I have ever had. Thank you for opening up your home and your hearts to me. I will forever be grateful for everything you did for me.

Contents at a Glance

Contents

Icons Used in This Book



Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a **show** command).
- *Italics* indicate arguments for which you supply actual values.
- Vertical bars | separate alternative, mutually exclusive elements.
- Square brackets [ ] indicate optional elements.
- Braces { } indicate a required choice.
- Braces within brackets [{ }] indicate a required choice within an optional element.

Introduction

This book is intended to give you an in-depth understanding of IPv6 and related protocols. This book is for those who are new to the networking field, such as computer networking students, and also for network engineers with many years of experience administering larger enterprise networks. The only prerequisite is a basic understanding of networking protocols, including IPv4.

This second edition is a complete reorganization and almost a complete rewrite of the first edition, with a lot of new content. I have been a teacher for over 20 years, and this book is written for students of IPv6, as a self-study guide for learning IPv6. This book is designed to walk you through learning about IPv6 as if you were in a classroom, with the instructor explaining each concept. The information has been organized to help both those who want to read the book from cover to cover and also for those looking for specific information.

There is a great deal to learn about IPv6, and it is much more than just becoming familiar with a larger address. A brief look at the contents of this book will give you an idea of what is covered and what is needed to have a good understanding of IPv6.

My approach to writing this book was to do my best to explain each concept in a simple, step-by-step approach, as well as to include the critical details. It was a challenging balance between providing as much information as possible and not overwhelming you, the reader. IPv6 is not difficult to learn but involves multiple protocols and processes that might be new to some.

Don't be overwhelmed by all the details. For example, although I have included a brief description of each field in the protocols discussed in the book, it isn't necessarily important that you understand the details of each one. I mention this throughout the book. But I did feel it necessary not to leave out or hide these details from you.

RFCs are cited throughout the book. It was important to include these references for two reasons. First of all, I wanted to give you the authoritative source for the material in this book so that you have resources for more information. Second, IPv6 is currently and will continue to be a moving target for quite some time. Although it has been around for many years, additional development and fine-tuning are still taking place. If you are not familiar with reading RFCs, don't be intimidated. Most of them are not difficult to read, and they do their best to explain the topic clearly.

Review questions are included at the end of each chapter to help you understand some of the fundamental concepts discussed in the chapter. The review questions provide a high-level overview of some the key points discussed in the chapter. They are not meant as a detailed assessment of all the material covered in the chapter. An "IPv6 Command Quick Reference" has been included as Appendix B for Cisco IOS, Windows, Linux, and Mac OS X commands.

At times in this book I introduce a technology or concept but state that it is covered in more detail in a later chapter. I do this to explain the concept as it relates to the topic being discussed without getting lost in the details. The details are covered where appropriate. I feel it is better to revisit some of the more advanced topics after you have

a more complete understanding of the entire IPv6 topic. At times I state that a concept is "beyond the scope of this book." I suggest resources for those who might be interested in learning more about those topics.

The objective of this book is to explain IPv6 as clearly as possible. At times it was like herding cats, trying to decide which topic to cover first. Chapter 2 is an IPv6 primer designed to give an overview of the main topics. Having this overview will make it easier as you progress through the rest of the book.

Readers are welcome to use the resources on my website, www.cabrillo.edu/~rgraziani, for IPv6, CCNA, or CCNP information. You can email me at graziani@cabrillo.edu to obtain the username and password for all my materials.

Goals and Methods

The most important goal of this book is to provide a thorough yet easy-to-understand introduction to IPv6. It is written for both computer networking students and seasoned network engineers. This book is also intended to provide a foundation in IPv6 that will allow you to build on. It explains topics that might be a little more challenging to grasp.

Another goal of this book is to be a resource for IPv6. The book is organized to make it as easy as possible to find information on specific topics. I have included command syntax, RFCs, and links to Cisco white papers to help guide you toward a better understanding of many of the topics.

Who Should Read This Book

This book is intended for anyone seeking a solid understanding of the fundamentals of IPv6, such as network engineers, network designers, network technicians, technical staff, and networking students, including those who are part of Cisco Networking Academy. You should have a basic familiarity with IPv4 and networking protocols before you begin reading this book.

Professionals deploying or planning to deploy IPv6 within a network will find this book useful. It provides examples, figures, IOS commands, and recommendations for configuring Cisco IOS IPv6 technology. Although Cisco devices are used in this book, those using non-Cisco equipment will also find this book helpful. The vast majority of protocols and technologies are IETF standards. Configuration and verification commands for Windows, Linux, and Mac OS are also included throughout the book.

How This Book Is Organized

If you are new to IPv6, you should read this book from cover to cover. However, if you have some knowledge of IPv6, it is designed to be flexible and allows you to easily move between chapters and sections of chapters to cover just the material you want to review. A common topology is used throughout the book except in a few cases.

Chapters 1 through 3 provide an introduction to IPv6, the reasons for moving to IPv6, an IPv6 primer, and a comparison of the IPv4 and IPv6 protocols. Chapters 4 through 7 discuss the different types of IPv6 address, including how to represent IPv6 addresses, global unicast addresses, link-local unicast addresses, and IPv6 multicast addresses. Chapters 8 through 11 discuss dynamic IPv6 addressing methods. Dynamic address allocation differs significantly in IPv6 and IPv4. These chapters discuss Stateless Address Autoconfiguration (SLAAC), stateless DHCPv6, and stateful DHCPv6. The chapter on SLAAC discusses the reason for permanent and temporary addresses and how to manage them using Cisco IOS and host operating systems. These chapters include the Cisco IOS commands and configuration examples for stateless and stateful DHCPv6. Chapters 12 and 13 discuss ICMPv6 and ICMPv6 Neighbor Discovery Protocol. These protocols and messages are introduced in earlier chapters, beginning with Chapter 2. Chapters 12 and 13 examine ICMPv6 and ICMPv6 Neighbor Discovery in more detail. Chapters 14 through 16 cover routing IPv6, including the IPv6 routing table, classic EIGRP for IPv6, EIGRP named mode, traditional OSPFv3, and OSPFv3 with address families. The last chapter, Chapter 17, introduces deploying IPv6 and transitioning from IPv4 to IPv6. In case you intend to read all the chapters, the order in the book is sequential.

The following list highlights the topics covered in each chapter and the book's organization:

- **Chapter 1, "Introduction to IPv6":** This chapter discusses how the Internet of today requires a new network layer protocol, IPv6, to meet the demands of its users. It also examines the limitations of IPv4 and describes how IPv6 resolves these issues while offering other advantages as well. This chapter examines the rationale of IPv6 and concerns regarding IPv4 address depletion. It presents a brief history of both IPv4 and IPv6. The IPv4 migration technologies CIDR and NAT are also discussed.
- **Chapter 2, "IPv6 Primer":** This chapter introduces some of the basic concepts and protocols that are explained in more detail throughout the rest of the book, including IPv6 address types, the basics of dynamic address allocation, and the hexadecimal number system, which is used to represent IPv6 addresses. This chapter gives an overview of some IPv6 concepts that are helpful in learning IPv6. This chapter also highlights many of the differences in IPv6.
- **Chapter 3, "Comparing IPv4 and IPv6":** This chapter compares and contrasts the IPv4 and IPv6 protocols. It also explores how fragmentation is handled. It discusses the IPv6 extension headers as well.
- **Chapter 4, "IPv6 Address Representation and Address Types":** This chapter introduces IPv6 addressing and address types. It discusses representation of IPv6 addresses, along with the different formats for representing IPv6 addresses and the rules for compressing the IPv6 notation. This chapter provides an introductory look at the different types of IPv6 addresses, including unicast, multicast, and anycast. It also discusses prefix length notation.
- **Chapter 5, "Global Unicast Address":** This chapter examines the global unicast address in detail. It examines the different parts of a global unicast address as well as

manual configuration of a global unicast address for Cisco IOS and host operating systems. The chapter also covers subnetting of IPv6, along with prefix allocation.

- **Chapter 6, "Link-Local Unicast Address":** This chapter examines link-local addresses and includes static and dynamic link-local address configuration examples. It explains the EUI-64 process, along with the significance of a link-local address in IPv6.
- **Chapter 7, "Multicast Addresses":** This chapter examines multicast addresses, including well-known and solicited-node multicast. It discusses the advantages of a multicast address over a broadcast address (the broadcast address does not exist in IPv6) and how IPv6 multicast addresses are mapped to Ethernet MAC addresses.
- **Chapter 8, "Basics of Dynamic Addressing in IPv6":** This chapter introduces and compares the three methods of dynamic address allocation: Stateless Address Autoconfiguration (SLAAC), stateless DHCPv6, and stateful DHCPv6. The chapters that follow discuss these methods in more detail.
- **Chapter 9, "Stateless Address Autoconfiguration (SLAAC)":** This chapter discusses the SLAAC process in detail. It includes a Wireshark examination of an ICMPv6 Router Advertisement message suggesting SLAAC. This chapter discusses the use of the privacy extension and temporary addresses with SLAAC-generated addresses, including the different states and lifetimes. It also explains how to manage privacy options on host operating systems.
- **Chapter 10, "Stateless DHCPv6":** This chapter examines SLAAC and other stateless DHCPv6 services. It covers DHCPv6 terminology and message types, along with the DHCPv6 process between the client and server. This chapter explains the rapid-commit option and relay agents.
- **Chapter 11, "Stateful DHCPv6":** This chapter examines stateful DHCPv6 services, similar to DHCP for IPv4. It also introduces a common method for providing IPv6 address space to homes using DHCPv6 with Prefix Delegation.
- **Chapter 12, "ICMPv6":** This chapter examines ICMPv6, which is a much more robust protocol than ICMPv4. It covers ICMPv6 error messages, including Destination Unreachable, Packet Too Big, Time Exceeded, and Parameter Problem. It also covers the ICMPv6 Echo Request and Echo Reply informational messages, along with Multicast Listener Discovery messages.
- **Chapter 13, "ICMPv6 Neighbor Discovery":** This chapter examines ICMPv6 Neighbor Discovery, including Router Solicitation, Router Advertisement, Neighbor Solicitation, Neighbor Advertisement, and Redirect messages. Not only does IPv6 resolve larger address space issues but ICMPv6 with Neighbor Discovery Protocol also presents a major change in network operations, including link-layer address resolution (ARP in IPv4), Duplicate Address Detection (DAD), Stateless Address Autoconfiguration (SLAAC), and Neighbor Unreachability Detection (NUD). This chapter discusses the IPv6 neighbor cache and neighbor cache states, similar to those of the IPv4 ARP cache.

- **Chapter 14, "IPv6 Routing Table and Static Routes":** This chapter examines the IPv6 routing table. It also discusses the configuration of IPv6 static routes, which are similar to static routes for IPv4. It explains IPv6 default routes and route summarization, as well as CEF for IPv6.
- **Chapter 15, "EIGRP for IPv6":** This chapter discusses EIGRP for IPv6. It begins with a comparison of EIGRP for IPv4 and EIGRP for IPv6. It discusses configuration and verification of classic EIGRP for IPv6 and EIGRP named mode (for IPv4 and IPv6).
- **Chapter 16, "OSPFv3":** This chapter discusses OSPFv3. It begins with a comparison of OSPFv2 (IPv4 only), traditional OSPFv3 (IPv6 only), and OSPFv3 with address families (IPv4 and IPv6). It also discusses configuration and verification of traditional OSPFv3 and OSPFv3 with address families.
- **Chapter 17, "Deploying IPv6 in the Network":** This chapter covers strategies for deploying IPv6, including creating an IPv6 address plan, configuring IPv6 VLANs, and implementing transparent failover at the first-hop router, using ICMPv6 or a first-hop redundancy protocol (FHRP). This chapter also discusses IPv4 and IPv6 integration and coexistence, including dual stacking, NAT64, and tunneling.
- **Appendix A, "Configuring NAT64 and IPv6 Tunnels":** This appendix provides configuration examples and additional information on NAT64 and IPv6 tunnels, introduced in Chapter 17.
- **Appendix B, "IPv6 Commands Quick Reference":** This appendix provides a summary of the Cisco IOS, Windows, Linux, and Mac OS commands used in this book.
- **Appendix C, "Answers to Review Questions":** This appendix provides the answers to the Review Questions at the end of each chapter.

This page intentionally left blank

Part I

Introduction to IPv6

This page intentionally left blank

Chapter 1

Introduction to IPv6

"I am a little embarrassed about that because I was the guy who decided that 32-bit was enough for the Internet experiment. My only defense is that that choice was made in 1977, and I thought it was an experiment. The problem is the experiment didn't end, so here we are."

—Vint Cerf, speaking in 2011, credited as one of the co-founders of the Internet

Internet Protocol version 6 (IPv6), the successor to Internet Protocol version 4 (IPv4), has been a long time coming. For more than 35 years, IPv4 has been an integral part of the Internet evolution. But it was designed at a time when there were fewer than 600 hosts on the Internet, before the World Wide Web, email, video streaming, mobile phones, and countless other innovations. This chapter discusses the limitations of IPv4 and how after many years of proclaiming the need for IPv6, IPv6 is really finally here.

IPv6 Is Here

IPv4 provides for a maximum 4.29 billion 32-bit addresses, which seemed like more than enough addresses when IPv4 became a standard in 1980. There were about 4.5 billion people on Earth at the time, so even if every person on the planet needed one IPv4 address, it seems like that should have been enough.

The number of IP addresses needed today far exceeds the world's population for several reasons. First of all, IPv4 addresses are often allocated in groups of addresses, as network addresses. Places such as companies, schools, homes, cafés, and airports are allocated network addresses for their users. As I go from my home to the café, or to my college, I receive a different IP address to access each network. So, in most cases, I need a different IPv4 address wherever I go.

But a much larger reason we need so many more IP addresses is the number of devices per person that are being connected to the Internet—and the number of people who still need access to the Internet. A lot of the devices that have Internet connections are obvious: computers, laptops, mobile phones, and tablets. But then there are also devices that many people don't realize have IP addresses, such as cable or satellite subscriber set-top boxes and cars. Beginning in 2016, statistics compiled by Chetan Sharma Consulting showed that U.S. Internet service providers (ISPs) connected more new cars to their networks than they did new mobile phones.[1] We will examine this more closely later in this chapter.

IPv6 provides more addresses than IPv4. As mentioned earlier, IPv4, with its 32-bit address space, provides for 4,294,967,296, or 2^{32}, addresses. In comparison, the 128-bit IPv6 address space provides for 340 undecillion addresses! How large is 340 undecillion? It equates to 340,282,366,920,938,463,463,374,607,431,768,211,456, or 2^{128}. As you can see, IPv6 provides a tremendous—almost unimaginable—number of addresses.

But IPv6 is much more than just a lot of addresses. The designers of IPv6 took the opportunity to improve IP and related protocols. Throughout this book we discuss the many changes and improvements made to IPv6 along with other protocols, such as ICMPv6.

Many people have been skeptical about the need for IPv6. The proponents of IPv6 have been preaching the need to move to IPv6 since the mid-1990s. And yet, after all these years, many people still don't see the need for IPv6.

However, after countless proclamations that the IPv4 sky is falling, a more mature IPv6 is finally being deployed at a growing rate. IPv6 is now enabled by default on every major host operating system, including Windows, Mac OS, and Linux. All mobile operating systems are also IPv6-enabled, including Google Android, Apple iOS, and Windows Mobile. Because of the enormous number of mobile devices, the mobile carriers are leading the way in IPv6 deployments. For example, World IPv6 Launch measurements showed that over 75% of Verizon Wireless's overall traffic is over IPv6[2]. T-Mobile phones are IPv6-only; they use a special protocol translation technique when communicating with IPv4-only devices.[3] Gaming stations such as Microsoft's Xbox One and Sony's PlayStation are also IPv6-enabled. IPv6 is here.

Note For information and presentation material from service providers, search for "IPv6" at www.nanog.org/archives/presentations, the North American Network Operators' Group (NANOG) meeting presentation archives.

Comcast, the world's largest media company, has enabled IPv6 throughout its network, which includes the majority of residences in the United States. Most Comcast residential customers don't realize that they may be using IPv6 when communicating with Google, Facebook, Netflix, LinkedIn, YouTube, and many other content providers. As of this writing, more than 50% of Internet web content is available over IPv6. The four major

US mobile providers—Verizon Wireless, T-Mobile USA, Sprint Wireless, and AT&T Wireless—send the majority of their traffic over IPv6 to major IPv6-capable content providers.

IPv6 is here and is being adopted…finally. Given all these pronouncements over the years, it's easy to ignore seeing the need to transition to IPv6. Since 1996, we have been hearing how we must begin to migrate to IPv6, yet the IPv4 Internet has continued to survive and even grow.

What's changed now, so that we have to transition to IPv6? It's simple: We're out of IPv4 addresses. The need for IP addresses is real. It is predicted that the Internet of Things (IoT)—everyday objects having Internet connectivity—will add about another 35 billion devices by 2020. Cloud services such as Amazon AWS and Microsoft Azure now support IPv6. About half of the world's population is yet to get Internet access, particularly in Africa and Asia. Already about 5.5 billion mobile phones in the world require Internet connectivity; and remember that IPv4 allows for only 4.29 billion addresses. We simply don't have the IPv4 address space to accommodate current needs—let alone future growth. (We'll talk more about the details of the depletion of IPv4 address space and the need for more addresses later in this chapter.)

For those who haven't already started the transition to IPv6, now is a good time to begin to become familiar with IPv6. If you're a network administrator, your Windows, Mac, Linux, Android, and iOS clients are already running IPv6. At the very least, it is necessary to understand and secure your IPv6 network to prevent man-in-the-middle (MITM) and denial-of-service (DoS) attacks.

The following sections discuss issues including the following:

- Why transition to IPv6?
- "Short-term" solutions to IPv4 address exhaustion: CIDR and NAT with private IPv4 addresses
- What happened to IPv5?
- The development history of IPv6
- Some of the myths about IPv6
- Transitioning to IPv6

Why Transition to IPv6?

What are the advantages of using IPv6? Really, who cares? If you're a network administrator, you might be thinking, "I have enough work to do, and my IPv4 network works just fine." And for now, you may still have plenty of IPv4 addresses available. However, continuing to ignore IPv6 will begin to have a real business impact on your network. At the very least, ignoring IPv6 could put your network at risk.

The following sections discuss these benefits of IPv6 and reasons to begin the transition to IPv6:

- IPv4 address depletion
- Access to IPv6-only customers
- Better performance
- Securing your current network

IPv4 Address Depletion

There is no getting around it. We are running out of IPv4 address space. Sooner or later, network operators will need to transition their networks to IPv6. Even if you're not ready to implement IPv6 on your network right now, there are many reasons you need to understand IPv6.

As mentioned earlier, choosing to ignore IPv6 as IPv4 exhaustion continues may put your business and your network at risk. IPv4 address depletion is discussed in more detail later in this chapter.

Access to IPv6-Only Customers

Already, there are places in the world that are IPv6-only. In some areas of the world today, it is simply not possible to get an IPv4 address. This will become more and more the norm as IPv4 address space becomes even less available. Unless you want to risk isolating your business from these users, it is going to become increasingly important for your network to be enabled for IPv6.

Yes, there are translation techniques that allow IPv4-only and IPv6-only networks to communicate with each other, but they are not always reliable, and they are often accompanied by degraded performance. Also, any sort of translation mechanism makes it difficult to measure the quality of the user experience and has the potential to break certain applications.

Better Performance

Increased performance can be another benefit of transitioning to IPv6. Many content providers are seeing substantial increases in performance with IPv6. Facebook sees 20% to 40% better performance of news feeds with IPv6. Tests at Time Warner Cable show about a 15% performance increase with IPv6. Asia-Pacific Network Information Centre (APNIC) says that IPv6 traffic is faster over trans-Pacific links due to better routing aggregation.[4]

Studies on why IPv6 may be faster than IPv4 are still incomplete. However, the Internet Society (ISOC) has stated, "The reality is that connections from users on IPv6 networks to services that run over IPv6 *are going to be faster* than connections to services still on legacy IPv4 networks that have to go through middleboxes such as NAT devices or application-layer gateways (ALGs)."[5]

As discussed in more detail later in this chapter, network address translation (NAT) was never intended to be a permanent solution to the IPv4 exhaustion problem. It was meant as a temporary solution. Unfortunately, NAT breaks many applications and adds latency.

Securing Your Current Network

As mentioned previously, many home subscribers are seeing performance benefits thanks to IPv6. This is because residential service providers such as Comcast have enabled IPv6 to all their residential customers. What have customers had to do to enable their computers and mobile devices for IPv6? Nothing. This is because IPv6 has been enabled on all major operating systems—Windows, Mac OS, Linux, iOS, and Android—for years.

Even if your provider isn't running IPv6, your host operating systems are. Example 1-1 shows a Windows host running both IPv4 and IPv6. This is known as *dual stack*, meaning a device is running both the IPv4 and IPv6 protocols at the same time. Notice in Example 1-1 that the Windows host has a link-local IPv6 address. (Link-local addresses are explained Chapter 4, "Link-Local Unicast Address.") An IPv6 device automatically creates an IPv6 link-local address during startup. So without requiring the user to do anything, this Windows device is enabled and running IPv6.

Example 1-1 *Windows Running IPv6 by Default*

```
PC> ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  : localdomain
   Link-local IPv6 Address . . . . : fe80::50a5:8a35:a5bb:66e1%7
   IPv4 Address. . . . . . . . . . : 192.168.1.101
   Subnet Mask . . . . . . . . . . : 255.255.255.0
   Default Gateway   . . . . . . . : 192.168.1.1
```

Note If the **ipconfig** command displays **IPv6 Address** without the preceding **Link-local**, it is a *global unicast address*, which is an Internet-routable IPv6 address. The **ifconfig** command shows similar output on Mac OS and Linux devices.

Because host operating systems are enabled for IPv6 by default, it is critical for network administrators to become familiar with IPv6. A network may be secured against DoS, MITM, and other attacks for IPv4, and it's important to provide the same security measures for IPv6. Host operating systems are ready and willing to use IPv6 to communicate, even if you're not.

IPv4

Before discussing IPv6 in more detail, let's look at the history of IPv4. IPv4 is the current Layer 3 protocol used on the Internet and most other networks. IPv4 has been an integral part of the Internet evolution over the past 35 years. It was originally described in RFC 760 (January 1980) and replaced by RFC 791 (September 1981).

Can you imagine what the Internet was like in those early years? IPv4 was introduced when there were fewer than 600 hosts on the Internet, so its address space of 4.29 billion seemed enormous. Although many of the same concepts of packet switching apply today, obviously the number of users on the Internet and how the Internet is used are vastly different than they were back then. Even with the advent of the World Wide Web in the early 1990s, there were only about 16 million users on the Internet worldwide; in 2016 there were more than 3.6 billion users. And remember that in the early 1990s, there was about one device per user. Today each user has multiple devices. The total number of devices connected to the global Internet today is 15 billion.

The reasons to migrate from IPv4 to IPv6 are discussed later in this chapter. For now, just recognize that presently the Internet is a much different place than it was 30, 10, or even 5 years ago. Today's Internet is much more than just web pages, email, and file transfers. With the explosive growth of mobile devices and peer-to-peer networking, along with the potential impact of Internet-ready consumer goods, we have gone from just an Internet of computers to an *Internet of Things* (*IoT*). Originally the idea of the IoT focused on objects having radio-frequency identification (RFID) tags for identification and inventory purposes. It has since evolved into something much larger, including smart grids, smart and connected cities, home automation, automotive computing, and more. A new and growing trend in the IoT is wearable devices that do everything from measuring your heart rate to projecting an image of your smart phone on your arm.

IPv4 Address Depletion

> *"IPv4 depletion is THE reason to make sure you have a plan to deploy IPv6… IPv6 implementation is the only viable solution to the dwindling pool of IPv4 space."*
>
> —American Registry for Internet Numbers (ARIN)

The number of people accessing the Internet is increasing dramatically. Even with short-term solutions like NAT, we are in the final stages of public IPv4 address availability. Table 1-1 provides some world Internet usage and population statistics, as of June 30, 2016, that reveal some interesting facts regarding IPv4 address allocation. Notice that North America has a penetration rate of about 89%. This means that the vast majority of people in North America use the Internet. Asia, on the other hand, has a penetration rate of only about 46%. The problem becomes apparent when you compare the two populations. Whereas North America has a population of about 360 million, Asia has 4 billion people. As Asia, Africa, and other areas of the world become increasingly connected to the Internet, there is no way IPv4 can keep up, even with NAT.

Table 1-1 *World Internet Usage and Population Statistics as of June 30, 2016*

World Region	Population (2016 Est.)	Population % of World	Number of Internet Users	Penetration Rate	Growth 2000–2016
Africa	1,185,529,578	16.2%	340,783,342	28.7%	7,448.8%
Asia	4,052,652,889	55.2%	1,846,212,654	45.6%	1,515.2%
Europe	832,073,224	11.3%	614,979,903	73.9%	485.2%
Latin America/ Caribbean	626,119,788	8.5%	384,751,302	61.5%	2,029.4%
Middle East	246,700,900	3.4%	141,489,765	57.4%	4,207.4%
North America	359,492,293	4.9%	320,067,193	89.0%	196.1%
Oceania/ Australia	37,590,820	0.5%	27,540,654	73.3%	261.4%
World total	7,340,159,492	100.0%	3,675,824,813	50.1%	918.3%

Note: The Internet penetration rate is the percentage of the population of a country or region that uses the Internet.

Source: The information in this table was obtained from www.internetworldstats.com, copyright © 2001–2016, Miniwatts Marketing Group. All rights reserved worldwide.

There has also been extraordinary growth in the number of devices connected to the Internet. GlobalWebIndex estimates that consumers own 3.64 devices per person, and this number continues to increase every year.[6] When you start tallying up the Internet media devices, smart TVs, smart home devices, wearables, and other IoT devices, you can see the phenomenal number of devices needing Internet connections. This number is, again, much larger than IPv4 can accommodate.

The Internet Assigned Numbers Authority (IANA) assigns IPv4 addresses to the five Regional Internet Registries (RIRs) in /8 address blocks. (IANA is also responsible for allocating IPv6 address space to RIRs.) Using this address space, RIRs then redistribute the IPv4 network addresses to ISPs and other end customers. The five RIRs and the areas they manage are as follows:

- **African Network Information Centre (AFRINIC):** Africa
- **American Registry for Internet Numbers (ARIN):** United States, Canada, some parts of the Caribbean region, and Antarctica
- **Asia-Pacific Network Information Centre (APNIC):** Asia, Australia, New Zealand, and neighboring countries
- **Latin America and Caribbean Network Information Centre (LACNIC):** Central America, South America, and most of the Caribbean region
- **Réseaux IP Européens (RIPE) Network Coordination Centre:** Europe, the Middle East, and Central Asia

On January 31, 2011, IANA allocated the last two blocks of IPv4 address space, 39.0.0.0/8 and 106.0.0.0/8, to APNIC, the RIR for the Asia-Pacific region. This triggered an IANA policy that the remaining five /8 addresses would be distributed equally among the five RIRs. At this point, IANA had now run out of IPv4 addresses.

On September 24, 2015, ARIN, the RIR for North America, had officially run out of IPv4 addresses. Currently, four out of the five RIRs have no more IPv4 addresses to allocate. AFRINIC is the last RIR to have any IPv4 addresses.[7] Although AFRINIC still has plenty of IPv4 address space, it is urging its subscribers to service both IPv6 and IPv4 to avoid lagging behind the rest of the world.[8]

Does this mean that IPv4 addresses are no longer available for end customers? No. Customers can still get IPv4 addresses from most ISPs. However, many ISPs are severely limited. In the wake of all this IPv4 address depletion, a "gray market" for IPv4 addresses has emerged. Several websites are serving as brokers for organizations that want to sell or lease IPv4 address space. ISOC has stated that buyers and sellers should make sure that any transfers use the appropriate RIR processes so that they don't compromise network operations or security.

Later in this chapter we will look at some of the benefits that IPv6 provides. However, the most compelling reason for moving to IPv6 is simply that we are out of IPv4 address. The "killer application" of IPv6 is the Internet itself—preserving the technologies of today and allowing for future growth and innovations.

There has been some inefficiency in the allocation of the 4.29 billion IPv4 addresses over the years. Even if it were possible to reassign the entire IPv4 address space in a more efficient manner, it would be a very short-term solution. IPv6 antagonists continue to find ways to extend the use of IPv4 without preparing to migrate to IPv6. It has often been said that this is a little like rearranging deck chairs on the *Titanic*. It might make things a little more comfortable for a while, but sooner or later, you must come to the realization that you're in trouble.

When Bob Kahn and Vint Cerf first developed the Transmission Control Protocol/Internet Protocol (TCP/IP) suite of protocols, including IPv4 and its 32-bit addresses, they never envisioned that the Internet would be what it is today—something that many rely on not just for communications but also for a variety of economic, social, political, and educational reasons.

The eventual exhaustion of IPv4 address space was recognized in the early 1990s, even before the World Wide Web came to be. And then the advent of the World Wide Web brought an explosion in the size and diversity of the Internet population. The people using the Internet were no longer limited to a relatively small group of computer-savvy users who could navigate this unknown global network using slightly cumbersome and less-than-intuitive tools such as File Transfer Protocol (FTP), newsgroups, and Unix-to-Unix Copy (UUCP). With Tim Berners-Lee's development of Hypertext Transfer Protocol (HTTP) at the European Organization for Nuclear Research (CERN) in Geneva, followed by the first web browser, Mosaic, created by Marc Andreessen and his team at the

National Center for Supercomputing Applications at the University of Illinois at Urbana–Champaign, all of a sudden, people who had never owned a personal computer were accessing the Internet. It was becoming clear that 4.29 billion addresses were not going to last long.

So, in the early 1990s, the Internet Engineering Task Force (IETF) began development of a new version of IP known as IP Next Generation, which later became IPv6. However, IPv6 was a long-term solution, and a short-term solution was needed immediately. Several short-term solutions were put into place. Two of the most important, discussed in the following sections, were Classless Inter-Domain Routing (CIDR) and NAT with private IPv4 addresses.

CIDR

In the early years, IPv4 network addresses were allocated using one of three classes—Class A, Class B, or Class C—as shown in Table 1-2.

Table 1-2 *Classful Unicast IPv4 Addresses*

Address Class	First Octet Range	Number of Possible Networks	Number of Hosts per Network
Class A	0 to 127	128 (2 are reserved)	16,777,214
Class B	128 to 191	16,384	65,534
Class C	192 to 223	2,097,152	254

There were a total of five different classes:

- **Class A:** Has a subnet mask of 255.0.0.0, or /8
- **Class B:** Has a subnet mask of 255.255.0.0, or /16
- **Class C:** Has a subnet mask of 255.255.255.0, or /24
- **Class D:** Used for multicast
- **Class E:** Experimental

Class A networks, with a /8 mask, can each have a total of 16,777,214 (2^{24}) hosts (with two addresses reserved for the network and broadcast addresses). The 128 (2^{7}) Class A networks and the enormous number of hosts per network account for 50% of the total IPv4 address space, as shown in Figure 1-1.

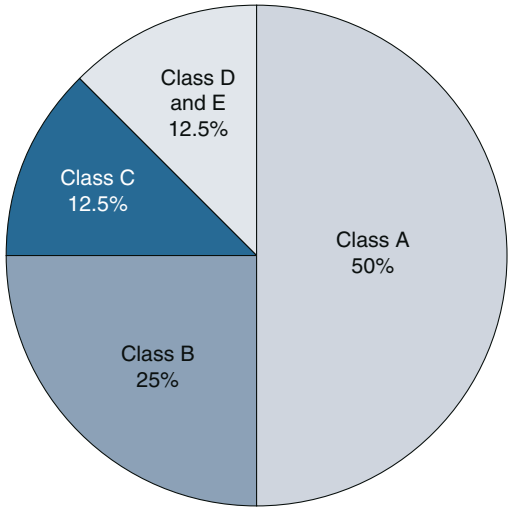


Figure 1-1 *Classful IPv4 Address Allocation*

Class B networks have a /16 mask, which means they can have a total of 65,534 (2^{16}) hosts per network. The 16,384 (2^{14}) Class B networks represent 25% of all IPv4 addresses.

Class C networks, with a /24 mask, can have a maximum of only 254 (2^{8}) hosts for each Class C network address. Even though there are 2,097,152 (2^{21}) Class C networks available, this represents only 12.5% of the total available IPv4 address space.

Because network administrators could obtain a network address from only one of these three classes, it made for a very inefficient method of allocating addresses. Most Class A and B networks had enormous numbers of unused addresses, while most Class C networks had comparatively very few.

In 1992, the IETF replaced this allocation method with Classless Inter-Domain Routing (CIDR, pronounced *cider*), which is described in RFC 1338 and later in RFC 1519. Network addresses were no longer allocated based on one of three classes. The addresses could be assigned with any size mask, as shown in Figure 1-2. (Note that Figure 1-2 shows the total number of possible host addresses and does not subtract two addresses for the network and broadcast addresses.) This is in contrast to the limitations of allocating a prefix length of just /8 (Class A), /16 (Class B), or /24 (Class C). The additional prefix lengths gave RIRs and ISPs more flexibility in allocating addresses to customers and made for a much more efficient distribution of the limited IPv4 address space.

Class A	**11111111.00000000.00000000.00000000**	**/8 (255.0.0.0)**	**16,777,216 host addresses**
	11111111.10000000.00000000.00000000	/9 (255.128.0.0)	8,388,608 host addresses
	11111111.11000000.00000000.00000000	/10 (255.192.0.0)	4,194,304 host addresses
	11111111.11100000.00000000.00000000	/11 (255.224.0.0)	2,097,152 host addresses
	11111111.11110000.00000000.00000000	/12 (255.240.0.0)	1,048,576 host addresses
	11111111.11111000.00000000.00000000	/13 (255.248.0.0)	524,288 host addresses
	11111111.11111100.00000000.00000000	/14 (255.252.0.0)	264,144 host addresses
	11111111.11111110.00000000.00000000	/15 (255.254.0.0)	131,072 host addresses
Class B	**11111111.11111111.00000000.00000000**	**/16 (255.255.0.0)**	**65,536 host addresses**
	11111111.11111111.10000000.00000000	/17 (255.255.128.0)	32,768 host addresses
	11111111.11111111.11000000.00000000	/18 (255.255.192.0)	16,384 host addresses
	11111111.11111111.11100000.00000000	/19 (255.255.224.0)	8,192 host addresses
	11111111.11111111.11110000.00000000	/20 (255.255.240.0)	4,096 host addresses
	11111111.11111111.11111000.00000000	/21 (255.255.248.0)	2,048 host addresses
	11111111.11111111.11111100.00000000	/22 (255.255.252.0)	1,024 host addresses
	11111111.11111111.11111110.00000000	/23 (255.255.254.0)	512 host addresses
Class C	**11111111.11111111.11111111.00000000**	**/24 (255.255.255.0)**	**256 host addresses**
	11111111.11111111.11111111.10000000	/25 (255.255.255.128)	128 host addresses
	11111111.11111111.11111111.11000000	/26 (255.255.255.192)	64 host addresses
	11111111.11111111.11111111.11100000	/27 (255.255.255.224)	32 host addresses
	11111111.11111111.11111111.11110000	/28 (255.255.255.240)	16 host addresses
	11111111.11111111.11111111.11111000	/29 (255.255.255.248)	8 host addresses
	11111111.11111111.11111111.11111100	/30 (255.255.255.252)	4 host addresses
	11111111.11111111.11111111.11111110	/31 (255.255.255.254)	2 host addresses
	11111111.11111111.11111111.11111111	/32 (255.255.255.255)	"Host Route"

Figure 1-2 *Classless IPv4 Address Allocation*

NAT with Private Addresses

NAT along with private IPv4 addresses has arguably been the reason IPv4 has survived all these years—to the deferment, and perhaps detriment, of IPv6. Although NAT was meant to be a "short-term" solution to the IPv4 exhaustion problem, for more than 20 years, it has been a critical tool used on almost every user network.

The following private IPv4 addresses are defined by RFC 1918:

- 10.0.0.0–10.255.255.255 (10.0.0.0/8)
- 172.16.0.0–172.31.255.255 (172.16.0.0/12)
- 192.168.0.0–192.168.255.255 (192.168.0.0/16)

You will most likely recognize the 192.168.0.0 or 10.0.0.0 address used by many home routers that perform NAT.

NAT allows multiple hosts using private IPv4 addresses within their internal network to share one or more public IPv4 addresses when accessing the global Internet. A public IPv4 address is one that is routable in the global Internet. In most cases, a network

connecting to the Internet needs only one or a few public IPv4 addresses. Internally, the network uses private IPv4 address space, providing up to 16 million unique addresses. Typically a single public IPv4 address can be used to translate several hundred or even a few thousand internal private IPv4 addresses. NAT was initially defined in RFC 1631, *The IP Network Address Translator (NAT)*.

Note Public IPv4 addresses are addresses that are globally routed by entities that provide Internet connectivity.

The Internet Assigned Numbers Authority (IANA) allocated three ranges of addresses that are considered private IPv4 addresses, as specified in RFC 1918. These private network addresses can be assigned to devices in private networks but are not routable in the global Internet. A private IPv4 address must be translated to a public IPv4 address before it can be forwarded by an Internet-facing router.

Note Although this is by far the most common use, NAT is not utilized only for translation between private and public IPv4 addresses. It can be used between any pair of addresses.

Note RFC 6598, *IANA-Reserved IPv4 Prefix for Shared Address Space*, assigns another block of shared address space, 100.64.0.0/10. Shared address space is distinct from RFC 1918 private address space because it is intended for use on service provider networks. However, it can be used in a manner similar to an RFC 1918 private address space.

Port address translation (PAT) is a form of dynamic NAT that maps multiple IPv4 addresses to a single IPv4 address using different TCP/UDP port numbers. PAT is also known as network address port translation (NAPT), NAT overloading, single-address NAT, and port-level multiplexed NAT. With PAT, each computer on a private network can be translated to the same public IPv4 address but with a different source port number assigned. Whenever the term NAT is mentioned in this book, it also includes PAT.

Figure 1-3 shows an example of NAT. In this case, each of the devices on the home network will receive an RFC 1918 private address—for example, a 192.168.1.0/24 address. The user's home router will perform NAT, translating all Internet-bound packets to a single public IPv4 address.

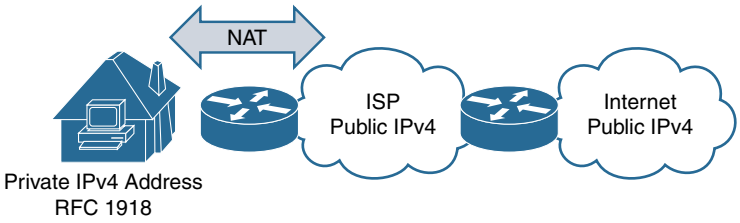


Figure 1-3 *Basic NAT for IPv4*

In many cases, the public IPv4 address is that of the router's Internet-facing interface. The home devices are usually clients, initiating communications to a server on the Internet with a public IPv4 address such as Facebook's or Amazon's. As the server responds to the client requests, the home router translates the public IPv4 address back to the client's original private IPv4 address.

Problems with NAT

In our need to conserve public IPv4 address space, we have tended to overlook some of the inherent problems with NAT. RFC 1631 even cautioned about negative effects of NAT: "It hides the identity of hosts. While this has the benefit of privacy, it is generally a negative effect." Although NAT was designed as a short-term solution, it became a rather permanent fixture in today's networks.

At the very least, NAT means that our routers, application gateways, firewalls, and other devices must perform extra processing to make NAT work, which also causes latency. The following are some of the major issues with NAT:

- **Checksum recalculations:** When carrying TCP segments, modifying the IPv4 header also means the IPv4 checksum must be recalculated.
- **ICMP manipulation:** Many ICMP messages, such as the Destination Unreachable message, embed in their payload the original IPv4 header that led to the ICMP message being generated. Because the original IPv4 address was translated by NAT, the NAT device must translate these addresses as well.
- **IPsec issues:** NAT invites complications when using Internet Protocol Security (IPsec) in transport mode. If IPsec AH (Authentication Header) is used, the NAT translation breaks the integrity check because the packet was modified during transport. NAT modifies the TCP/UDP checksum, which causes the integrity check to fail at the other end. Therefore, NAT cannot be used with IPsec in transport mode. Although it may work in tunnel mode, there can be issues in that case as well.
- **Breaking end-to-end reachability:** NAT causes problems with one of the key principles of the Internet: end-to-end reachability. For most users, this has not been an issue as the Internet has been comprised of mainly client/server networks. Generally, the Internet has been used by users and consumers who download content

such as web pages, email, and music. However, this is changing as the web continues to transition to more of a two-way medium, relying on video conferencing applications like Skype and smart home IoT applications that remotely manage the home. NAT makes accessing a device with a private IPv4 address difficult. Therefore, peer-to-peer, IoT, and many other types of services must provide an intermediary device, a kind of server with a public IPv4 address that both end devices can connect to—and that breaks pure end-to-end reachability. There are methods such as port forwarding to allow direct access to a device with a private IPv4 address, but they add another layer of complexity and potential problems.

- **Performance:** The process of having to translate addresses as packets leave and re-enter a network creates delay. This is especially a problem when NAT has to be performed more than once, such as when carrier-grade NAT (CGN) is used.

Many service providers use CGN, also called large-scale NAT (LSN), as another way to conserve IPv4 address space (see Figure 1-4). With CGN, instead of giving user networks their own public IPv4 addresses, a user's private IPv4 address is translated using NAT to another private IPv4 address to connect to its provider's network. The ISP typically uses an RFC 6598 address.

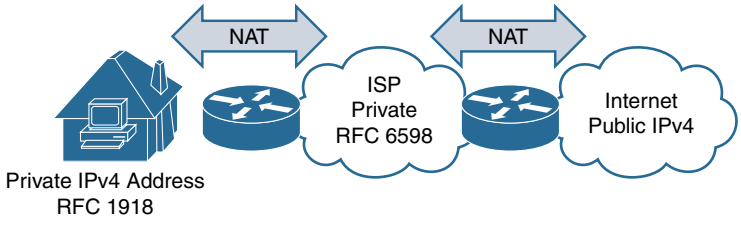


Figure 1-4 *Carrier-Grade NAT*

When the ISP forwards this packet to another provider (the public IPv4 Internet), NAT is performed once again, this time translating the RFC 6598 address to a public IPv4 address. The reverse process takes place for the returning packets. As you can imagine, this adds additional complexity and latency. For those familiar with port forwarding, CGN makes this even more difficult. (Port forwarding is the process of intercepting data traffic and modifying a combination of the IP address and TCP/UDP port number in order to redirect the data to a specific device.)

NAT is Not Security

Many people have argued and continue to argue the security merits of NAT. NAT has certain side effects that resemble security, but NAT itself is not a security feature. The IETF never intended NAT to be used for security. Many have also argued and demonstrated that NAT actually makes a network less secure and introduces its own security problems. RFC 1631 states that "NAT reduces the number of options for providing security" and makes debugging more difficult. Another excellent reference is RFC 2993, *Architectural Implications of NAT*.

However, we can't ignore the fact that one of the *perceived* security benefits of NAT is that it hides internal, private IPv4 addresses from the outside world. This makes accessing a private network address from the public Internet very difficult. Without getting too deep into this topic, it really isn't NAT that provides this apparent security. What provides the security is that NAT requires *state*. A NAT device must remember which private addresses and port numbers got mapped to which public addresses and port numbers. This statefulness means that any device performing NAT must also act as a stateful firewall.

I think we can just about all agree that what provides security for a network is a stateful firewall or other type of security appliance. After all, if NAT really did provide the security we deem necessary for our networks, we wouldn't have any issues with viruses, malware, ransomware, or any of the other problems plaguing our devices. Besides, most devices hiding behind NAT get infected by malicious email attachments and their users end up going to rogue websites, downloading infected files, falling for phishing schemes, and even using fake antivirus software.

For more information about the perceived security nature of NAT, you may want to check out these articles and this excellent and very funny video:

- **"Is NAT a Security Feature?" by Ivan Pepelnjak:** blog.ipspace.net/2011/12/is-natsecurity-%20feature.html
- **"IPv6 Security Myth #3—No IPv6 NAT Means Less Security" by Chris Grundemann:** www.internetsociety.org/deploy360/blog/2015/01/ipv6-security-myth-3-no-ipv6-nat-means-less-security/
- **"'Fanboy' Series—IPv6 and NATs" by Andrew Yourtchenko:** www.youtube.com/watch?v=v26BAlfWBm8

Is there NAT for IPv6? Yes, but it's not the same as NAT for IPv4. Most of the versions of NAT for IPv6 have to do with protocol translation between IPv6 and IPv4. NAT66 (described in a long-expired draft RFC; see tools.ietf.org/html/draft-mrw-behave-nat66-02) and NPTv6 (described in RFC 6296) are used for translation between two IPv6 addresses. NAT66 and NPTv6 are both designed to support address independence, not stateful translation between public and private IPv6 addresses. The various IPv6 NAT options are explained further in Chapter 17, "Deploying IPv6 in the Network."

NAT Example

Figure 1-5 shows an example of a network that uses NAT. Company XYZ has the public IPv4 address 209.165.200.248/29. This gives the company only six usable host addresses: 209.165.200.249 through 209.165.200.254. Company XYZ has hundreds of hosts, all of which might need access to the Internet at any given time. Because the company has only six public IPv4 addresses, it uses private IPv4 address space and NAT to meet its Internet access needs.

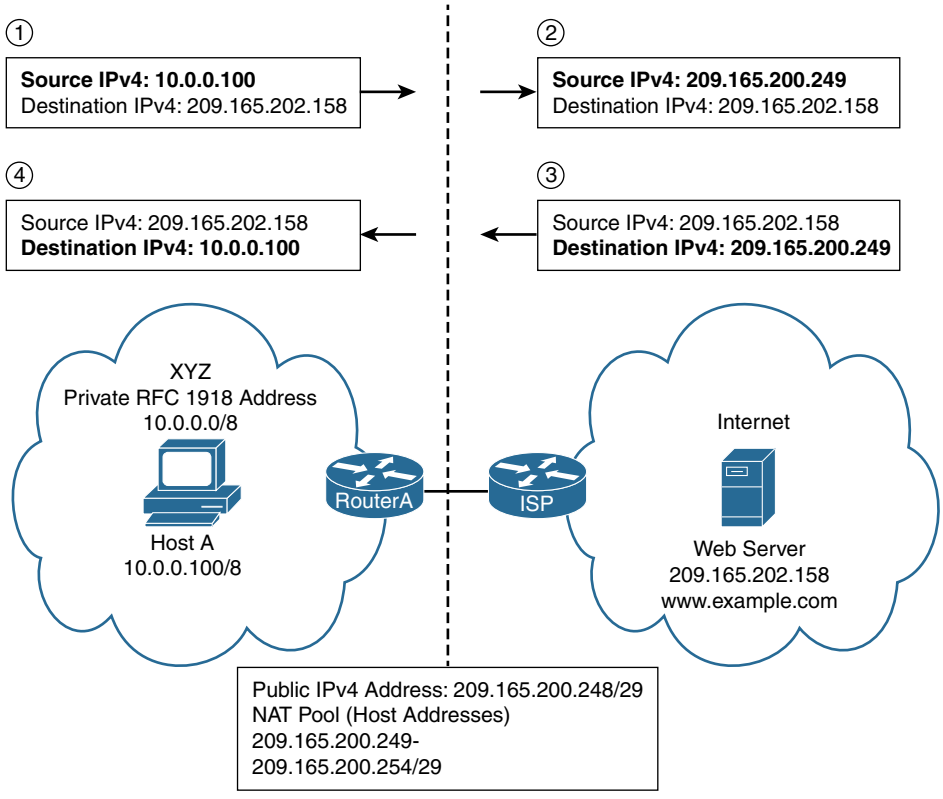


Figure 1-5 *Example of IPv4 NAT*

Company XYZ's network administrator has chosen to use the private network address 10.0.0.0/8. This provides more than enough host IPv4 addresses to meet the company's needs. RouterA is Company XYZ's router, which connects the private network to the Internet by forwarding packets to the ISP router. The steps in Figure 1-5 show how NAT translates a private IPv4 address to and from a public IPv4 address. For simplification, the port translations that are part of PAT have not been included.

Step 1. HostA on Company XYZ's network sends a packet with its source IPv4 address 10.0.0.100 and the destination IPv4 address 209.165.202.158 to the www.example.com web server.

Step 2. RouterA receives the packet and performs the NAT translation before forwarding the packet to the ISP router. RouterA replaces the packet's private source IPv4 address of 10.0.0.100 with the public address 209.165.200.249. The router also likely change the source TCP/UDP port number as part of NAT. (Again, PAT has not been included in this discussion to keep it simple.)

RouterA maintains a NAT table to keep track of all the addresses and port assignments that it modifies so that it can put the original address and port numbers back for the return traffic in step 4.

Step 3. The www.example.com web server returns the packet to the host with the source IPv4 address 209.165.202.158 and the destination IPv4 address 209.165.200.249.

Step 4. RouterA receives the return traffic for HostA. Using its NAT table, RouterA identifies the destination IPv4 address and destination port number, and it modifies these with the original address, 10.0.0.100, and the original port number.

What About IPv5?

Whatever happened to IPv5? No, the Internet doesn't have an issue with odd or prime numbers. In the late 1970s, a family of experimental protocols, known as Internet Stream Protocol (ST) and later ST2, was developed. Originally defined in IEN-119 (1979), ST was later revised in RFC 1190 and RFC 1819.

ST was an experimental resource reservation protocol intended to provide Quality of Service (QoS) for real-time multimedia applications such as video and voice. ST consisted of two protocols—ST (Internet Stream Protocol) and Stream Control Message Protocol (SCMP). Internet Stream Protocol version 2 (ST-II or ST2) was not designed as a replacement for IPv4. The idea was that a multimedia application would use both protocols—IPv4 for the transfer of traditional packets and ST2 for packets carrying real-time data.

Although it was never recognized as IPv5, when encapsulated in IPv4, ST uses IP Protocol Number 5 (RFC 1700). In other words, although it was never implemented, the designation "IP version 5" was already taken. Today's standard for resource reservation is the transport layer protocol Resource Reservation Protocol (RSVP), which can be used to provide receiver-initiated setup over IPv4. RSVP is described in RFC 2205.

The Fascinating History of IPv6

The story of how IPv6 came to be the protocol it is today is a fascinating one. Really! We might just assume that protocols are always developed with a specific logical process similar to the scientific method, with the final result being the one-and-only inevitable solution. Well, that's not always how it works, as is the case with IPv6.

The story of how IPv6 became the successor to IPv4 is a story that you might not have expected. It illustrates both the technical and human sides of protocol development—a process that at times brings intense disagreement on what technology or standard is best. The story behind IPv6 also shows how Internet governance went from being controlled mostly by individuals from American companies, universities, and research institutions to a more international collaboration.

Some Background

To get a bit of background on the development of IPv6, let's look at the world of the Internet back in 1990:

- In August 1990 the first web page was accessible.
- There were about 3 million users, and 73% of them were in the United States.
- There was no Amazon, no Google, no Yahoo, no Facebook. (There was not even Netscape, which you may not be old enough to remember.)
- The most popular Internet application was text-based email.

It's important to note that up to this point, the Internet was very US-centric. Not only were most of the users Americans, but the governing organizations such as the Internet Architecture Board (IAB; originally called the Internet Configuration Control Board) and the IETF consisted primarily of Americans working at US corporations, universities, and research institutions.

The Advanced Research Projects Agency Network (ARPANET), the precursor to today's Internet, was developed by the US Department of Defense. ARPANET came to life in 1969, interconnecting the University of California Los Angeles (UCLA), the Augmentation Research Center at Stanford Research Institute (SRI), the University of California Santa Barbara (UCSB), and the University of Utah. And for the next 20 years, the Internet was primarily controlled by Americans at US institutions.

Note To learn more about the creation of the Internet and those who were responsible for its development, I highly recommend the national bestselling book, *Where Wizards Stay Up Late: The Origins of the Internet*, by Katie Lyons.

Established in 1979 by Vint Cerf, the IAB is responsible for the direction of the Internet, Internet protocol architecture, and the ultimate responsibility for approving protocols. In 1990 the IAB consisted of 11 members (there are 13 today), mostly Americans with a core philosophy of "working code and rough consensus."

The IAB established the IETF in 1986. The IETF had no formal membership, all were volunteers—again mostly Americans. Its primary role was and still is to develop and propose standards (protocols) that make up the Internet protocol suite, TCP/IP.

In 1990 it was not a foregone conclusion that TCP/IP would be the Internet protocol suite of the future. The International Organization for Standardization (ISO) was ready with its own protocol suite, the Open Systems Interconnection (OSI) that it had been developing with the International Telecommunications Union (ITU). ISO, founded in 1947, in Geneva, Switzerland, is an international organization that promotes worldwide commercial and industrial standards. TCP/IP had become the de facto standard (which means it was the standard only because it was so widely used). But ISO still thought its OSI suite would ultimately become the officially endorsed standard of the Internet

(the de jure standard). It was just a matter of time before these two standards organizations would face off, and it would happen over IPv6.

Note ISO is not an acronym but refers to the Greek word *isos*, which means "equal."

IPv4 Address Exhaustion and the Need for More International Involvement

In 1990, both the IAB and the IETF began discussions about the shortage of IPv4 addresses. The sizes of Internet routing tables were increasing rapidly, and the number of Internet users was exploding. The IAB and the IETF agreed that it was time to begin designing and testing a new network layer protocol as the successor to IPv4. In addition to increasing the size of the address space, this was also a unique opportunity to fix the limitations of IPv4 and develop a protocol to ensure reliable growth and enhanced performance for the future.

Note Much of this process is well documented in RFC 1752, *Recommendation for IPng*.

During an IETF meeting in August 1990, it was estimated that current IPv4 address assignment rate would be depleted by 1994. This was even before the arrival of the World Wide Web, which was soon to revolutionize how we use the Internet. At the time, the Web was still under development by Tim Berners-Lee at CERN (French acronym for the European Organization for Nuclear Research in Switzerland). As we would soon see, this would bring a vast number of new Internet users that would greatly add to the problem IPv4 address depletion.

At this time, the IAB also had concerns about the Internet being so US-centric. Not only were the IAB, IETF, and other Internet governing bodies made up of mostly Americans, but the US had most of the IPv4 address space. At the time, IPv4 addresses were allocated by a single US entity, IANA, and the US had received the majority of IPv4 addresses.

Note In 1990, the process of IPv4 address allocation was delegated by IANA to SRI International's Networking Information Center in Marina Del Rey, California, which was funded by the US Department of Defense.

Vint Cerf, IAB chair, wanted to see more international participation. He advocated that IPv4 addresses should be allocated by international organizations. This is how the five RIRs were eventually created.

The battle for the Internet suite of protocols had two contenders: IETF's TCP/IP suite and ISO's OSI suite. TCP/IP was the dominant suite, with a working set of protocols. TCP/IP

also had the backing of the IETF, which documented all protocols and processes using RFCs (Request for Comments). OSI had limited deployment, and the management of its standards used a much more cumbersome and bureaucratic process (at least in the eyes of the IETF). OSI also had the backing of the US National Institute of Standards and Technology (NIST) and large US corporations like Digital Equipment Corporation (DEC).

And the winner is…. In the summer of 1992, the IAB had a meeting in Kobe, Japan. Without consulting with the IETF, as was customary, the IAB, in an effort to be more internationally inclusive, proposed OSI's Connectionless-mode Network Protocol (CLNP) as the replacement for IPv4. The CLNP proposal was called TCP and UDP with Bigger Addresses (TUBA). It was to replace IPv4 and its 32-bit address space with a variable-length, 160-bit CLNP address.

And the name of the new protocol? The protocol was to be named IPv7, erroneously skipping IPv6. What about the TCP/IP protocol suite? The plan was to use as much of the upper layers of TCP/IP as possible, which is why TUBA included "TCP and UDP" in its name.

When the IAB presented this proposal to the IETF, there was immense outrage and disbelief among the IETF members. First of all, they were furious that this was done without consulting them, not to mention discarding IPv4 in its entirety. The IETF also feared losing control over the standards development process to the more complex and bureaucratic ISO. The IETF's message was blunt and unified: Reject the IAB's mandate to replace IPv4 with CLNP.

A Call for Proposals

Vint Cerf and the IAB got the message. In 1993, IETF decided to open up the process with a call for proposals to replace IPv4. The announcement was made in RFC 1550, *IP: Next Generation (IPng) White Paper Solicitation*. This is why you might see some older documentation refer to IPv6 as IPng.

Note RIPng is Routing Information Protocol next generation, which is RIP for IPv6.

The Internet Engineering Steering Group (IESG), headed by Scott Bradner and Allison Mankin, would make the final decision. Three proposals were submitted:

- **Common Architecture for the Internet (CATNIP):** CATNIP proposed integrating IPv4, Internetwork Packet Exchange (IPX), and CLNP. IPX was part of the Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX) suite of protocols, used primarily on networks employing the Novell NetWare operating systems. CLNP is an OSI standard defined in ISO 8473 and is the equivalent of IPv4 for the OSI suite of protocols. CATNIP is defined in RFC 1707.

- **Simple Internet Protocol Plus (SIPP):** SIPP recommended increasing the IPv4 address size from 32 bits to 64 bits, along with making additional improvements to the IPv4 header for more efficient forwarding. SIPP is defined in RFC 1710.
- **TCP/UDP over CLNP-Addressed Networks (TUBA):** TUBA was aimed at minimizing the risk associated with migration to a new IP address by replacing IPv4 with CLNP and its address size of 20 bytes (160 bits). TCP, UDP, and the traditional TCP/IP applications would run on top of CLNP. TUBA is defined in RFCs 1347, 1526, and 1561.

The IETF formed the IP Next Generation working group in 1993. In July 1994, it was no surprise that the IETF announced its recommendation that the IP-based SIPP, written by Steve Deering, Paul Francis, and Bob Hinden, but with an address size of 128 bits, should replace IPv4.

A More IP Version of IPv6

In 1994, the IESG chose SIPP, but with a 128-bit IP address. The choice of SIPP with a 128-bit IP address meant that the IETF would continue to retain control of the standards development process.

In 1995, IETF published RFC 1883, *Internet Protocol, Version 6 (IPv6) Specification*, which later became obsolete and was replaced by RFC 2460 in 1998. As mentioned earlier, the name IPv5 was already taken, and the next available version was IPv6. So, in 2001, the IPng working group was renamed the IPv6 working group.

In 1996, the IETF formed an IPv6 test bed network known as *6bone*. The 6bone network started out using IPv6 over IPv4 tunneling/encapsulation, operating over the IPv4-only Internet to support IPv6 transport. Later it migrated to native links for IPv6. 6bone was phased out in 2006.

RIRs began allocating IPv6 addresses to their customers in 1999. Adoption was slow at first, but in 2007, RIRs began receiving significantly more requests for IPv6 address space. This was because RIRs supported a more widespread deployment of IPv6. Recently, with IANA running out of IPv4 addresses and now with four out of five RIRs depleted of IPv4 addresses, IPv6 adoption is seeing much more rapid growth.

In 2000, many vendors began adding IPv6 support to their mainstream products. Cisco Systems added IPv6 support with Cisco IOS Software Release 12.2(2)T. Linux vendors started supporting IPv6 in 2000, and Microsoft announced IPv6 support with Windows XP in 2001.

The IAB, IETF, RIRs, and other Internet governing bodies are much more international today than they were in 1990. The desire of the IAB to be a more inclusive collaboration of representatives from countries all over the world has become more of a reality.

Note Besides RFC 2460, there are dozens of other RFCs related to IPv6. An excellent list of these RFCs can be found at sites.google.com/site/ipv6center/ipv6-rfcs.

Note To learn more about the politics and personal side of protocol development, including IPv6, I highly recommend the book *Protocol Politics: The Globalization of Internet Governance (Information Revolution and Global Politics)*, by Laura DeNardis.

IPv6: More Than Just Longer Addresses

Many people who are new to IPv6 are surprised to learn that IPv6 is much more than just a larger source and destination IP address. Developers of IPv6 took this opportunity to not only improve IP but also many of the protocols and processes related to IP. This foresight has continued, with standards and informational RFCs that go well beyond the Internet of today. For example, the home of tomorrow may look more like an enterprise network of today. Perhaps a home will be connected to multiple ISPs, each providing differentiated services. A home network may include multiple subnets for personal use, business needs, entertainment systems, guest access, smart home devices, and so on. A good discussion of these home networking concepts can be found in RFC 7268, *IPv6 Home Networking Architecture Principles*.

So, why an entire book on IPv6? As mentioned earlier, IPv6 is much more than just a larger IP address. The following is just a sampling of some of the important features of IPv6 that are discussed in this book:

- An IPv6 address (global unicast address) includes a Subnet ID field that can be defined by the organization so there is no need to borrow bits from the host portion of the address to create subnets. This allows the organization to easily design and manage subnets for its own purposes, not for conserving address space as with IPv4.
- The IPv6 header has been significantly modified to include one new field and the removal of others.
- A new ICMPv6 Neighbor Discovery Protocol (NDP) has four new message types. ICMPv6 is used instead of ARP for resolving Layer 2 to Layer 3 addresses. NDP is also used by IPv6 routers to send messages suggesting how a device can dynamically receive its IPv6 addressing information.
- A non-routable IPv6 link-local address has significant uses in IPv6, including during neighbor discovery.
- A new solicited-node multicast address is used instead of a broadcast address to make neighbor discovery more efficient.

- IPv6 includes Stateless Address Autoconfiguration (SLAAC), which allows devices to obtain unique and globally routable addresses without the services of a DHCPv6 server.
- Stateless and stateful DHCPv6 provides several options for a device to obtain some or all of its addressing information.
- There is a neighbor cache, which is similar to the ARP cache for IPv4 but with multiple states.
- Devices can have multiple IPv6 addresses on the same subnet or different subnets to allow for greater privacy and additional features. These options can be managed at the host or network level.

And there is much more as well. Don't worry about understanding these concepts right now. This book explains them and many more of the nuances of IPv6 in a step-by-step, simple manner.

IPv6 Myths

There are several misperceptions or myths regarding IPv6. IPv6 has been around quite some time, initially introduced in 1995 with RFC 1883 and later obsoleted with RFC 2460 in 1998. Over the years, as IPv6 evolved and as people discussed the merits of the new protocol, certain misconceptions ensued. Let's take a look at some of them:

- **IPv6 is more secure than IPv4.** Some may think that IPv6 is more secure than IPv4 because it uses IPsec. IPsec is a security protocol suite for authenticating and encrypting IP packets that was initially developed for IPv4. IPsec does not make IPv6 any more secure than IPv4. IPsec can't stop all attacks against IPv6, ICMPv6, or any of the other related protocols. The fact is that neither IPv4 nor IPv6 is any more secure than the other. IPsec was originally required on all implementations of IPv6, but RFC 6434 changed it to be only a recommendation.
- **IPv6 is less secure than IPv4.** Some believe that IPv6 is less secure than IPv4 because it doesn't use NAT. As we have already discussed, NAT is not security. Although NAT provides a side effect of hiding your private IPv4 addresses from the public Internet, this does not equate to security.
- **IPv6 will replace IPv4.** There is no switchover date for going from IPv4 to IPv6. IPv4 and IPv6 will coexist for the foreseeable future. So don't worry; you still get to use all of your IPv4 knowledge along with your newfound IPv6 knowledge.
- **IPv6 isn't necessary.** Some believe that because they have plenty of IPv4 addresses, they don't need to understand or implement IPv6. As mentioned at the beginning of this chapter, for both business and security reasons, it is important to begin to implement IPv6. Ignoring IPv6 could potentially isolate your online services from some consumers, and it could also make your network vulnerable to certain types of IPv6 MITM and DoS attacks.

- **IPv6 is too complex.** IPv6 and related protocols such as ICMPv6 may be new and different, but they are not any more complex than IPv4. In many ways, IPv6 is actually much easier. Thanks to the Subnet ID field in the IPv6 header, subnetting IPv6 is simple. If you can count in hexadecimal, then you can subnet in IPv6. In addition, there is no fragmentation of IPv6 packets by routers, so these fields don't exist in the IPv6 header. IPv4's fragmentation and reassembly can be a complicated process to understand.
- **IPv6 improves QoS.** Another common misconception is that IPv6 provides better QoS. Both IPv4 and IPv6 use the same Differentiated Serves and Integrated Services fields for QoS. IPv6 does provide a new Flow Label field that has the potential to improve the efficiency of flows in an IPv6 network. Although many operating systems set the Flow Label field for IPv6 packets, currently there aren't a lot of implementations that look at it. The Flow Label field is discussed in Chapter 3, "Comparing IPv4 and IPv6."

Transitioning to IPv6

> *"The IPv6 protocol standard is critical for supporting the Internet's continued development. Network operators, content providers, software and hardware developers, and enterprises, among others, need to implement IPv6 in order to ensure efficiency, global connectivity, and long-term growth of the Internet."*
>
> —Internet Society (ISOC) Policy Brief: Adoption of IPv6, April 2016

When will the world transition to IPv6? Well, there is no specific date to switch from IPv4 to IPv6. IPv4 and IPv6 will most likely coexist for many years. The transition to IPv6 is in progress and will continue to gain momentum for the reasons we have discussed in this chapter. In June 2011, ISOC held its first World IPv6 Day. The goal was for service and content providers to test their IPv6 implementations. Since then, IPv6 adoption by service and content providers has continued to increase, as has the number of networks enabled for IPv6.

IPv6 has a variety of tools to help with the transition from IPv4 to IPv6, including tunneling and NAT. Tunneling encapsulates an IPv6 packet into an IPv4 packet so that it can be delivered over IPv4-only networks. NAT provides a mechanism to translate an IPv4 address to an IPv6 address or an IPv6 address to an IPv4 address. However, these are just temporary transition tools and in most cases no longer needed.

Whether with a home network or a large enterprise network, the goal should be native IPv6 connectivity. Native IPv6 means no translation techniques or tunneling protocols are needed. Large service providers such as Comcast, AT&T, Time Warner, Verizon, Sky Broadband, Deutsche Telekom, and many others are offering native IPv6. If your ISP is not providing native IPv6 connectivity, ask when it plans to deliver IPv6.

As we mentioned earlier, the killer application for IPv6 is the Internet itself. As ISOC, IETF, and the RIRs have been saying for years, the Internet cannot sustain itself with IPv4. It can't continue to grow and evolve without more addresses, and the only long-term solution is IPv6.

Now is the best time for IT departments to begin preparation for IPv6—before there is a pressing need or an immediate requirement. IT organizations need time to familiarize themselves with IPv6, educate their workforces, and test their IPv6 implementations. Preparation should focus on all aspects of the IT department, such as network operations, security, application developers, content developers, and the data center, just to name a few.

Here are some of the things that can be done to start the migration to IPv6:

- **Include IPv6 in your IT strategy:** IPv6 needs to be a part of your IT strategic planning for both the short term and the long term. IPv6 needs to be integrated into your IT planning processes, status meetings, training, and purchasing plans.
- **Provide training:** The optimum time to begin training about IPv6 or at least becoming familiar with it is before you have a requirement to implement it. Education/training is the best way for IT staff to lose their fear and intimidation of IPv6. Training can also include non-IT staff, such as salespeople. A friend of mine who's been a network engineer for over 25 years told me he was hoping to have retired by the time he had to learn IPv6. As you read this book, you will see that IPv6 is not as complicated as you might think; in many ways, it's easier to deal with than IPv4. For example, subnetting doesn't require any bitmapping, so IPv6 subnets can be easier to configure and identify.
- **Inventory equipment, applications, and services:** Develop a list of devices on your network (or use an existing inventory list) and check whether these devices currently support IPv6 or what upgrade/replacement is necessary. Include the following:
  - Clients and servers
  - Routers and multilayer switches
  - Layer 2 switches
  - Firewalls and other security devices
  - Printers, webcams, phones, and any other devices that have IP addresses on your network
  - Applications, operating systems, and software services
- **Keep an eye on purchasing:** Your purchasing plan should confirm that any new equipment purchased is IPv6-compatible. Ask vendors whether their hardware/software supports IPv6. If it doesn't, ask them what their product implementation plan is for including IPv6 support.
- **Create a test lab:** Creating a lab that is off the production network is a great way to learn and play with IPv6. Then, as you become more comfortable with IPv6, you can

introduce it into your network as you continue to run IPv4. The lab should simulate the operational network in all aspects. It should come as close to the live network environment as possible—with the same routers, switches, firewalls, load balancers, applications, end systems, and so on.

Summary

After reading this chapter, you should have a better understanding of the limitations of IPv4 and why it is important to begin transitioning to IPv6.

Here are some of the key items discussed in this chapter:

- How we use the Internet today is much different than how it was used when IPv4 was developed. Today we have more users, more devices, and new demands. We have moved from just an *Internet of computers* to also an Internet of things.
- Although no one knows exactly when, we will eventually run out of IPv4's 4.29 billion addresses, but the fact is that the Internet is in the final stages of public IPv4 address availability.
- IPv6, with its 128-bit address scheme, provides more than enough globally unique addresses to support the growth of the Internet.
- IPv4 and IPv6 will coexist for the foreseeable future. IPv6 includes tools and migration strategies that allow both protocols to coexist.
- The combination of CIDR, NAT, and private addressing has helped slow the depletion of IPv4 address space. However, NAT introduces complexity, latency, and other problems.
- One of the perceived security benefits of NAT is that it provides address hiding. But NAT is not considered security. Because NAT requires state, a NAT device must act as a stateful firewall.
- In addition to providing a larger address space, IPv6 offers additional enhancements, such as ICMPv6 Neighbor Discovery Protocol (NDP) for Layer 3 to Layer 2 address resolution, and Stateless Address Autoconfiguration (SLAAC) that allows a device to create an IPv6 address without the services of DHCP.
- Now is the best time for IT departments to begin familiarizing themselves with IPv6.
- The Internet is the killer application for IPv6. IPv6 is the successor to IPv4 that will ensure the growth of the Internet.

Review Questions

1. What are two reasons network operators need to begin to transition to IPv6?
2. What temporary measures did the IAB and IETF implement when they recognized the impending shortage of IPv4 addresses?

3. List two problems with NAT.
4. Is NAT considered security? Why or why not?
5. Why isn't IPv5 the replacement for IPv4?
6. In 1992, what protocol did the IAB recommend as the replacement for IPv4?
7. In 1994, what protocol did the IESG recommend and the IAB select as the replacement for IPv4?
8. List three myths regarding IPv6.

References

Endnotes

1. Chetan Sharma Consulting, *US Wireless Market Update Q1 2016*, www.chetansharma.com/usmarketupdateq12016.htm.
2. *World IPv6 Launch Measurements*, www.worldipv6launch.org/measurements/.
3. *Case Study: T-Mobile US Goes IPv6-Only Using 464XLAT*, www.internetsociety.org/deploy360/resources/case-study-t-mobile-us-goes-ipv6-only-using-464xlat/.
4. APNIC, *IPv6 Performance—Revisited*, blog.apnic.net/2016/08/22/ipv6-performance-revisited/.
5. Internet Society, *Facebook News Feeds Load 20-40% Faster over IPv6*, www.internetsociety.org/deploy360/blog/2015/04/facebook-news-feeds-load-20-40-faster-over-ipv6/.
6. GlobalWebIndex, *Digital Consumers Own 3.64 Connected Devices*, www.globalwebindex.net/blog/digital-consumers-own-3.64-connected-devices.
7. *IPv4 Address Report*, www.potaroo.net/tools/ipv4.
8. *APRINIC IPv6 Programme*, afrinic.net/services/ipv6-programme.

RFCs

RFC 760, *DoD Standard, Internet Protocol*, USC, IETF, www.ietf.org/rfc/rfc760.txt, January 1980.

RFC 791, *Internet Protocol, DARPA Internet Program Protocol Specification*, USC, www.ietf.org/rfc/rfc791.txt, September 1981.

RFC 1190, *Experimental Internet Stream Protocol, Version 2 (ST-II)*, CIP Working Group, www.ietf.org/rfc/rfc1190.txt, October 1990.

RFC 1338, *Supernetting: An Address Assignment and Aggregation Strategy*, V. Fuller, BARRNet, www.ietf.org/rfc/rfc1338, June 1992.

RFC 1347, *TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing*, Ross Callon, DEC, www.ietf.org/rfc/rfc1347.txt, June 1992.

RFC 1519, *Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy*, V. Fuller, BARRNet, www.ietf.org/rfc/rfc1519.txt, September 1993.

RFC 1526, *Assignment of System Identifiers for TUBA/CLNP Hosts*, D. Piscitello, Bellcore, www.ietf.org/rfc/rfc1526.txt, September 1993.

RFC 1550, *IP: Next Generation (IPng) White Paper Solicitation*, S. Bradner, Harvard University, www.ietf.org/rfc/rfc1550.txt, December 1993.

RFC 1561, *Use of ISO CLNP in TUBA Environments*, D. Piscitello, Core Competence, www.ietf.org/rfc/rfc1561.txt, December 1993.

RFC 1700, *Assigned Numbers*, J. Reynolds, ISI, IETF, www.ietf.org/rfc/rfc1700.txt, October 1994.

RFC 1707, *CATNIP: Common Architecture for the Internet*, M. McGovern, Sunspot Graphics, www.ietf.org/rfc/rfc1707.txt, October 1994.

RFC 1710, *Simple Internet Protocol Plus White Paper*, R. Hinden, Sun Microsystems, www.ietf.org/rfc/rfc1710.txt, October 1994.

RFC 1752, *The Recommendation for the IP Next Generation Protocol*, S. Bradner, Harvard University, www.ietf.org/rfc/rfc1752.txt, January 1995.

RFC 1819, *Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+*, ST2 Working Group, www.ietf.org/rfc/rfc1819.txt, August 1995.

RFC 1883, *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, Xerox PARC, www.ietf.org/rfc/rfc1883.txt, December 1995.

RFC 1918, *Address Allocation for Private Internets*, Y. Rekhter, Cisco Systems, IETF, www.ietf.org/rfc/rfc1918.txt, February 1996.

RFC 2205, *Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification*, R. Braden, ISI, www.ietf.org/rfc/rfc2205.txt, September 1997.

RFC 2235, *Hobbes' Internet Timeline*, R. Zakon, MITRE, www.ietf.org/rfc/rfc2235.txt, November 1997.

RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, Cisco Systems, www.ietf.org/rfc/rfc2460.txt, December 1998.

RFC 2993, *Architectural Implications of NAT*, T. Hain, Microsoft, www.ietf.org/rfc/rfc2993.txt, November 2000.

RFC 3330, *Special-Use IPv4 Addresses*, IANA, www.ietf.org/rfc/rfc3330.txt, September 2002.

RFC 3927, *Dynamic Configuration of IPv4 Link-Local Addresses*, S. Cheshire, Apple Computer, www.ietf.org/rfc/rfc3927.txt, May 2005.

Websites

Google's list of IPv6 RFCs, sites.google.com/site/ipv6center/ipv6-rfcs

Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020 White Paper, www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

Internet World Stats, www.internetworldstats.com/stats.htm

North American Network Operators' Group (NANOG) IPv6 presentations, Search for IPv6, www.nanog.org/archives/presentations

This page intentionally left blank

Chapter 2

IPv6 Primer

"It ain't an easy job but when you bring a herd into town and you ain't lost one of them, ain't a feeling like it in the world."

—Cat Herders, Electronic Data Systems (EDS) commercial, aired during Super Bowl XXXIV

Learning IPv6 is a little like herding cats. Where to start? There are a lot of new concepts, and wherever you start, it always seems like it might have been better to have started somewhere else. This is because each topic relies on some previous knowledge of some of the other topics.

This chapter provides a brief introduction to concepts, protocols, and terminology that will give you the information you need to help you better understand the rest of the book. This chapter also gives you a better idea of some of the changes to IPv6 and related protocols such as ICMPv6 (Internet Control Message Protocol version 6).

Except for the first topic on the hexadecimal number system, everything else discussed in this chapter will be re-introduced and discussed more thoroughly in later chapters. So don't worry about completely understanding any of these IPv6 protocols and concepts for now. After reading this chapter, you will probably have more questions. But rest assured: They will be addressed later in this book.

This chapter introduces the following topics:

- **Hexadecimal number system:** IPv6 uses hexadecimal numbers to represent IPv6 addresses. This section is intended for those who are new to the hexadecimal number system or need a little review.
- **IPv6 address types:** IPv6 addresses include new address types and changes to some of the address types used in IPv4. This chapter looks at four of the most common address types: global unicast, link-local unicast, unspecified unicast, and

solicited-node multicast. Understanding these types will help you better understand many of the protocols and message types that use these different types of addresses, such as ICMPv6.

- **Address terminology:** IPv6 uses some terms that may be unfamiliar, such as *prefix* and *prefix length*. This chapter explains some of these basic terms that are used throughout the book.
- **ICMPv6 Neighbor Discovery Protocol (NDP):** ICMPv6 is similar to ICMP for IPv4. However, ICMPv6 has been extended, via ICMPv6 NDP, to support additional functions. ICMPv6 has four new message types: Neighbor Solicitation (NS), Neighbor Advertisement (NA), Router Solicitation (RS), and Router Advertisement (RA). These messages are used for address resolution (similar to Address Resolution Protocol [ARP] in IPv4) and dynamic address allocation. Getting a brief look at these messages now will help you better understand IPv6 address types and dynamic address allocation.
- **Dynamic address allocation:** IPv6 devices can receive their addressing information without the services of a Dynamic Host Configuration Protocol version 6 (DHCPv6) server. In this chapter we will introduce the three methods sent by a router's ICMPv6 Router Advertisement message for dynamic address allocation.

Again, everything we introduce in this chapter is also discussed in much more detail in later chapters.

Hexadecimal Number System

An IPv6 address is 128 bits in length, and, as we will see, hexadecimal is the ideal number system for representing long strings of bits. An IPv6 address with long strings of hexadecimal digits may look intimidating at first, but don't worry. Chapter 4, "IPv6 Address Representation and Address Types," looks at how IPv6 addresses are represented. You will see that in most cases, hexadecimal numbers make it easier to determine the different parts of an address.

If you understand the decimal, or base 10, number system, you can understand any number system, including the hexadecimal, or base 16, number system. You may already be familiar with binary, or base 2, but even if you're not, you will still be able to understand base 16. The same general rules apply to all number systems.

When looking at integer-based number systems, there are three general rules:

- **Rule #1:** Base *n* number systems have *n* number of digits:
  - The base 10 (decimal) number system has 10 digits.
  - The base 2 (binary) number system has 2 digits.
  - The base 16 (hexadecimal) number system has 16 digits.

- **Rule #2:** All number systems begin with 0.

  Combining Rule #1 and Rule #2, we get:

  - Base 10 has 10 digits, starting with 0: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Base 2 has 2 digits, starting with 0: 0, 1
  - Base 16 has 16 digits, starting with 0: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- **Rule #3:** The rightmost column, or least significant digit, is always the column of 1s (ones). Each preceding column is *n* times the previous column (where *n* is the base *n* number system). Using base 10 as an example, the first column is the column of 1s; the next column is 10 times the 1s column, or a column of 10s; the next column is 10 times the 10s column, or a column of 100s; and so on. Understanding this makes it very easy to convert other number systems to base 10 (see Table 2-1).

Table 2-1 *Number Systems*

Base *n* Number System	n^3	n^2	n^1	n^0
Base 10	1,000	100	10	1
Base 2	8	4	2	1
Base 16	4,096	256	16	1

- As you can see in Table 2-1, the columns in the three number systems include:
  - **Base 10:** 10,000s, 1000s, 100s, 10s, 1s
  - **Base 2:** 128s, 64s, 32s, 16s, 8s, 4s, 2s, 1s
  - **Base 16:** 4,096s, 256s, 16s, 1s

When you understand these three rules, you're ready to examine the hexadecimal number system more closely. The hexadecimal number system, base 16, has 16 digits, beginning with 0. Table 2-2 shows these 16 digits and their equivalents in decimal and binary.

Table 2-2 *Decimal, Hexadecimal, and Binary*

Decimal (Base 10)	Hexadecimal (Base 16)	Binary (Base 2)
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Let's apply the three rules to the hexadecimal number system:

- **Rule #1:** The hexadecimal number system has 16 digits.
- **Rule #2:** Table 2-2 illustrates the 16 hexadecimal digits and their decimal and binary equivalents starting with 0. Notice that you needed unique alphanumeric digits, A through F, to represent the decimal values 10 through 15.
- **Rule #3:** For representing IPv6 addresses in hexadecimal, you only need to use the rightmost column, or the column of 1s.

Using Table 2-2, it is easy to convert from one number system to any of the others. With one known value, you can easily determine the values for the other two number systems. The following are three examples, in which subscripts are used to differentiate between the number systems:

- If your known value is 14_{10}, you can find that it equates to E_{16} and 1110_2.
- If your known value is A_{16}, you can find that it equates to 10_{10} and 1010_2.
- If your known value is 0010_2, you can find that it equates to 2_{10} and 2_{16}.

Why use hexadecimal numbers to represent IPv6 addresses? Hexadecimal is a natural fit for IPv6 because any 4 bits (half of a byte, or half of an octet) can be represented as a single hexadecimal digit. In other words, there are 16 unique combinations of 4 bits, and there are also 16 digits in a hexadecimal number system, so it is a perfect match. Because one hexadecimal digit can represent 4 bits, two hexadecimal digits can represent a single byte, or octet. For this reason, hexadecimal is commonly used in computer science, computer networking, and other areas of computer technology.

Note 4 bits is half a byte or half an octet, also known as a *nibble*. You will sometimes see alternative spellings such as *nybble* and *nyble*.

IPv6 Address Types

In Chapter 4 we will discuss the hexadecimal format of an IPv6 address and examine in detail all the different IPv6 address types, including the ones we will discuss here. For now, you just need a brief introduction to four common address types that are used in many of the protocols discussed throughout this book.

Global Unicast Address (GUA)

An IPv6 global unicast address (GUA) is a globally unique and routable IPv6 address. It is equivalent to a public IPv4 address. A GUA begins with either a hexadecimal 2 or 3. A GUA can be either a source or destination IPv6 address.

The following is an example of a global unicast address:

```
2001:db8:cafe:1::100
```

Don't worry about the format of this address for now. We will explain it in Chapter 4.

Link-Local Unicast Address

A link-local address is a unicast address that is local only on that link. The term *link* refers to a logical network segment or subnet. Link-local addresses are limited to the particular link and are not routable beyond the local subnet.

An IPv6 device doesn't have to have a global unicast address but it must have a link-local address. In other words, any device that is IPv6-enabled requires a link-local address. The device must be capable of self-generating the link-local address on the IPv6-enabled interface.

Link-local addresses commonly begin with fe80, as in this example:

```
fe80::a299:9bff:fe18:50d1
```

Link-local addresses are typically created automatically by the host operating system, which is why you see these addresses already configured on devices with

Windows, Mac OS, and Linux operating systems. A link-local address can be either a source or destination IPv6 address.

Figure 2-1 illustrates the difference between an IPv6 global unicast address and a link-local address.

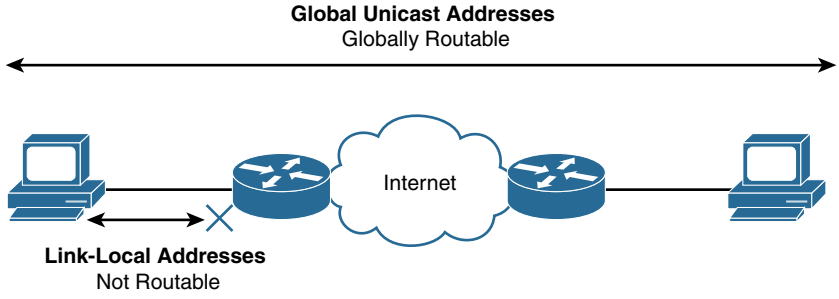


Figure 2-1 *Comparison Between IPv6 Global Unicast Address Addresses and Link-Local Addresses*

Unspecified Address

An IPv6 unspecified address is an all-0s address that indicates either the absence or anonymity of an IPv6 address. Unspecified addresses are used only as source addresses and never forwarded by an IPv6 router.

Solicited-Node Multicast Address

A solicited-node multicast address is a special multicast address that is used by some IPv6 protocols in lieu of the broadcast address that is used in IPv4. When an IPv4 broadcast address is encapsulated in an Ethernet frame, Ethernet uses a broadcast address for the destination MAC address. All devices on the Ethernet network (IPv4 subnet) will receive and process the frame. This means the Ethernet NIC must accept the frame and pass it up to the appropriate Layer 2 or Layer 3 protocol for processing. Some protocols, such as ARP, are not encapsulated in an IPv4 packet but the message may still be sent using an Ethernet broadcast address.

Note IPv6 does not have a broadcast address but does include an all-IPv6 devices multicast address. Chapter 4 discusses multicast address types.

This can be an inefficient process at times, such as in an ARP Request (see Figure 2-2). The ARP Request from PC1 is only trying to communicate with PC3. PC1 knows the IPv4 address of PC3 but not its Ethernet MAC address. However, the ARP Request is

encapsulated in an Ethernet frame that uses a Layer 2 broadcast, which means every device on the network must process the Ethernet frame and pass it to its ARP process to determine if it is the target of the ARP Request.

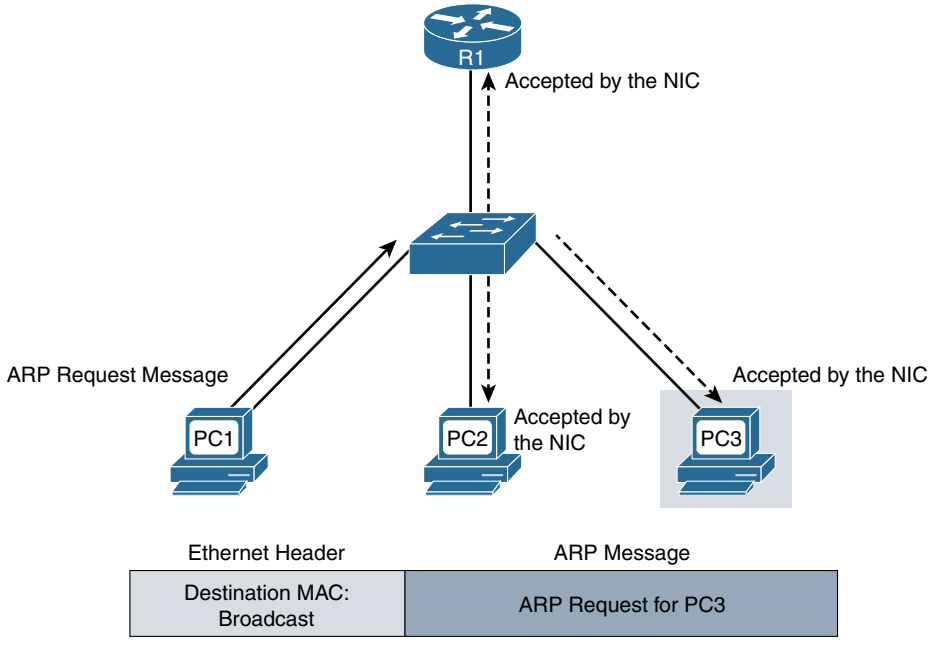


Figure 2-2 *ARP Request in IPv4*

Instead of ARP, IPv6 uses ICMPv6 Neighbor Discovery Protocol (NDP). As shown in Figure 2-3, PC1 sends a Neighbor Solicitation message, which is the equivalent of an ARP Request message in IPv4. PC1 knows the IPv6 address of PC3 but not its Ethernet MAC address.

The Neighbor Solicitation message, as shown at the bottom of Figure 2-3, is encapsulated in an IPv6 header. The destination IPv6 address is a solicited-node multicast address, which is mapped to a special Ethernet multicast address. The Ethernet multicast address is used as the destination MAC address in the Ethernet header.

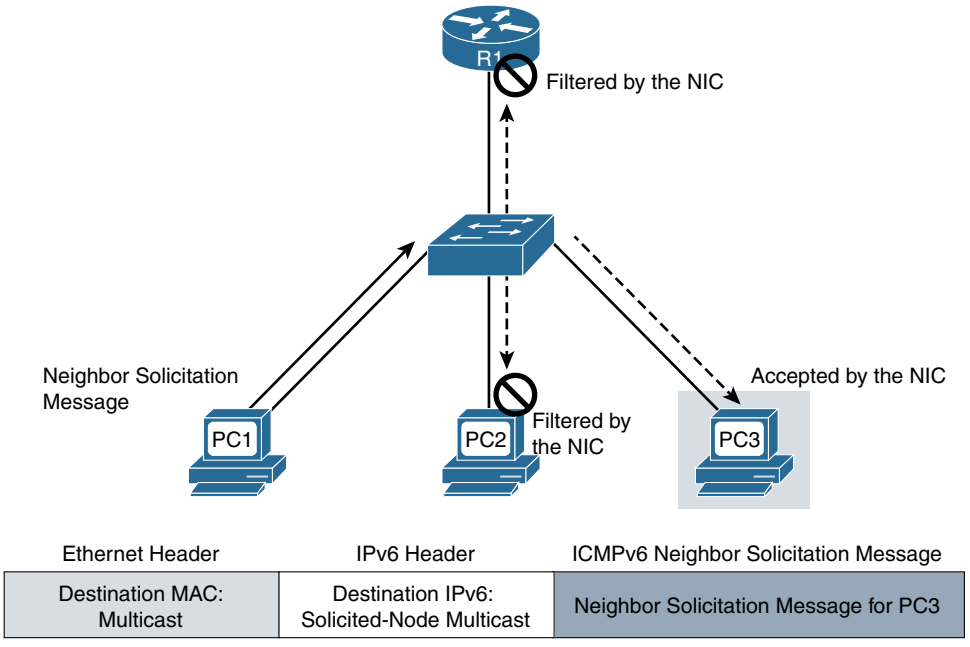


Figure 2-3 *Neighbor Solicitation Message in IPv6*

Don't worry about the details for now. All this will be explained in later chapters. At this point, it is just important to know that solicited-node multicast address is more efficient than an Ethernet broadcast.

As each device receives the Neighbor Solicitation message, its NIC examines the multicast destination MAC address and determines whether it is the target for this Ethernet frame.

In the Figure 2-3 example, the NICs of both PC2 and R1 examine the destination MAC address (multicast address) and realize that they are not the destination. The NICs of PC2 and R1 therefore discard the frame without passing it up to a higher-layer protocol.

However, when PC3's NIC examines the destination MAC address (a multicast address), it realizes that this message is *most likely* for this device, so it copies in the rest of the frame and passes the payload to the IPv6 process for further processing. (Here I say *most likely* because, as you will learn later, it is possible that more than one NIC will have the same multicast address.)

Like all other multicast addresses, the solicited-node multicast address can only be a destination address. It is also never routed off the link or subnet. Solicited-node multicast addresses begin with ff02:0:0:0:0:1:ff (kind of ugly, I know).

Solicited-node multicast addresses can be confusing and a little intimidating. But don't worry, you will understand completely after reading Chapter 7, "Multicast Addresses." For now, just know that this type of address can be used in lieu of an IPv4 broadcast address to make processing more efficient.

Address Terminology

IPv6 uses terminology that may be unfamiliar to you. Some of the terms are also used in IPv4. The following are a few terms you need to know that are used throughout this chapter and the rest of the book:

- **Prefix:** The prefix is the network portion of an IPv6 address. In an IPv4 address, we sometimes call this the network portion of the address, or the network prefix.
- **Prefix length:** The prefix length is the number of most-significant or leftmost bits that define the prefix, the network portion of the address. This is equivalent to the subnet mask in IPv4. IPv6 addresses are 128 bits, so the prefix length can be /0 to /128.
- **Interface ID:** The Interface ID is equivalent to the host portion of an IPv4 address. IPv6 uses the term *Interface ID* because any type of device can have an IP address, not just a host computer. A device with an IPv6 interface may range anywhere from a common server or client computer to an espresso machine or biomedical sensor. The term *interface* is used because an IP address (IPv4 or IPv6) is assigned to an interface, and a device may have multiple interfaces.
- **Node or device:** An IPv6 node or device is anything that can have an IPv6 address, including traditional devices such as computers and printers, along with other types of devices such as webcams, embedded devices, and Internet of Things (IoT) devices. The terms *node* and *device* are used interchangeably in this book.

Figure 2-4 illustrates the prefix, prefix length, and Interface ID of an IPv6 global unicast address. In this example, the prefix length /64 means the prefix is 64 bits, which leaves another 64 bits for the Interface ID.

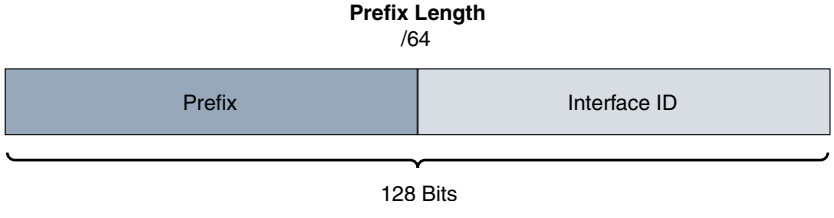


Figure 2-4 *Example of an IPv6 Global Unicast Address*

ICMPv6 Neighbor Discovery Protocol (NDP)

ICMPv6 Neighbor Discovery Protocol (NDP) adds new functionality for ICMPv6. NDP is used for on-link (same subnet) device discovery and messaging.

NDP includes five message types: Router Solicitation, Router Advertisement, Neighbor Solicitation, Neighbor Advertisement, and Redirect messages. The first four messages are new with ICMPv6. The Redirect message is also part of ICMPv4 but contains additional functionality.

All these messages are discussed in various contexts in later chapters.

Neighbor Solicitation (NS) and Neighbor Advertisement (NA) Messages

The Neighbor Solicitation and Neighbor Advertisement messages are used for messaging between any two devices on the same link (subnet). For example, these messages are used for address resolution and are the equivalent of ARP for IPv4. As shown in Figure 2-5, the Neighbor Solicitation messages and the Neighbor Advertisement messages are comparable to an ARP Request and an ARP Reply respectively. We will examine NS and NA messages and their uses in more detail later in this book.

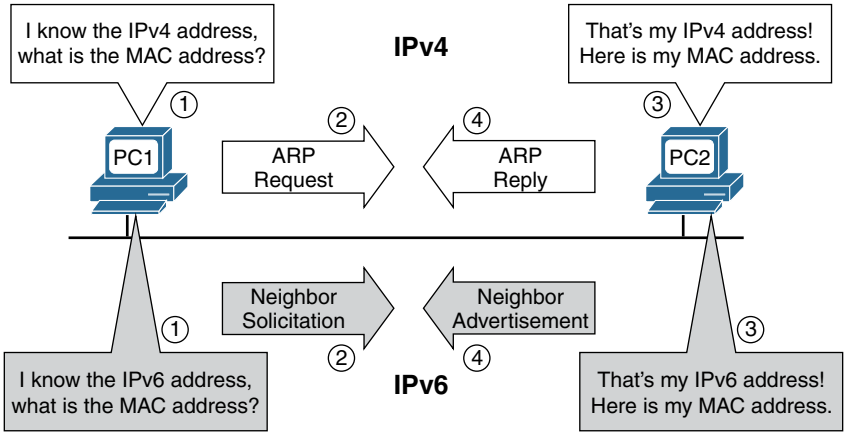


Figure 2-5 *Comparison of IPv4 ARP Request and ARP Reply to IPv6 NS and NA Messages*

Router Solicitation (RS) and Router Advertisement (RA) Messages

The Router Solicitation and Router Advertisement messages are used for messaging between a device and a router on the same link (subnet).

As discussed further in the next section, the Router Advertisement message is sent by a router as a suggestion to devices about how to dynamically obtain their IPv6 addressing information. The Router Solicitation message is sent by a device to request a Router Advertisement message from the router, as shown in Figure 2-6. The RA message is discussed further in the next section and in greater detail in later chapters.

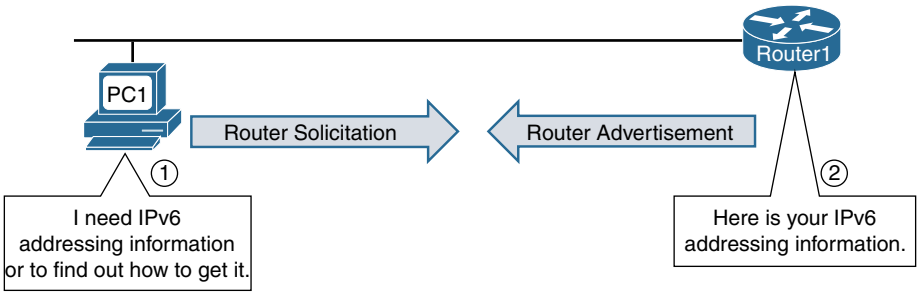


Figure 2-6 *ICMPv6 Router Solicitation and Router Advertisement Messages*

Dynamic Address Allocation

In IPv4, devices have two ways to get IPv4 addressing information, which includes an IPv4 address, subnet mask, default gateway address, domain name, and Domain Name Service (DNS) server address:

- Static or manual configuration
- Dynamically from a DHCPv4 server

As with IPv4, IPv6 addresses can be statically assigned. However, when it comes to dynamic addressing, IPv6 has a different approach. IPv6 uses the ICMPv6 Router Advertisement message to suggest to devices how to obtain their IPv6 addressing information. An IPv6 router sends a Router Advertisement message periodically (every 200 seconds in Cisco IOS) or when it receives a Router Solicitation request from a device. The RA message is typically sent to the all-IPv6 devices multicast address (ff02::1), so every IPv6 device on the link (network) receives it. (It can also be sent as a unicast message.) Other routers do not forward RA messages.

The Router Advertisement message includes addressing information for IPv6 devices that includes the following:

- The network prefix and prefix length, along with other information about the link (subnet).
- The address of the default gateway. (This is a link-local address of the router's egress interface, the source IPv6 address of the RA message.)
- Three flags that are used to suggest to a device how to obtain its IPv6 addressing information. These flags are the Autonomous Address Configuration Flag (A flag), the Other Configuration Flag (O flag), and the Managed Address Configuration Flag (M flag).
- Optional information such as a domain name and a list of DNS server addresses.

Unlike an IPv4 device, an IPv6 device can determine all of its addressing dynamically without the services of a DHCP server.

As shown in Figure 2-7, the Router Advertisement message can be one of three methods:

- **Method 1—Stateless Address Autoconfiguration (SLAAC):** The device uses the information in the RA message for all of its addressing needs, including using the prefix in the RA to create an IPv6 global unicast address. The device will use the source IPv6 address of the RA as its default gateway. Method 1 is the default in Cisco IOS.
- **Method 2—SLAAC and stateless DHCPv6 server:** Similar to Method 1, the device uses SLAAC to create a global unicast address and uses the source IPv6 address of the RA as the address for the default gateway. However, this method also suggests to the device that it needs to contact a stateless DHCPv6 server for additional information that is not contained in the RA message. This information may be a list of DNS server addresses. It's important to note that a stateless DHCPv6 server does not provide or maintain any IPv6 global unicast addressing information. A stateless server only provides network information common to all devices on the network.

Note The RA message doesn't specify what information is available from the stateless DHCPv4 server. A list of DNS addresses should be included in the RA message only if both the router and the receiving device support this capability (see RFC 6106, *IPv6 Router Advertisement Options for DNS Configuration*). This is discussed in detail in Chapter 9, "Stateless Address Autoconfiguration (SLAAC)."

- **Method 3—Stateful DHCPv6 server:** This method is similar to DHCP for IPv4. The RA message suggests to the device that it use a DHCPv6 server for all its IPv6 addressing needs, including a global unicast address. However, the device must dynamically obtain its default gateway address from the RA message, as in the previous two methods.

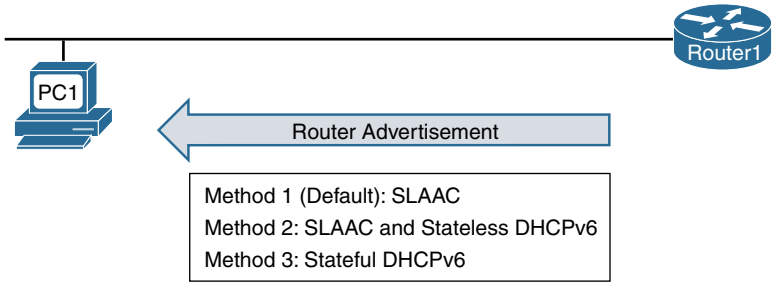


Figure 2-7 *ICMPv6 Router Advertisement Methods*

Method 1 and Method 2 both suggest that the client device uses SLAAC to create its own IPv6 global unicast address. The client uses the prefix in the Router Advertisement message and then creates a 64-bit Interface ID, which can be generated in one of two ways:

- **Random 64-bit value:** The operating system can generate a random 64-bit value for the Interface ID. (This is the default on the Windows operating systems.)
- **EUI-64 (Extended Unique Identifier):** This method uses the interface's 48-bit Ethernet MAC address and inserts 16 more bits (fffe) in the middle. The seventh bit gets flipped. Don't worry about this process right now; it will be explained in more detail in Chapter 9. (This is the default on Linux and Mac OS.)

Note Cryptographically Generated Addresses (CGA) is a third option in which to generate the Interface ID. This is done by computing a cryptographic one-way hash function from a public key and involves the Secure Neighbor Discovery Protocol (SEND). CGA and SEND are beyond the scope of this book. For more information on CGA and SEND, I recommend the book, IPv6 Security, by Eric Vyncke.

In either case, the device is able to create its own global unicast address without the services of DHCPv6.

Neither a stateless nor stateful DHCPv6 server provides the default gateway address. It can only be dynamically obtained from the RA message. This means that even if a stateful DHCPv6 is used, the RA messages must be sent so that hosts can dynamically receive their default gateway information.

When the RA suggests the use of stateful DHCPv6, some host operating systems will use both stateful DHCPv6 and SLAAC for creating two different addresses. In most cases, network administrators that implement stateful DHCPv6 don't usually want the devices to create another address using SLAAC. Therefore, the information in the RA message must be configured so that the host uses the RA message for the default gateway address but does not assign a second address using SLAAC. (This mystery is solved in Chapter 12, "Stateful DHCPv6".)

Summary

This chapter provides an introduction to some terms and concepts that are explained more thoroughly throughout this book. The goal of this chapter is to provide just enough information to help you understand the protocols and concepts discussed in other chapters. Everything in this chapter will be discussed later in the book and in more detail.

Here are some of the key items discussed in this chapter:

- IPv6 addresses are represented using the hexadecimal number system. Any 4 bits can be represented with a single hexadecimal digit.

- IPv6 has a variety of address types, some of which are introduced in this chapter:
  - **Global unicast address (GUA):** This is a globally unique and routable IPv6 address equivalent to a public IPv4 address.
  - **Link-local unicast address:** Required for an IPv6 device, the link-local address is limited to the link and not routable beyond the subnet.
  - **Solicited-node multicast address:** This special multicast address is used by some IPv6 protocols in lieu of a broadcast address in IPv4. A solicited-node multicast address is mapped to an Ethernet MAC multicast address, which can be filtered by the device's NIC. This allows the NIC to determine whether the information might be for this device instead of unnecessarily passing the message to an upper-layer protocol for examination.
- ICMPv6 Neighbor Discovery Protocol (NDP) is used for on-link (same subnet) device discovery and messaging. It has five message types, four of which are new:
  - **Neighbor Solicitation and Neighbor Advertisement messages:** Used for Layer 3 to Layer 2 address resolution, the Neighbor Solicitation and Neighbor Advertisement messages are similar to the ARP Request and ARP Reply messages in IPv4.
  - **Router Solicitation and Router Advertisement messages:** A Router Solicitation message is sent by a device to request a Router Advertisement message from an IPv6 router. The RA message suggests to a device how to obtain its addressing information dynamically. The RA message also contains information that a device can use to create its own address along with the address of the default gateway.
- IPv6 performs dynamic address allocation differently than IPv4. The ICMPv6 Router Advertisement message is sent by an IPv6 router to suggest to devices how to obtain this information. The RA message will advertise one of three methods:
  - **Method 1—SLAAC:** Devices use the information in the RA message to create their own IPv6 global unicast address. The RA message contains all the information a device needs, which may include the domain name and DNS server addresses.
  - **Method 2—SLAAC and stateless DHCPv6:** Devices use the information in the RA message to create their IPv6 global unicast addresses. The RA message may contain other information as well, but the device should contact a stateless DHCPv6 server for additional information. A stateless DHCPv6 server may provide information such as DNS server addresses.
  - **Method 3—Stateful DHCPv6:** This method is similar to DHCP for IPv4. This method suggests to the device that it contact a stateful DHCPv6 server for all its addressing information, including a global unicast address.

Review Questions

1. Convert the following 4-bit binary values to hexadecimal.
   a. 0010
   b. 1100

c. 1111

d. 1101

e. 0000

f. 1010

2. Convert the following hexadecimal values to binary using 4 bits.

   a. E

   b. 3

   c. 8

   d. B

   e. 1

   f. 9

3. Match each description to one or more of the following types of IPv6 addresses:

   - Global unicast address
   - Link-local address
   - Unspecified address
   - Solicited-node multicast address

   Descriptions:

   a. Every IPv6-enabled interface must have this type of address.

   b. Equivalent to a public IPv4 address.

   c. Can only be used as a source address.

   d. Mapped to an Ethernet MAC address that can be filtered by the NIC.

   e. Used in the absence of an IPv6 address or for anonymity.

   f. Can be created using SLAAC or obtained from a stateful DHCPv6 server.

   g. Not routable.

4. Match each description to one of the following types of ICMPv6 Neighbor Discovery messages:

   - Neighbor Solicitation
   - Neighbor Advertisement
   - Router Solicitation
   - Router Advertisement

Descriptions:

a. Includes the prefix, the prefix length, and the source IPv6 address used as the default gateway address.

b. Equivalent to an ARP Request message for IPv4.

c. Equivalent to an ARP Reply message for IPv4.

d. Sent by a device to request how to obtain IPv6 address information dynamically.

5. Match each description to one or more of the following types of Router Advertisement provisioning methods:

   - Method 1—SLAAC
   - Method 2—SLAAC and stateless DHCPv6
   - Method 3—Stateful DHCPv6

   Descriptions:

   a. Similar to DHCP for IPv4.

   b. Provides the address for the default gateway.

   c. Provides the prefix and prefix length to be used to create a global unicast address.

   d. Uses a DHCPv6 server only for DNS or domain name.

   e. The DHCPv6 server is not necessary for any information.

   f. The DHCPv6 server allocates the global unicast address.

References

RFCs

RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, Cisco Systems, www.ietf.org/rfc/rfc2460.txt, December 1998.

RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, R. Droms, Cisco Systems, www.ietf.org/rfc/rfc3315, July 2003.

RFC 3736, *Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6*, R. Droms, Cisco Systems, www.ietf.org/rfc/rfc3736, April 2004.

RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, A. Conta, Transwitch, www.ietf.org/rfc/rfc4443.txt, March 2006.

RFC 4861, *Neighbor Discovery for IP Version 6 (IPv6)*, T. Narten, IBM, www.ietf.org/rfc/rfc4861.txt, September 2007.

RFC 4862, *IPv6 Stateless Address Autoconfiguration*, S. Thompson, Cisco Systems, www.ietf.org/rfc/rfc4862, September 2007.

Chapter 3

Comparing IPv4 and IPv6

"It's exactly the same, only completely different."

—Anonymous

Chapter 2, "IPv6 Primer," introduces some of the differences in how IPv6 operates compared to IPv4. It talks, for example, about ICMPv6 Neighbor Discovery Protocol (NDP), with its four new message types that change the way IP performs address resolution and dynamic addressing.

This chapter focuses on the IPv6 protocol header, and compares it to the IPv4 header. The structure of the IPv6 header is discussed in RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*. In addition to the main IPv6 header, there is a new type of IPv6 header known as an *extension header*, which is also discussed in this chapter.

This chapter examines the following topics:

- Comparing the IPv4 and IPv6 headers
- IPv6 over Ethernet
- Packet analysis using Wireshark
- IPv6 extension headers
- IPv4 and IPv6 differences at a glance

Comparing the IPv4 and IPv6 Headers

Figure 3-1 shows the structure of the IPv4 header, as defined in RFC 791, *Internet Protocol, DARPA Internet Program Protocol Specification*. As you can see in the figure, some fields are the same as or similar to fields in the IPv6 header, while others have been removed in IPv6.

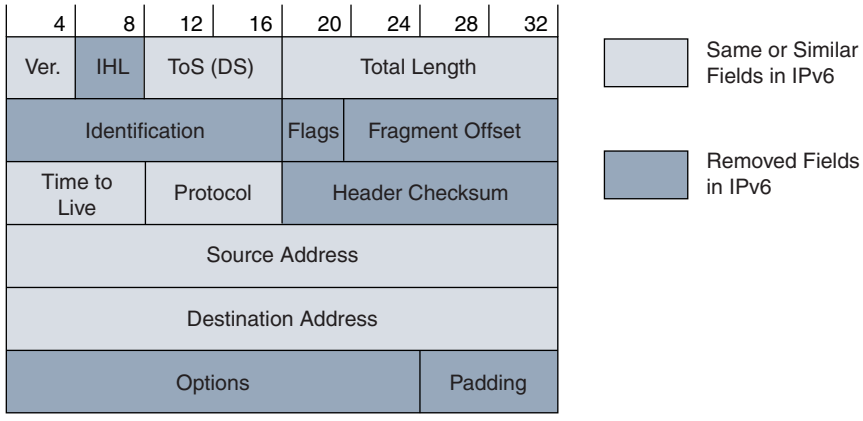


Figure 3-1 *IPv4 Header*

IPv6 is defined in RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*. Figure 3-2 shows the basic structure of the IPv6 header or what is sometimes referred to as the *main IPv6 header*. The main IPv6 header can also point to one or more IPv6 extension headers. Extension headers are explained later in this chapter.

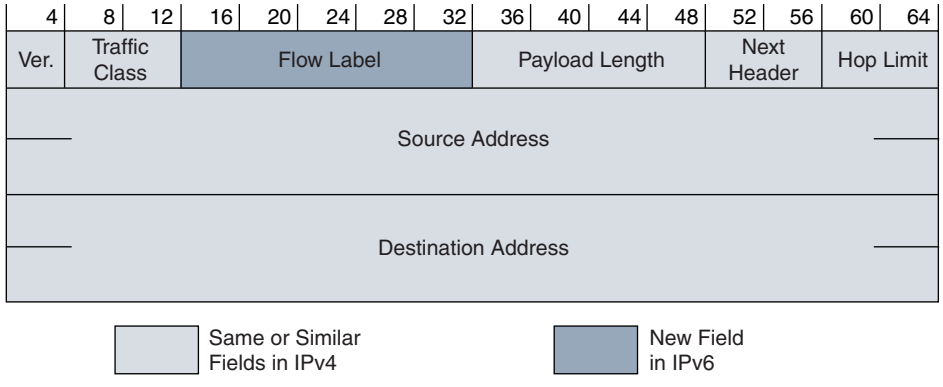


Figure 3-2 *IPv6 Header*

In comparing the two headers, notice that the IPv6 header is illustrated here as a 64-bit-wide header, compared to the 32-bit-wide header typically seen for IPv4. Even though either header can be drawn in either bit-wide format, we will use the 64-bit- wide format for IPv6.

Although 64-bit CPUs were not common at the time IPv6 was being defined, the designers of IPv6 decided to take advantage of 64-bit CPUs. As a result, all IPv6 fields start on an even 64-bit boundary or a multiple of 64. The advantage is that 64-bit CPUs can read one 64-bit-wide memory word at time. However, this structure doesn't negatively affect 32-bit CPUs because a 64-bit boundary is also a 32-bit boundary.

Note If the nuances of why you commonly see IPv6 shown using a 64-bit-wide header are lost on you, don't worry about it. I have included it just as a point of interest.

Note It has been said that one of the reasons the IPv6 address was changed from the SIPP proposal of 64 bits to a 128-bit address, instead of something in between, was to use a multiple of 64.

A quick comparison of the two headers reveals that the IPv6 header is a simpler protocol with fewer fields than its IPv4 counterpart. This makes IPv6 a leaner protocol and provides more efficient processing. As you will soon see, another advantage of IPv6 is that it uses a fixed 40-byte header, compared to the variable-length header used in IPv4.

The IPv4 and IPv6 Version Fields

Both IPv4 and IPv6 begin with the Version field, which contains the version number of the Internet Protocol (IP) header. This is probably obvious, but in IPv4 the value is always 4, and in IPv6, the value is always 6.

As we discussed in Chapter 1, "Introduction to IPv6," the experimental Internet Stream Protocol (ST2) in 1990 had already used the value 5 in the Version field. Although ST2 was never known as IPv5, when encapsulated in IPv4, it used the value of 5 for the version.

IPv4 Internet Header Length (IHL) Field

The IPv4 Internet Header Length (IHL) field, shown in Figure 3-3, is the length of the IPv4 header in 32-bit words, including any optional fields or padding. In effect, this points to where the IPv4 header ends and the data or payload begins. The minimum value is 5 (5 × 32-bit words = 160 bits, or 20 bytes [octets]). This is equal to the minimum size of an IPv4 header, excluding any Options or Padding. The Options and Padding fields can extend the length of the IHL beyond the minimum 20 bytes, up to a maximum of 60 bytes.

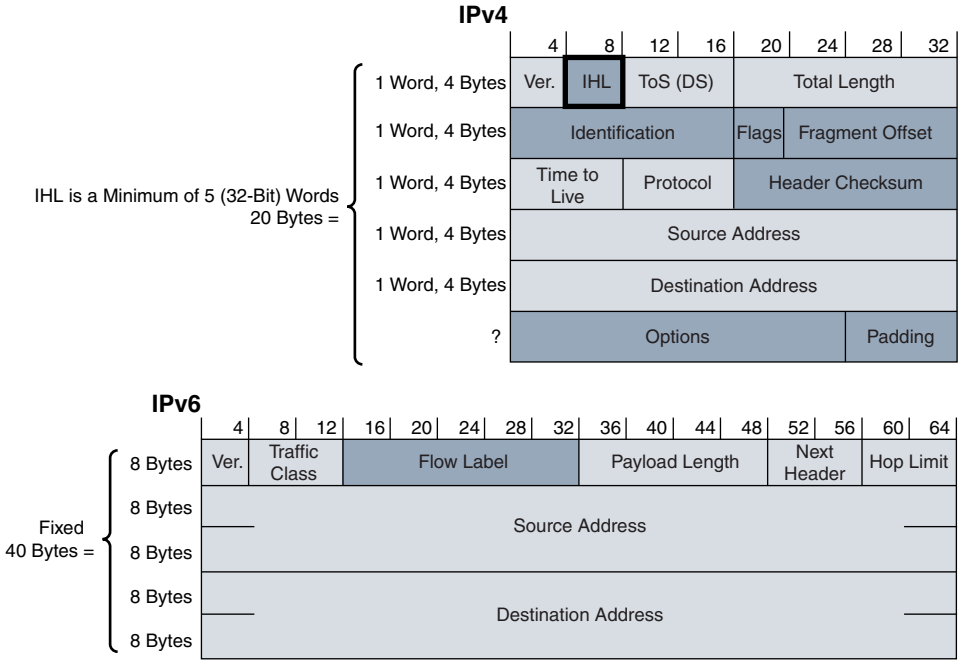


Figure 3-3 *IPv4 Internet Header Length (IHL) Field*

IPv6 doesn't have an IHL field because the main IPv6 header has a fixed length of 40 bytes, which allows for more efficient processing. As you will see later in this chapter, IPv6 uses an optional extension header to add any additional features to the IPv6 header. However, the extension header does not change the fixed length of the 40-byte main IPv6 header.

IPv4 Type of Service (ToS) and IPv6 Traffic Class Fields

The IPv4 Type of Service (ToS) field and the IPv6 Traffic Class field are identical fields; only the name was changed in IPv6. Figure 3-4 shows these fields for both protocols. IPv4 ToS and IPv6 Traffic Class are used to specify what type of treatment the packet should receive from routers. This information helps provide Quality of Service (QoS) features by offering different degrees of precedence. When multiple packets are queued to be transmitted out the same interface, the value in this field can be used to determine the treatment of the packets, including the order in which the packets are sent.

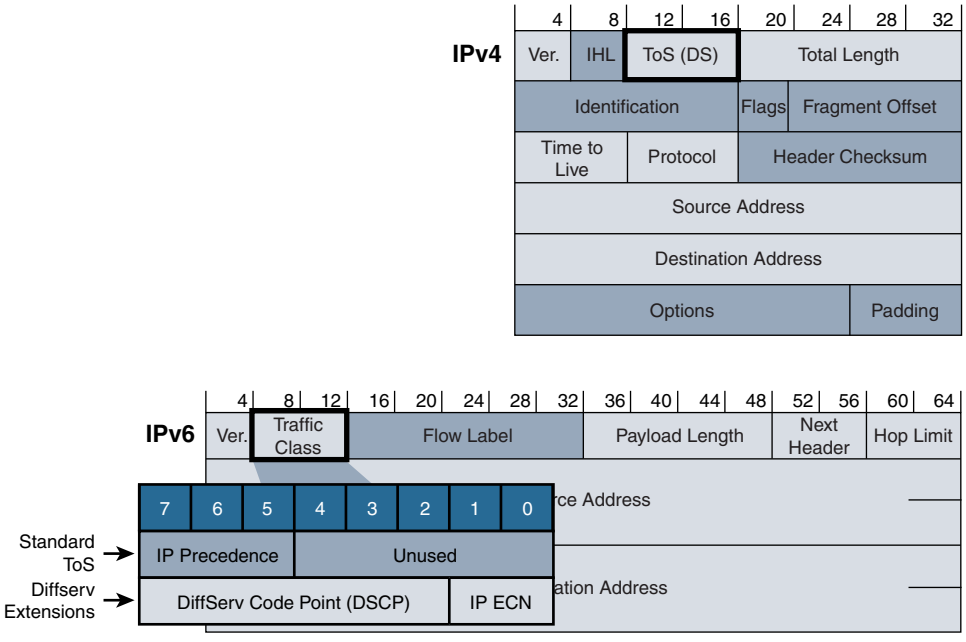


Figure 3-4 *IPv4 ToS Field and IPv6 Traffic Class Field*

The IPv4 ToS field, which used three IP Precedence field bits, was not widely used as originally designed, so in 1998, the Internet Engineering Task Force (IETF) redefined it further, using a technique called Differentiated Services (DS), specified in RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*.

IPv6 uses this same Differentiated Services technique. Using 6 bits for the DSCP (Differentiated Services Code Point) allows a possibility of 64 markings. This provides much more granularity in priority selection than the original IPv4 Precedence field and its 3 bits with 8 values.

Routers can ignore this field, but when implementing QoS, it's a good idea to become familiar with it. The original terminology of the IPv4 Type of Service has been superseded in IPv6 by the *diffserv* terminology used in RFC 2474.

Note DSCP and IP Precedence are beyond the scope of this book. One interesting note is that the IP Precedence value is actually the first 3 bits of the DSCP value, as shown in Figure 3-4. Therefore, both values cannot be used simultaneously. If DSCP with its additional 3 bits is used, it supersedes IP Precedence. To learn more about QoS, I recommend the book *End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks*, by Tim Szigeti et al.

IPv6 Flow Label Field

The IPv6 Flow Label field, as shown in Figure 3-5, is a new field used to tag a sequence or flow of IPv6 packets sent from a source to one or more destination nodes. This flow can be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as "real-time" service.

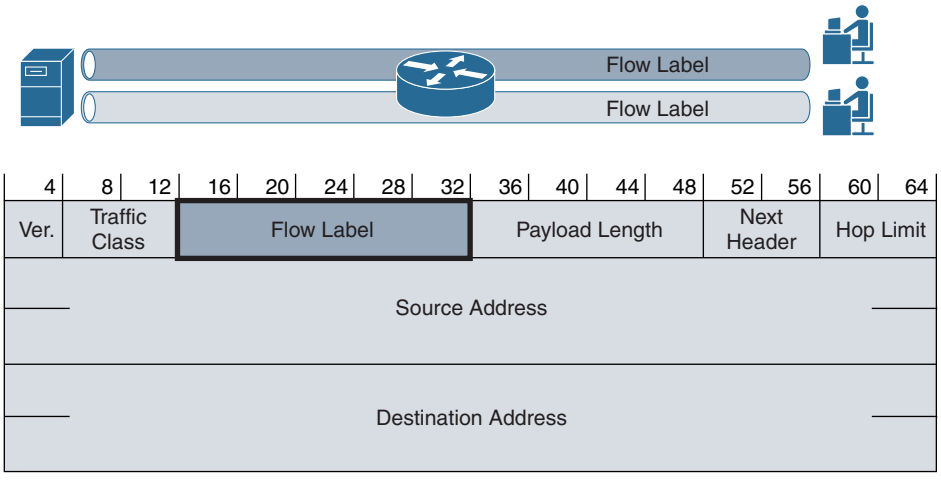


Figure 3-5 *IPv6 Flow Label Field*

The Flow Label field is used to help identify all the packets within the same flow to ensure that all the packets receive the same type of handling by the IPv6 routers. Flow Label usage is described in RFC 6437, *IPv6 Flow Label Specification*. Routers keep track of the individual packet flows. Because routers do not have to independently process each packet's header, these multipacket flows are processed more efficiently.

Currently, there aren't a lot of implementations that look at the Flow Label. Two use cases have been defined: Equal Cost Multi-Path (ECMP), defined in RFC 6438, *Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels*, and Server Load Balancing (SLB), defined in RFC 7098, *Using the IPv6 Flow Label for Load Balancing in Server Farms*. Many systems set the Flow Label for packets that belong to different TCP sessions. A Flow Label set to 0 means the traffic is not associated with any flow.

IPv4 Total Length Field, IPv6 Payload Length Field, and IPv6 Jumbograms

The IPv4 Total Length field is the length of the entire IPv4 packet, measured in bytes, including the IPv4 header and the data, as illustrated in Figure 3-6. This is a 16-bit field, so the maximum size of an IPv4 packet is 65,535 bytes. Most IPv4 packets are much smaller.

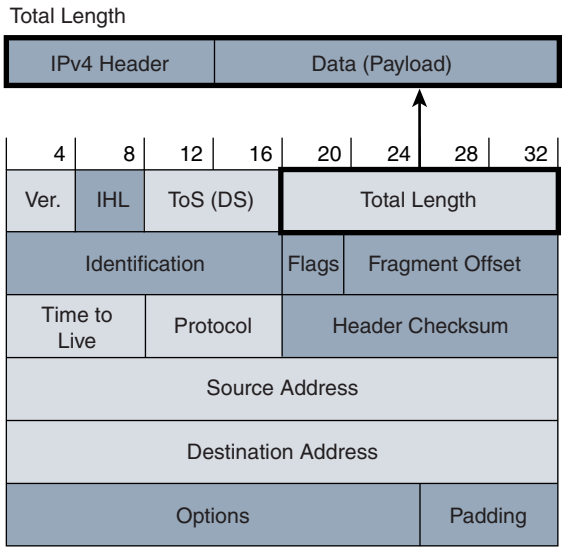


Figure 3-6 *IPv4 Total Length Field*

The IPv6 Payload Length field is a 16-bit field that indicates the length in bytes of just the payload following the main IPv6 header or, in other words, the data portion of the packet (see Figure 3-7). It does not include the main IPv6 header. If the IPv6 packet has one or more extension headers, they are included in the number of bytes contained in the Payload Length field. Extension headers are considered part of the payload.

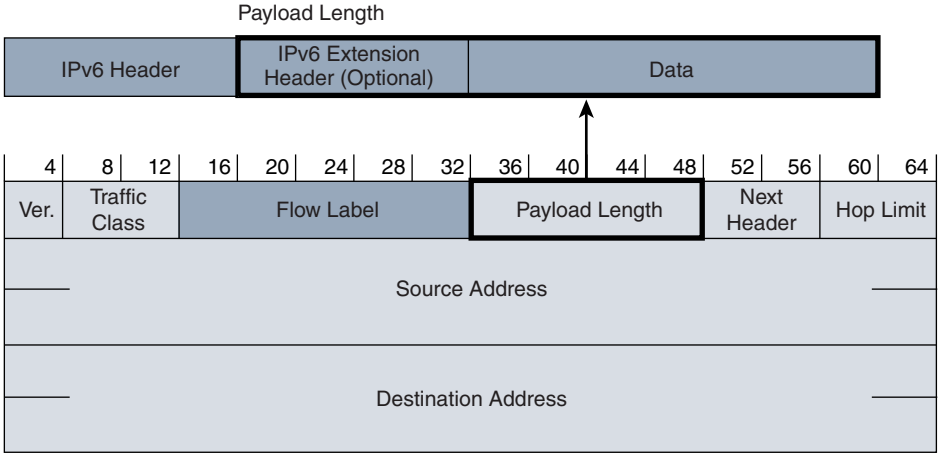


Figure 3-7 *IPv6 Payload Length Field*

The IPv6 Payload Length field is similar to the Total Length field in the IPv4 header, except for one important difference: IPv4's Total Length field includes both the IPv4 header and the data—the *total* length of the IPv4 packet. However, the IPv6 Payload

Length field specifies only the number of bytes of data or *payload*; it does not include the main IPv6 header. The IPv4 header can vary in length because of the Padding and Options fields, whereas the IPv6 header is fixed at 40 bytes.

Note The calculation to determine the number of bytes in the IPv4 payload (data) following the IPv4 header is IPv4 Total Length – IPv4 IHL = number of bytes in the IPv4 payload (data). Remember that the IPv4 IHL field is in 32-bit words, whereas the IPv4 Total Length field is in bytes.

As mentioned earlier, IPv4 Total Length is a 16-bit field, therefore allowing for IPv4 packet sizes up to 65,355 bytes. In reality, these packet sizes are much smaller due to the maximum transmission unit (MTU) of the links. IPv4 has no options for exceeding this theoretical limit. However, IPv6 does have the ability to carry larger payloads. This type of packet is known as a jumbogram, as described in RFC 2675, *IPv6 Jumbograms*. A *jumbogram* is an IPv6 packet that contains a payload greater than 65,535 bytes, the maximum allowed with the 16-bit IPv6 Payload Length field.

Jumbograms use the Jumbo Payload option in the Hop-by-Hop extension header, which is described later in this chapter. The Jumbo Payload option uses a 32-bit-length field to allow transmission of IPv6 packets with payloads between 65,536 and 4,294,967,295 (2^{32}–1) bytes (4 GB–1).

As with IPv4, the size of IPv6 packets is still limited by today's link layer MTUs. However, IPv6 jumbograms are being used for fast links inside or between supercomputers.

IPv4 and IPv6 MTUs

Most transmission links enforce a maximum packet length known as the MTU (maximum transmission unit). The IPv4 or IPv6 MTU is the total length of the IP packet, including the header, as shown in Figure 3-8.

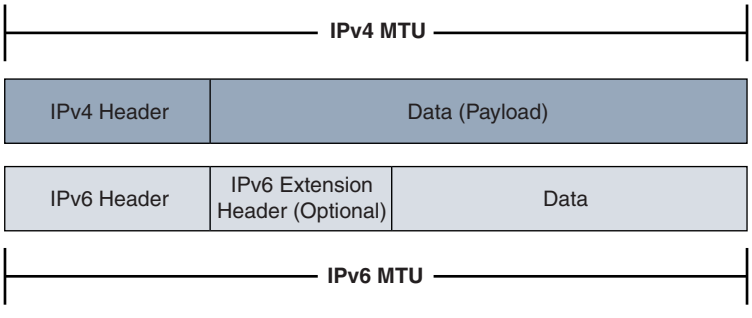


Figure 3-8 *IPv4 and IPv6 MTUs*

IPv4 requires that every node be able to forward an IPv4 packet of 68 bytes without any further fragmentation. This is because an IPv4 header can be as large as 60 bytes in length, which would leave a payload of only 8 bytes. The payload would have to be

an IPv4 fragment. Otherwise the payload would need to include header information for another protocol, which would make it larger than 8 bytes. Every IPv4 node that is the final destination of the IPv4 packet must be able to receive an IPv4 packet of a minimum size of 576 bytes, which can be all or fragments of the original packet.

IPv6 requires that every link have a minimum MTU of 1280 bytes, with a recommended MTU of 1500 bytes, compared to 68 bytes in IPv4.

IPv4 Fragmentation

IPv4 was designed for a wide variety of transmission links. The design of IPv4 accommodates MTU differences by allowing routers to fragment IPv4 packets when an MTU along the path is smaller than the sender's MTU. If a router receives an IPv4 packet that is larger than the MTU of the outgoing interface, this packet can be fragmented, depending on the options in the IPv4 header. Sometimes packets are fragmented into multiple packets at the source. The final destination of the IPv4 packet is responsible for reassembling the fragments into the original full-size IPv4 packet.

Fragmentation divides (fragments) IPv4 packets so that they can be forwarded out a link that doesn't support the size of the original packet. The destination device reassembles the fragments. IPv4 Identification, Flags, and Fragment Offset fields are used for packet fragmentation and reassembly (see Figure 3-9).

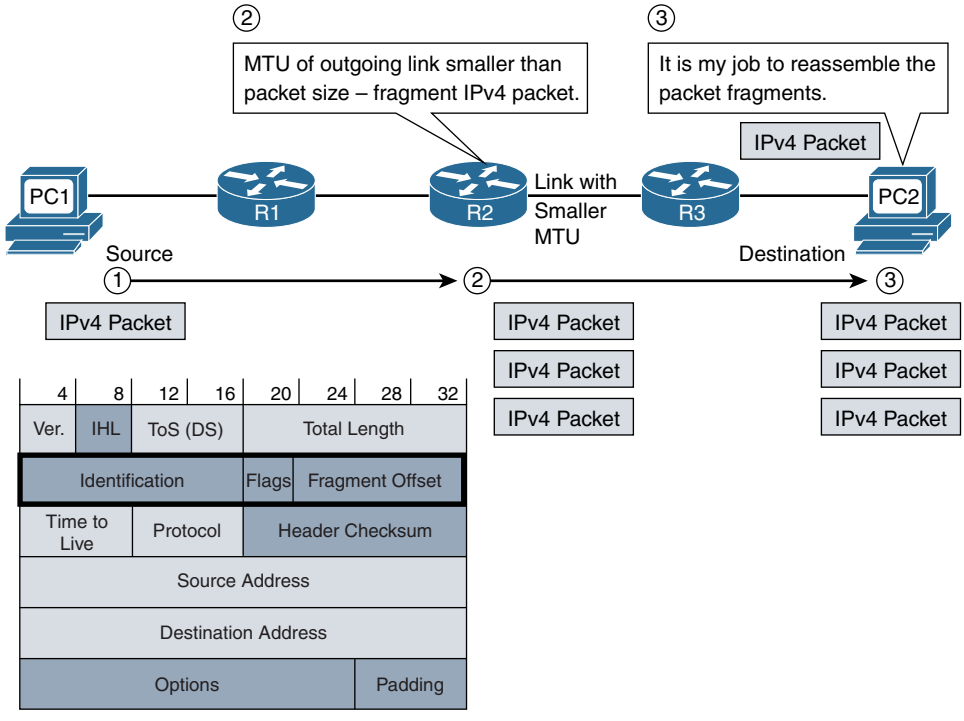


Figure 3-9 *IPv4 Fragmentation Fields*

The IPv4 fields used for fragmentation are as follows:

Note If you don't feel you need this information about IPv4 fields used for fragmentation, skipping it will not deter you from understanding IPv6.

- **Identification field (16 bits):** Most messages sent over a network consist of many packets. Each IPv4 packet within a message has a unique value in the 16-bit Identification field. When an IPv4 packet needs to be fragmented into two or more packets, the Identification field is a common value in all the fragmented packets to help the receiver reassemble the fragments.
- **Flags field (3 bits):** The first bit of this field is 0, which means it is reserved or not used. The second bit is known as the DF, or Don't Fragment, bit. When it is set to 1, the packet should not be fragmented. However, most protocols don't care about the fragmentation process and set this flag to 0, which means this packet can be fragmented if needed. The third bit is the More Fragments flag, which is used to indicate whether this is the last fragment (0) or whether there are more fragments to follow (1). If a packet is not fragmented, there is only one fragment in the entire packet, and this flag is set to 0.

Note The DF flag is very useful when testing the MTU of a path between the source and the destination. If the DF flag is set to 1, the IPv4 packet should not be fragmented. Any router along the path whose MTU is smaller than the packet drops the packet and sends an ICMP Destination Unreachable message back to the source. The ICMP message includes the MTU of the router's egress interface. Path MTU Discovery for IPv4 is beyond the scope of our discussion, but RFC 1191, *Path MTU Discovery*, explains this process.

- **Fragment Offset field (13 bits):** When an IPv4 packet is fragmented, this field specifies the offset or position where this data goes in units of 8 octets (64 bits). Basically, the Fragment Offset field notifies the receiver where to align this fragmented packet in relation to the other fragmented packets. The first fragment has offset zero. If the packet is not fragmented, this value is 0.

IPv6 Fragmentation: IPv6 Source Only

Unlike in IPv4, an IPv6 router does not fragment a packet unless it is the source of the packet. Intermediate nodes (routers) do not perform fragmentation. Compare the two headers in Figure 3-10 and notice that the fields used in the IPv4 header for

fragmentation do not exist in the IPv6 header. You will see how an IPv6 device fragments packets when it is the source of the packet later in this chapter, in the discussion of extension headers.

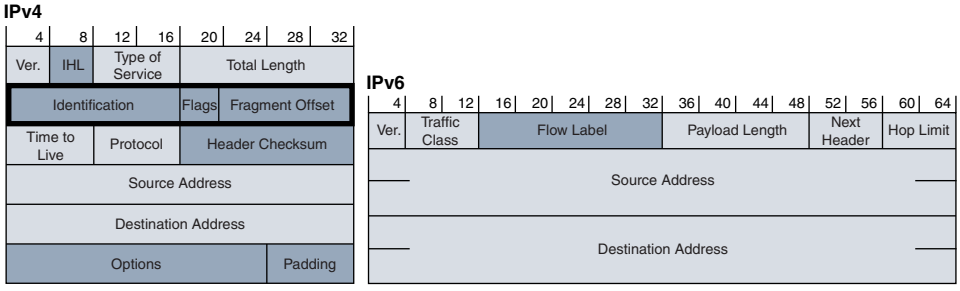


Figure 3-10 *Fragmentation Fields in IPv4 Header That Do Not Exist in IPv6 Header*

When an IPv6 router receives a packet larger than the MTU of the egress interface, the router drops the packet and sends an ICMPv6 Packet Too Big message back to the source. The Packet Too Big message includes the MTU size of the link in bytes so that the source can change the size of the packet for retransmission. Path MTU discovery and ICMPv6 Packet Too Big message are discussed in Chapter 12, "ICMPv6."

Data is usually transmitted as a series of packets, sometimes referred to as a *packet train*. The larger the packets, the fewer packets that may be needed to be transmitted. So it is preferred to use the largest-size packet possible that can be supported by all the links from the source to the destination. This is known as the Path MTU (PMTU). A device can use Path MTU discovery to determine the minimum link MTU along the path. RFC 1981, *Path MTU Discovery for IP Version 6*, suggests that IPv6 devices should perform ICMPv6 PTMU discovery to avoid source fragmentation.

IPv4 Protocol and IPv6 Next Header Fields

The IPv4 Protocol field indicates the protocol carried in the data portion of the IPv4 packet. IPv6 has a similar field, the Next Header field, that specifies the type of header expected after the main IPv6 header. Although it is similar to the IPv4 Protocol field, there is an additional option (discussed in a moment). These two fields are shown in Figure 3-11.

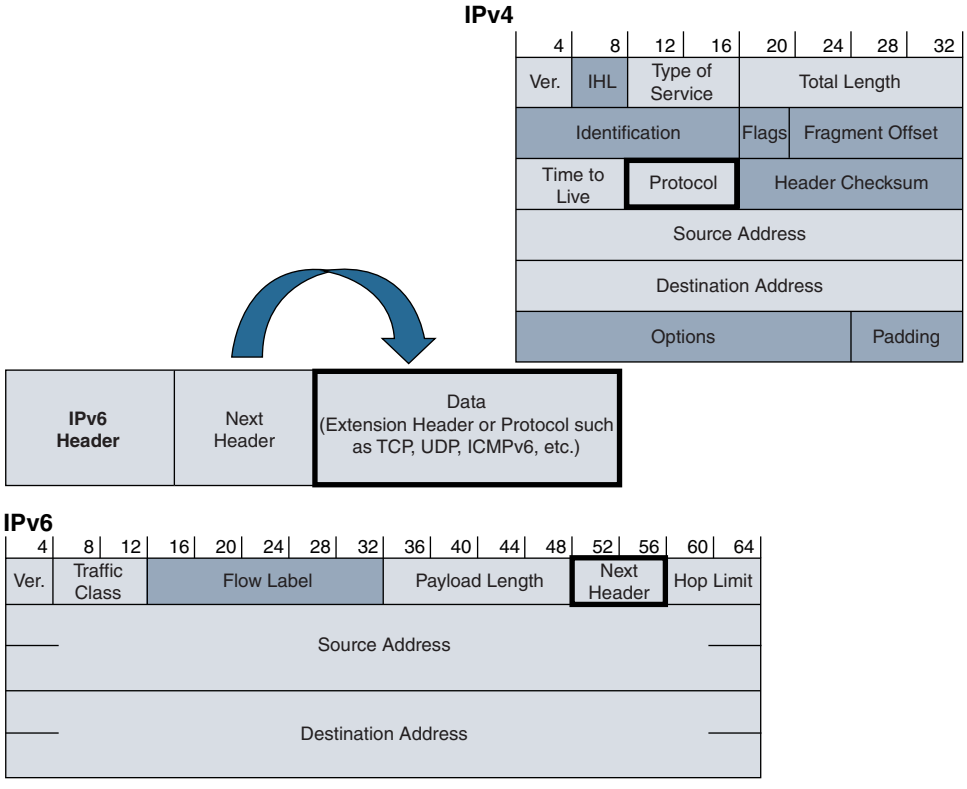


Figure 3-11 *IPv4 Protocol Field and IPv6 Next Header Field*

The initial values for various protocols are specified in RFC 1700, *Assigned Numbers*, and were later replaced by an online database maintained by the Internet Assigned Numbers Authority (IANA) at www.iana.org/assignments/protocol-numbers/protocol-numbers.xml. The same values used in the IPv4 Protocol field are used in the IPv6 Next Header field, along with some additional values for IPv6. Some of the most common values for both protocols are 6 for TCP and 17 for UDP.

In a situation when there is only the main IPv6 header and no extension headers, the Next Header field specifies the protocol carried in the data portion of the IPv6 packet. This is similar to the Protocol field in the IPv4 header. Table 3-1 shows some of the significant IPv6 Next Header field values.

Table 3-1 *Significant IPv6 Next Header Field Values*

Next Header Field Value (Decimal)	Next Header Field Value (Hexadecimal)	Description
0	0	Hop-by-hop options extension header for IPv6
1	1	Internet Control Message Protocol version 4 (ICMPv4)
2	2	Internet Group Management Protocol version 4 (IGMPv4)
4	4	IPv4 encapsulation
5	5	Internet Stream Protocol (ST)
6	6	Transmission Control Protocol (TCP)
8	8	Exterior Gateway Protocol (EGP)
17	11	User Datagram Protocol (UDP)
41	29	IPv6 encapsulation
43	2B	Routing extension header for IPv6
44	2C	Fragment header for IPv6
46	2E	Resource Reservation Protocol (RSVP)
47	2F	Generic Routing Encapsulation (GRE)
50	32	Encapsulating Security Payload (ESP)
51	33	Authentication Header (AH)
58	3A	Internet Control Message Protocol version 6 (ICMPv6)
59	3B	No Next Header for IPv6
60	3C	Destinations options extension header for IPv6
88	58	Enhanced Interior Gateway Routing Protocol (EIGRP)
89	59	Open Shortest Path First (OSPF)

Figure 3-12 shows three examples of the IPv6 Next Header field in decimal, indicating the information following the main IPv6 header. Usually, the information after the main IPv6 header is the data or payload, such as a TCP segment. But as the last example in Figure 3-12 illustrates, the Next Header field can also point to an extension header. Extension headers are examined later in this chapter.

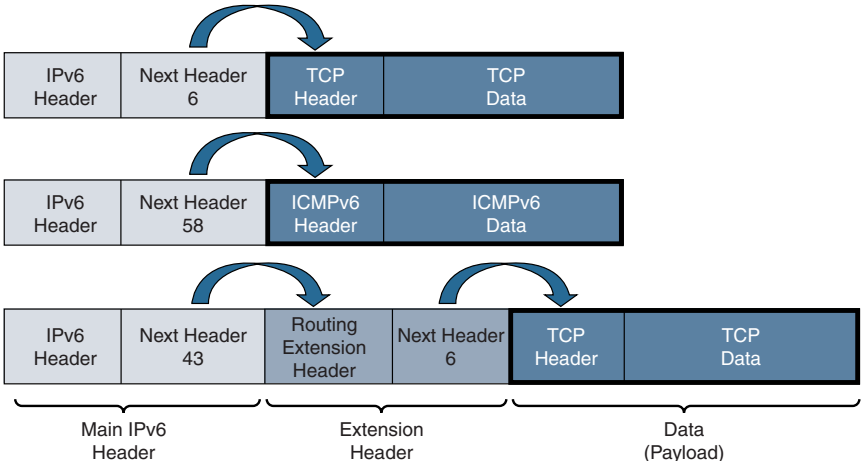


Figure 3-12 *IPv6 Next Header Examples*

IPv4 Time to Live (TTL) and IPv6 Hop Limit Fields

The IPv4 Time to Live (TTL) and IPv6 Hop Limit fields ensure that packets do not transit between networks for an indefinite period of time, as in the case of a routing loop. These fields are decremented by 1 each time a router receives the IPv4 or IPv6 packet. When the field contains the value 0, the packet is discarded, and an ICMPv4 or ICMPv6 Time Exceeded message is sent to the source of the packet. (ICMPv6 messages are discussed in Chapter 12.) Both of these fields are shown in Figure 3-13.

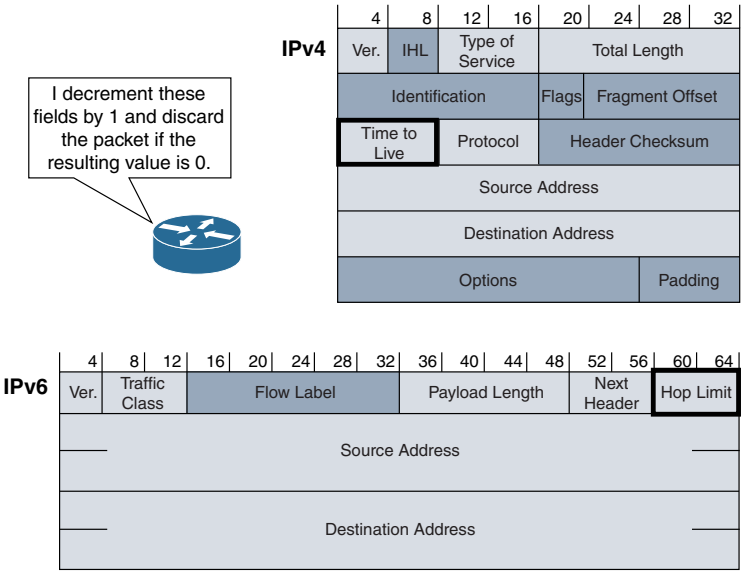


Figure 3-13 *IPv4 TTL Field and IPv6 Hop Limit Field*

The IPv4 Time to Live field was changed to Hop Limit in IPv6 to more accurately reflect the way routers treat this field by decrementing the field by 1. The IPv4 TTL field was originally intended to represent the actual maximum amount of time that the packet is allowed to traverse the network and not necessarily the number of router hops. RFC 791 states, "Even if no local information is available on the time actually spent, the field must be decremented by 1. The time is measured in units of seconds (i.e., the value 1 means one second). Thus, the maximum time to live is 255 seconds or 4.25 minutes." Instead of calculating the amount of time, routers just decrement the IPv4 TTL by 1, in effect making it the number of hops.

Checksums: IPv4, TCP, and UDP

A checksum for the IPv4 header is provided for protection against any corruption in transit. This is not the more complex cyclic redundancy check (CRC) used by Ethernet but a much simpler 16-bit checksum performed only on the IPv4 header. Each router along the path verifies and recomputes this field. If the checksum fails, the router discards the packet.

There is no Checksum field in the IPv6 header, as shown in Figure 3-14. The designers of IPv6 did not include a Checksum field because Layer 2 data link technologies such as Ethernet perform their own checksum and error control. Also, upper-layer protocols such as TCP and UDP have their own checksums, and therefore a checksum at Layer 3 is redundant and unnecessary.

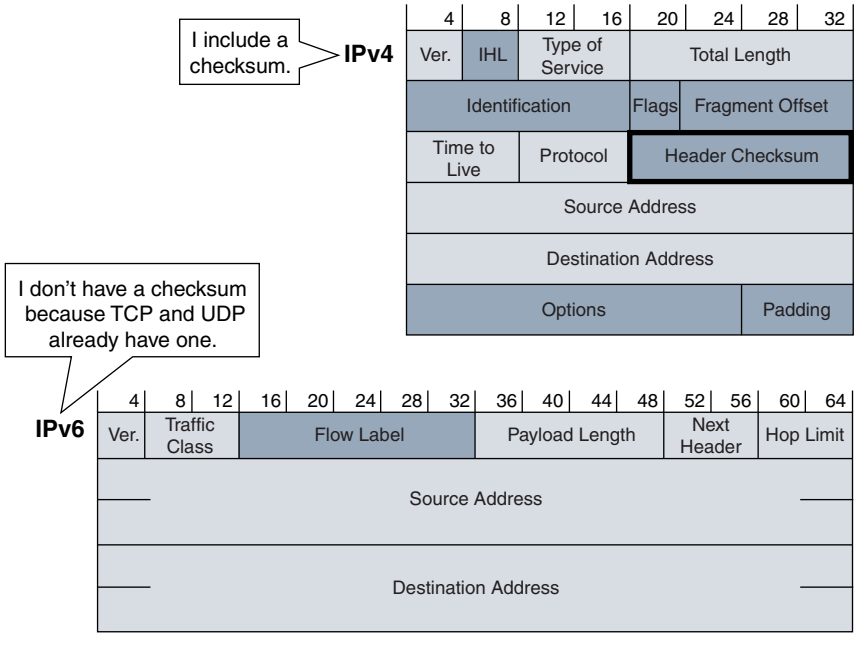


Figure 3-14 *IPv4 Checksum Field*

Because there is no Checksum field in IPv6, the UDP checksum, which is optional in IPv4, is therefore mandatory in IPv6. The Checksum field is used to verify the integrity of the UDP header and data.

The Checksum field in Transmission Control Protocol (TCP) is mandatory when carried over both IPv4 and IPv6. TCP and UDP run on top of IPv6 without any structural modifications to these protocols.

Checksums are used by various protocols and at different layers. A checksum is a fixed-size computation on a block of data for the purpose of detecting accidental errors that might have been introduced during transmission. Any transport layer or other upper-layer protocol that includes the IPv4 addresses in its checksum computation must be modified for use over IPv6. This is necessary to include the larger 128-bit IPv6 addresses. TCP and UDP both include a 16-bit checksum in their headers, as shown in Figure 3-15.

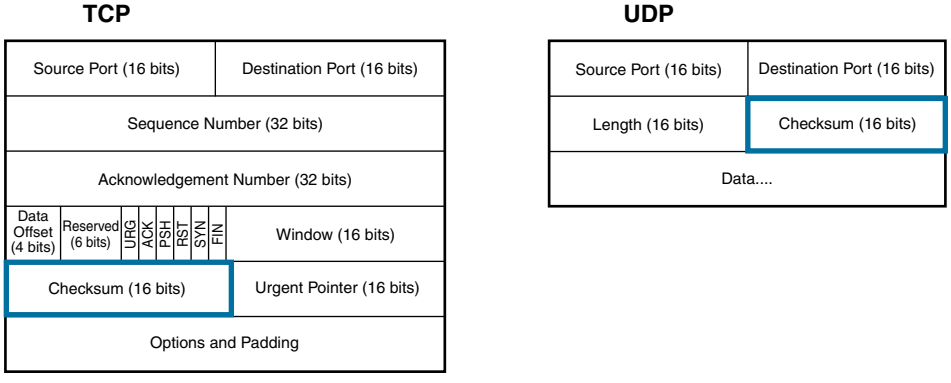


Figure 3-15 *TCP and UDP*

Both TCP and UDP generate a pseudoheader for checksum computational purposes only. The pseudoheader contains the fields from the network and transport layers that are used to compute the checksum. When TCP or UDP is transported over IPv6, this checksum includes the following:

- Source IPv6 address
- Destination IPv6 address
- Upper-layer payload length (length of the TCP/UDP header and data)
- IPv6 next-header value (including any extension headers)

Modification to both TCP and UDP protocols is required for transport over IPv6. Because both of these checksums use the IPv6 addresses, the internals of the protocol had to be rewritten to accommodate the longer addresses. Although the addresses are longer, the same algorithm is used to compute the checksum in IPv6 that is used in IPv4.

An IPv4 header includes a checksum, so the use of the checksum in UDP over IPv4 is optional. However, in the IPv6 header, the checksum was removed to improve processing speed. Besides the transport layer protocols, UDP and TCP already have a checksum. Because IPv6 does not include a checksum, it is no longer optional (but is now mandatory) when being carried over IPv6.

IP is a best-effort delivery protocol; it is the responsibility of the transport layer protocols to ensure the integrity of the data. The checksums for upper-layer protocols are described in RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*.

IPv4 and IPv6 Source Address and Destination Address Fields

The most notable and recognizable difference in IPv6 compared to IPv4 is the expansion of the source and destination addresses from a 32-bit address in IPv4 to a 128-bit address in IPv6. The source and destination addresses are shown in Figure 3-14.

The main functionality of the two addresses is the same for both protocols:

- The source address is the originator of the packet.
- The source address is always a unicast address. In IPv6, this can be a link-local unicast, unique local unicast, or unspecified unicast address.
- The destination address is the recipient or final destination of the packet.
- The destination address can be a unicast, multicast, anycast, or, in the case of IPv4, broadcast address. (There is no broadcast address in IPv6.)
- The destination address is used by routers to forward the packet along its path toward its ultimate destination.

Note Network address translation (NAT) can change either the source or destination IPv4 address to one of the translator's addresses, typically an RFC 1918 private IPv4 address.

IPv4 Options and Padding Fields, IPv6 Fixed Length

The IPv4 Options field is optional, so it might or might not appear in the IPv4 packet. It is variable in size and rarely used, so it is not usually included in most IPv4 packets. The Options field might contain the record route, timestamp, and traceroute used as an enhancement to the traceroute utility and described in RFC 1393, *Traceroute Using an IP Option*.

The IPv4 Padding field is used only if the IPv4 Options field is used and the size of the IPv4 header is no longer a multiple of 32 bits. When this is the case, the IPv4 header is padded with bits that are 0s so that it ends on a 32-bit boundary.

The IPv4 Options and Padding fields are shown in Figure 3-16, along with the fixed IPv6 header.

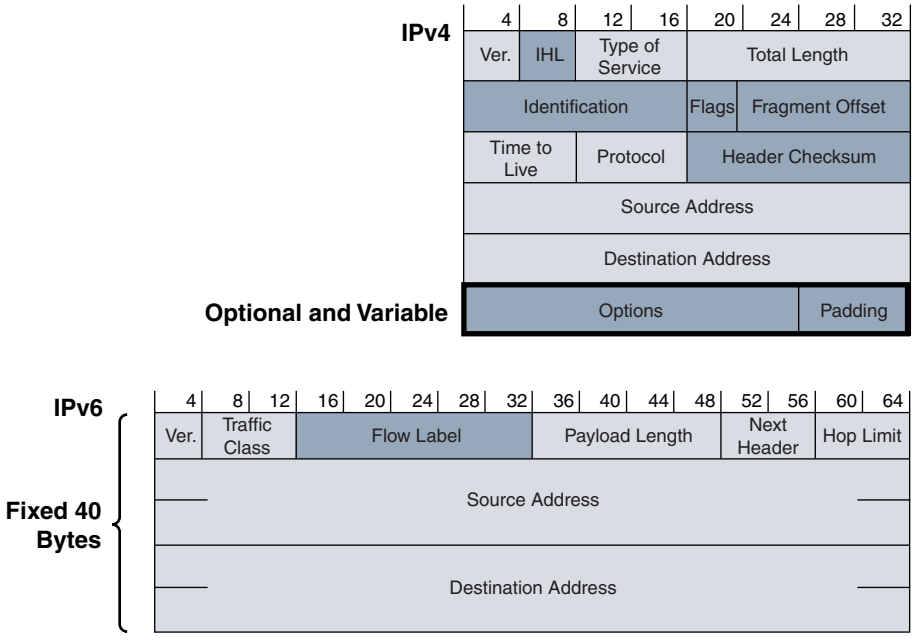


Figure 3-16 *IPv4 Options and Padding Fields and Fixed IPv6 Header*

IPv6 over Ethernet

Ethernet frames (Ethernet II frames and later, adapted for 802.3) use a 2-byte EtherType field to identify the protocol encapsulated in the data portion (payload portion) of the Ethernet frame. As shown in Figure 3-17, the EtherType field uses the hexadecimal value 86dd when the payload is an IPv6 packet. This is commonly written as 0x86dd, with the prefix 0x indicating a hexadecimal value.

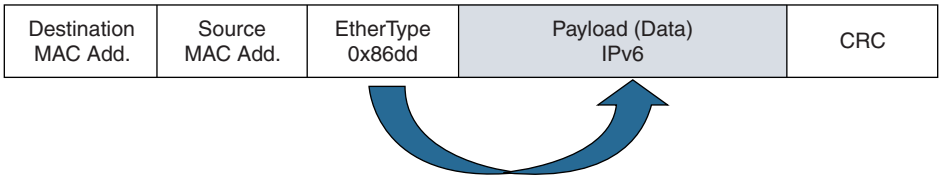


Figure 3-17 *IPv6 Packet Encapsulated in an Ethernet II Frame*

Packet Analysis Using Wireshark

To get a better view of IPv6 packets, let's take a look at one by using the packet analyzer Wireshark. Based on the network setup shown in Figure 3-18, Example 3-1 shows a simple ping from PC1 to PC2.

Note Wireshark is a network protocol analyzer for both IPv4 and IPv6. Wireshark is available for several operating systems and is available as a free download from www.wireshark.org.

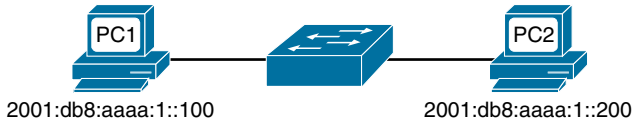


Figure 3-18 *PC1 Pinging an IPv6 Address on PC2*

Example 3-1 *Ping from PC1 to PC2*

```
PC1> ping 2001:db8:aaaa:1::200

Pinging 2001:db8:aaaa:1::200 from 2001:db8:aaaa:1::100 with 32 bytes of data:
Reply from 2001:db8:aaaa:1::200: time<1ms
Reply from 2001:db8:aaaa:1::200: time<1ms
Reply from 2001:db8:aaaa:1::200: time<1ms
Reply from 2001:db8:aaaa:1::200: time<1ms

Ping statistics for 2001:db8:aaaa:1::200:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC1>
```

Notice that some of the 0s in the addresses have been omitted in the output. IPv6 addresses and compressed formats are discussed in Chapter 4, "IPv6 Address Representation and Address Types." For now, be assured that these are the same addresses.

IPv6 addresses are 128-bit addresses and written in hexadecimal. They might look a little strange to you right now, but don't worry, you will become very familiar with them in the next few chapters. Figure 3-19 shows the Wireshark capture of the ICMPv6 Echo Request (ping), which is further described in Table 3-2. ICMPv6 is described in more detail in Chapter 12.

Note Figure 3-19 shows a Wireshark screen capture. Subsequent Wireshark captures will be displayed using Examples to make them easier to read.

In Table 3-2, notice the value 0x86dd in the Type field in the Ethernet II frame. This indicates that an IPv6 packet is the encapsulated payload (data).

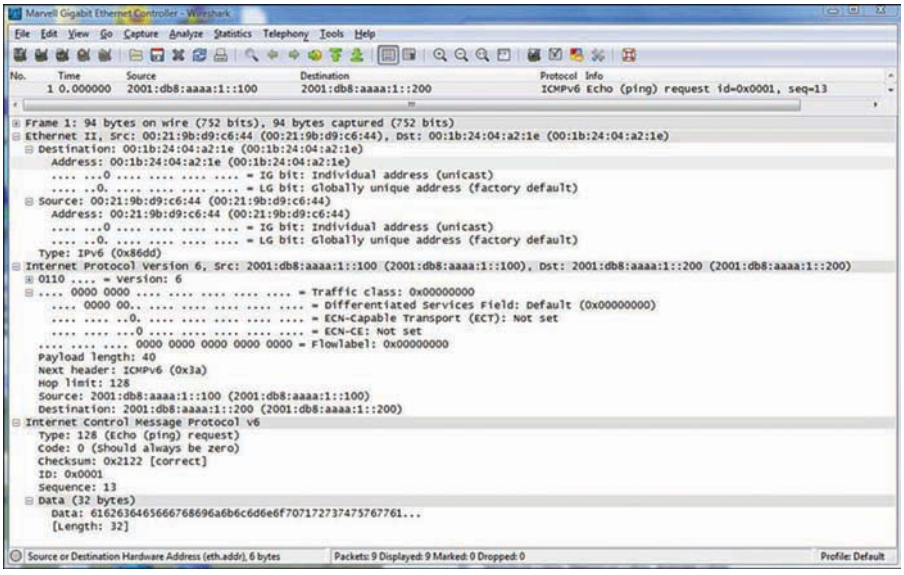


Figure 3-19 *Wireshark Capture of an IPv6 Packet*

Table 3-2 *Analysis of an Ethernet Frame Carrying an IPv6 Packet*

	Field Name	Size (Bits)	Value—Description
Ethernet II Header (16 Bytes)	Destination Address	48	**00:1b:24:04:a2:1e**—The destination MAC address in hexadecimal.
	Source Address	48	**00:21:9b:d9:c6:44**—The source MAC address in hexadecimal.
	Type	16	**0x86dd**—Indicates the protocol of the data (payload), which in this case is an IPv6 packet.
IPv6 Header (40 Bytes)	Version	4	**6**—Indicates IP version 6.
	Traffic Class	8	**0**—Default is 0.
	Flow Label	20	**0**—Default is 0.
	Payload Length	16	**40 bytes**—Indicates the size of the data, which in this case is an ICMPv6 message. Notice that the ICMPv6 message has a length of 40 bytes.
	Next Header	8	**58**—Identifies the following header as an ICMPv6 header. See Table 3-1 for a partial list of IPv6 Next Header values.

	Field Name	Size (Bits)	Value—Description
	Hop Limit	8	**128**—Indicates the maximum number of routers that this packet can traverse before being discarded.
	Source Address	128	**2001:0db8:aaaa:1::100**—The source IPv6 address in hexadecimal.
	Destination Address	128	**2001:0db8:aaaa:1::200**—The destination IPv6 address in hexadecimal.
ICMPv6 Header (40 Bytes)	Type	8	**128**—Identifies this as an ICMPv6 Echo Request message.
	Code	8	**0**—Not used; default is 0.
	Checksum	16	**0x2122**—16-bit checksum that is used to verify the integrity of the ICMPv6 header.
	ID	16	**0x0001**—Used to help match ICMPv6 Echo Request and Echo Reply messages.
	Sequence	16	**13**—Used to help match ICMPv6 Echo Request and Echo Reply messages.
	Data	256	Optional data of variable length depending upon the type of ICMPv6 message.

Extension Headers

Extension headers are an important addition to IPv6. They are optional headers that add flexibility and allow for future enhancements to IPv6. However, they can be difficult to understand. This section helps you gain a better understanding of this IPv6 feature. Some extension headers are fairly easy to grasp, while others are a little more difficult. But don't be too concerned if some of this seems a little vague. The purpose of this section is to familiarize you with the concept of what extension headers are and how they are used.

Extension headers are optional and follow the main IPv6 header. As discussed previously, the main IPv6 header includes a Next Header field, which has one of two purposes:

- To identify the protocol carried in the data portion of the packet
- To identify the presence of an extension header

The first purpose of the Next Header field is similar to that of the IPv4 Protocol field—to indicate which protocol is being carried in the data portion of the IPv6 packet (refer to Figure 3-12).

The second purpose is one of the more important additions to the IPv6 header: The Next Header field indicates an additional header known as an extension header. Immediately following the mandatory main IPv6 header, there can be zero, one, or several extension headers.

A field common in all extension headers is another Next Header field, which indicates whether another extension header follows or whether the protocol of the data (payload), like a TCP segment, follows, also shown in Figure 3-12. Therefore, the last extension header usually specifies which protocol is encapsulated as the data or payload, again, much like the Protocol field in IPv4. No Next Header is the only extension header that doesn't indicate a protocol or another extension header.

As mentioned earlier, the intention of extension headers is to provide flexibility to the main IPv6 header for future enhancements without requiring a redesign of the entire protocol. It also allows the main IPv6 header to be a fixed size for more efficient processing.

Note IPv4 uses the Options and Padding fields to accomplish what the extension headers do in IPv6. Unfortunately, that makes the IPv4 header variable length, which results in additional processing.

Remember that the designers of IPv6 wanted to optimize the protocol for 64-bit CPUs. Much like the main IPv6 header, extension headers must end on a 64-bit boundary. Most extension headers are of fixed length and multiples of 64 bits. If an extension header is variable in length, then it must use padding to ensure that it ends on a 64-bit boundary.

Table 3-3 summarizes the six extension headers that currently exist, as defined in RFC 2460. Figure 3-20 shows an example of an IPv6 packet using two extension headers. A brief description of Figure 3-20 is as follows:

- The main IPv6 header has all the fields discussed so far, including the source and destination addresses and a Next Header field. The Next Header field has the value 0, indicating that a Hop-by-Hop extension header immediately follows the main IPv6 header.
- The Hop-by-Hop extension header follows the main IPv6 header. Extension headers are explained in more detail in the next section, but for now, notice that the Hop-by-Hop extension header also contains its own Next Header field. In this example, the value 51 signifies that there is yet another extension header to follow, the Authentication Header (AH).
- The final extension header is the Authentication Header. Its Next Header field has a value of 6, indicating that a TCP upper-layer header is to follow. This also means that there are no more extension headers in this packet.

Table 3-3 *IPv6 Extension Headers*

Next Header Value (Decimal)	Extension Header Name	Extension Header Length (Bytes)	Variable-Length Options (TLV) Used?	Extension Header Description
0	Hop-by-Hop Options	Variable	Yes	Used to carry optional information, which must be examined by every router along the path of the packet.
43	Routing	Variable	No	Allows the source of the packet to specify the path to the destination.
44	Fragment	8	No	Used to fragment IPv6 packets.
50	Encapsulating Security Payload (ESP)	Variable	No	Used to provide authentication, integrity, and encryption.
51	Authentication Header (AH)	Variable	No	Used to provide authentication and integrity.
60	Destination Options	Variable	Yes	Used to carry optional information that only needs to be examined by a packet's destination node(s).

Note The Next Header field is used to chain together multiple IPv6 headers and the data portion of the packet at the end of the chain.

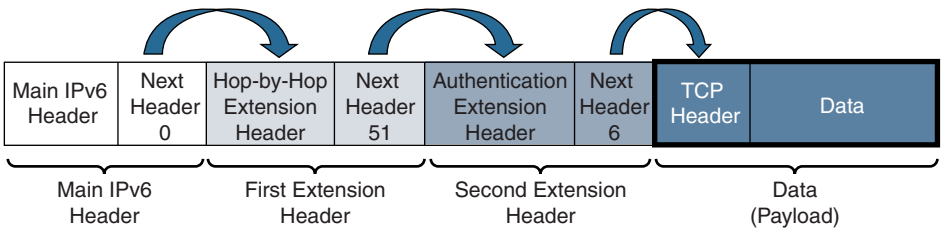


Figure 3-20 *Use of the Next Header Field in Extension Headers*

RFC 2460 recommends that when multiple extension headers are used in the same packet, those headers should appear in the following order:

1. Main IPv6 header
2. Hop-by-Hop Options header
3. Destination Options header
4. Routing header
5. Fragment header
6. Authentication Header (AH)
7. Encapsulating Security Payload (ESP) header
8. Destination Options header
9. Upper-layer protocol header

Note Extension headers are an important part of IPv6. They add flexibility and allow for future enhancements to IPv6. Extension headers enable IPv6 to do things not previously available in IPv4. But they are optional and used only in very specific circumstances. If you are new to IPv6, you might want to just scan the rest of this section on extension headers for now or use it as reference later. Extension headers are unique to IPv6, and it is important to be familiar with them. However, not having an in-depth understanding of them will not deter you from understanding the other topics discussed in this book.

Hop-by-Hop Options Extension Header

The Hop-by-Hop Options header is used to carry optional information that should be examined by every router along the path of the packet. (Previously it was mandatory, but it has been changed so that not every node must look at this header.) The Hop-by-Hop Options header is one of two extension headers that contain a variable-length options field similar to IPv4. As the name implies, this type of option should be examined by every router, at every hop along the path.

Note Destination Options is the other extension header that uses options. As its name implies, it contains information intended only for the final destination. Destination Options is discussed at the end of this section.

Options provide flexibility, allowing IPv6 packets to be supplemented with sets of values that are not defined in the standard group of extension headers. These sets of values are also referred to as Type-Length-Value (TLV) triplets. It has already been established that two extension headers use these options: Hop-by-Hop Options and Destination Options.

As shown in Figure 3-21, these two types of extension headers have a Next Header field and a Header Extension Length field, followed by one or more sets of options. Each option contains a set of Options Type, Options Length, and Options Data fields (Type, Length, Value [TVL]).

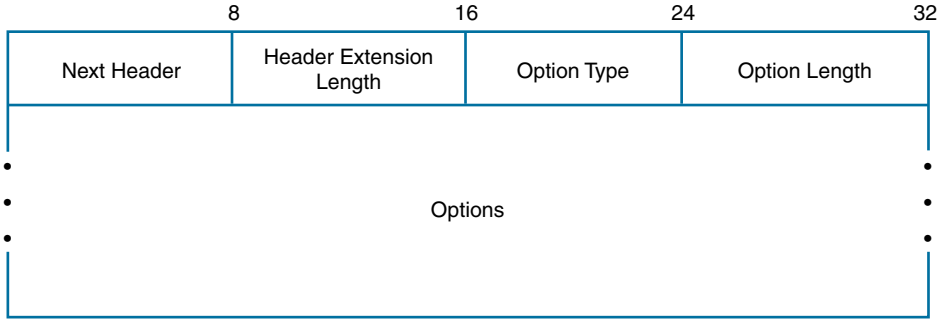


Figure 3-21 *Extension Header Options*

Figure 3-22 shows a Hop-by-Hop extension header referencing a Jumbo Payload Option. The Jumbo Payload Option is used to indicate that the size of this IPv6 packet, a jumbogram, is larger than 65,535 bytes. Because it is a Hop-by-Hop Option, this information is to be examined by each router along the path.

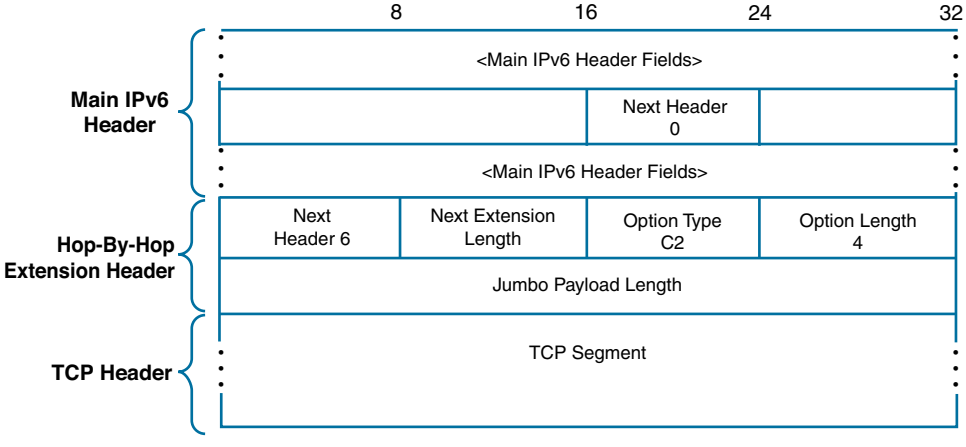


Figure 3-22 *Hop-by-Hop Extension Header with a Jumbo Payload Option*

The following describes the fields associated with the Hop-by-Hop extension header:

- **The main IPv6 header: Next Header (8 bits):** Along with other information in the main IPv6 header, there is a Next Header field with the value 0. This indicates that a Hop-by-Hop Options extension header follows this main header.

- **The Hop-by-Hop extension header:** This extension header contains the following:
  - **Next Header (8 bits):** The value 6 in the Next Header field indicates that a TCP header follows this header and that there are no other extension headers.
  - **Header Extension Length (8 bits):** This is the length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets. There can be multiple options, with each option consisting of a TLV (Options Type, Options Length, and Options Data fields).
  - **Option Type (8 bits):** This is the type of option carried in this header. The hexadecimal value C2 indicates that this is a Jumbo Payload Option.
  - **Option Data Length (8 bits):** This is the number of bytes in the Option Data field. The value 4 indicates that the Option Data field is 4 bytes (32 bits) long.
  - **Option Data (variable-length):** The data in this example is the Jumbo Payload Length. Jumbo Payload Length is a 32-bit field that indicates the size of the IPv6 packet in bytes, excluding the IPv6 header but including the Hop-by-Hop Options header and any other extension headers present. Jumbo Payload Length must be greater than 65,535 and can be up to 4,294,967,295 bytes.
- **TCP Segment:** Because there is only one option and no other extension headers, a TCP segment follows, as indicated by the Next Header value 6 in the previous Hop-by-Hop extension header.

If a Hop-by-Hop Options extension header is used, it immediately follows the main IPv6 header. We have only discussed the use of the Jumbo Payload Length as an example of using the Hop-by-Hop extension header. The Hop-by-Hop extension header may contain the following options:

- **Pad1:** Provides 1 byte of padding.
- **PadN:** Used when 2 or more bytes of padding are required.
- **Jumbo Payload Length:** Indicates that the size of this IPv6 packet, a jumbogram, is larger than 65,535 bytes.
- **Router Alert:** Informs routers to examine the contents of an IPv6 packet closely. This option is useful for situations in which a packet contains information that requires special processing by routers along the path.

Routing Extension Header

The Routing extension header allows the source of the packet to specify the path to the destination. This header contains a list of one or more intermediate routers on the way to a packet's destination. This function is very similar to the Loose Source option used in IPv4. The Routing header is identified by a Next Header value of 43 in the immediately preceding header.

There are currently four types of Routing headers, along with a type that has been deprecated:

- **Type 0:** Deprecated due to security concerns.
- **Type 1:** Used for Nimrod, a project funded by DARPA.
- **Type 2:** Used for Mobile IPv6.
- **Type 3:** Used to assign a routing algorithm and parameters applied to an anycast packet.
- **Type 4:** Used for segment routing.

Note Due to a number of potential vulnerabilities, including network discovery to DoS attacks, the Type 0 Routing extension header was deprecated in RFC 5095, *Deprecation of Type 0 Routing Headers in IPv6*.

Figure 3-23 shows the structure of a Type 2 Routing header used for mobility support in IPv6. This extension header allows the packet to be routed directly from a correspondent to the mobile node's care-of address, which provides information about the mobile node's current location.

The following fields are associated with the Routing extension header:

- **Next Header (8 bits):** Identifies the type of header immediately following the Routing header. It is either another extension header or the protocol of the payload.
- **Header Extension Length (8 bits):** This is the length of the Routing header in 8-octet units, not including the first 8 octets.
- **Routing Type (8 bits):** This value is 2.
- **Segments Left (8 bits):** This value is 1.
- **Reserved (32 bits):** This field is reserved. It is initialized to 0 for transmission and ignored on reception.
- **Home Address (128 bits):** This is the home address of the destination mobile node. This field is specific to mobile IPv6. See RFC 3775, *Mobility Support in IPv6*, for more information.

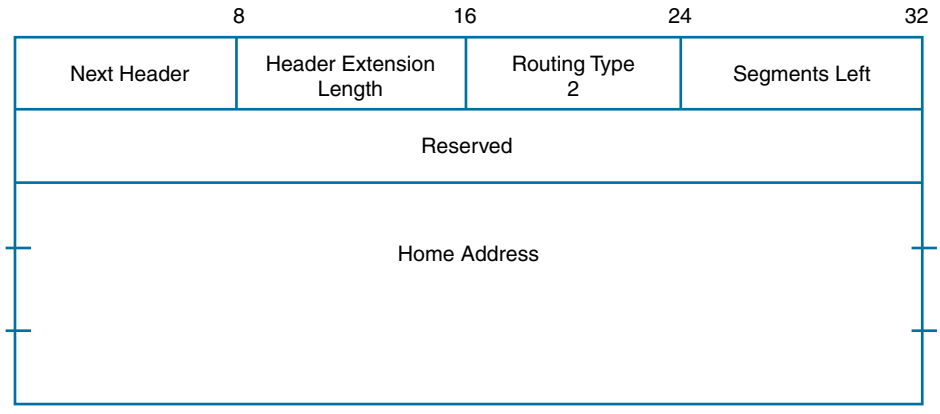


Figure 3-23 *Type 2 Routing Header*

Note The details of how the Routing extension header is processed by each router are beyond the scope of this book. If you are interested, refer to RFC 2460 or the Cisco Press book *Cisco Self-Study: Implementing Cisco IPv6 Networks*. Many ISPs cast a suspicious eye if the source node offers to help with packet next-hop selection, and it is common for packets containing routing extensions to be administratively blocked. Currently the only valid routing header is a Type 2 header, which is used for mobility in IPv6.

Fragment Extension Header

The Fragment extension header, as shown in Figure 3-24, is similar to the fields in the IPv4 header used for fragmentation. It is used when the source of the IPv6 packet needs to divide the packet into fragments and send each fragment as a separate packet. The recipient of the packet reassembles the fragments, each with its own main IPv6 header and a Fragment extension header.

Unlike in IPv4, an IPv6 router does not fragment a packet unless it is the source of the packet. Intermediate nodes do not perform fragmentation. Only the source of the packet can fragment it. If a router receives an IPv6 packet that is larger than the MTU of the egress interface, the router discards the packet and sends an ICMPv6 Packet Too Big error message back to the source.

Similar to IPv4, in IPv6, for every packet to be fragmented, the source generates a unique Identification value. This value is included in each of the fragment packets. The Identification value ensures that fragments from the original packet are properly reassembled. If the source needs to fragment additional packets within the same message, different Identification values are used.

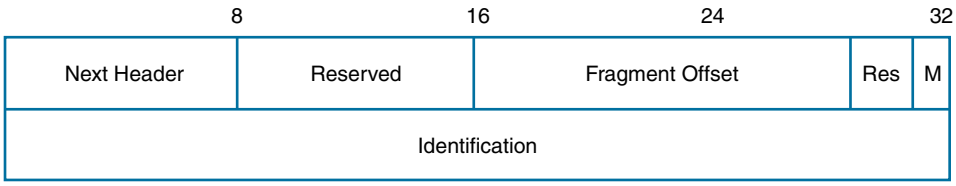


Figure 3-24 *Fragment Header*

The following fields are part of the Fragment header, as shown in Figure 3-24:

- **Next Header (8 bits):** Identifies the protocol number of the data, the fragmented part of the original packet.
- **Reserved (8 bits):** This field is reserved. It is initialized to 0 for transmission and ignored on reception.
- **Fragment Offset (13 bits):** This is the relative offset or position, in 8-octet units, of the fragmented data following this header, relative to the original packet. Much like the Fragment Offset field in IPv4, this field informs the receiver where to align the fragmented packet in relation to the other fragmented packets.
- **Res (2 bits):** This field is reserved. It is initialized to 0 for transmission and ignored on reception.
- **M flag (1 bit):** The M, or More Fragments, flag is used to indicate whether this is the last fragment (0) or whether there are more fragments to follow (1). This is similar to IPv4's More Fragments flag.
- **Identification (32 bits):** Similar to the same field in the IPv4 header, the Identification field is used to uniquely identify all fragmented packets within the same original packet. This field has been expanded from 16 bits in IPv4 to 32 bits in IPv6.

IPsec: AH and ESP Extension Headers

The following extension headers are used to implement two core security protocols in IPsec:

- Authentication Header (AH)
- Encapsulating Security Payload (ESP)

Before examining the AH and ESP extension headers, this section briefly discusses IPsec and the functions of these two security protocols. It is not meant to give you a thorough understanding of IPsec, AH, and ESP, but it will give you enough information to understand their importance and use in IPv6.

IPsec is a suite of protocols for securing delivery of packets in IP networks. Authentication Header (AH) and Encapsulating Security Payload (ESP) are the two main protocols used to provide authentication and integrity for all or part of an IPv6 packet. ESP includes the additional feature of encryption.

Note IPsec is part of both IPv4 and IPv6. IPsec is not required for devices that implement the IPv4 stack. Earlier RFCs state that IPsec is mandatory for all IPv6 implementations and that IPsec "must be supported." RFC 6434, *IPv6 Node Requirements*, relaxes this requirement, declaring that IPsec "should be implemented."

The Authentication Header (AH) is used to guarantee the authenticity and integrity of a packet. *Authentication* means that it ensures that the sender and receiver of the packet are really who they say they are. *Integrity* guarantees that the data was not altered in transit. AH provides authentication and integrity but does not provide encryption. *Encryption* is the process of transforming information (usually in plain text) by using an algorithm (called a cipher) to make it unreadable to anyone except those possessing special information (usually referred to as a key).

The Encapsulating Security Payload (ESP) provides authentication, integrity, *and* encryption. ESP not only protects a packet from being altered by intermediate devices but also protects the content of the packet from being viewed. ESP has its own authentication scheme, or it can be used in conjunction with AH. In summary, AH provides only authentication and integrity, while ESP provides both of these and encrypts the packet as well. But how much of a packet is authenticated or encrypted? The answer to this question depends on whether transport mode or tunnel mode is used in IPsec, as discussed next.

Transport and Tunnel Modes

As the name suggests, transport mode protects the transport layer and higher. The initial IP header is still used. Because the original source and destination IP addresses are in the main IP header intermediate devices, routers recognize IPsec participants. Transport mode is typically used for host-to-host communications.

Tunnel mode is employed to protect the entire contents of the IP packet, including the IP header. This is accomplished by encapsulating the original IP packet, including the IP header, in a new IP header with the tunnel endpoints used as the new source and destination IP addresses. The tunnel endpoints can be routers or can be the hosts themselves. Tunnel mode protects the entire IP packet, while transport mode does not. Figure 3-25 illustrates the differences between transport and tunnel modes.

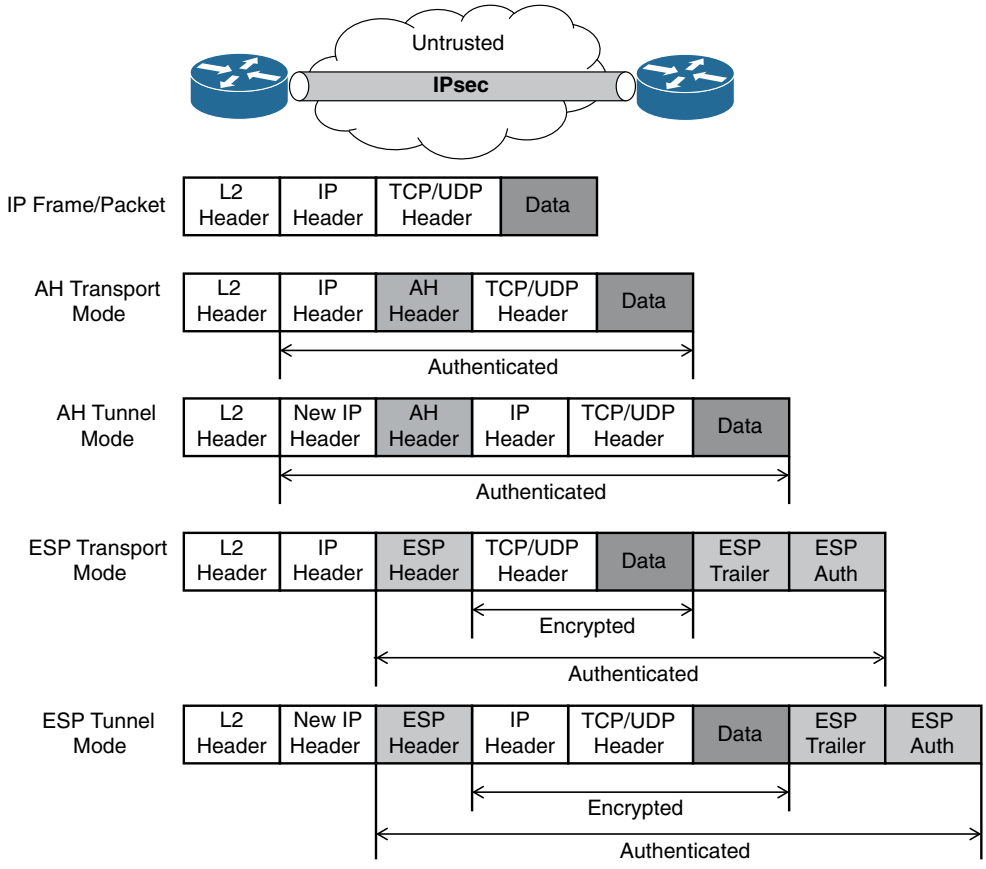


Figure 3-25 *Transport and Tunnel Modes*

Let's return to the extension headers used for AH and ESP. If you are new to IPsec, it is perfectly understandable if some of the details of AH and ESP are not yet perfectly clear.

Encapsulating Security Payload (ESP) Extension Header

The Encapsulating Security Payload (ESP) is a variable-length extension header. As discussed previously, this header is used to provide authentication, integrity, and encryption. The ESP extension header is identified by a Next Header value of 50 in the immediately preceding header.

Figure 3-26 shows the ESP extension header, which can be divided into four parts:

- **ESP Header:** SPI and Sequence Number fields
- **Payload:** ESP Payload Data field
- **ESP Trailer:** Padding, Pad Length, and Next Header fields
- **ESP Authentication Data**

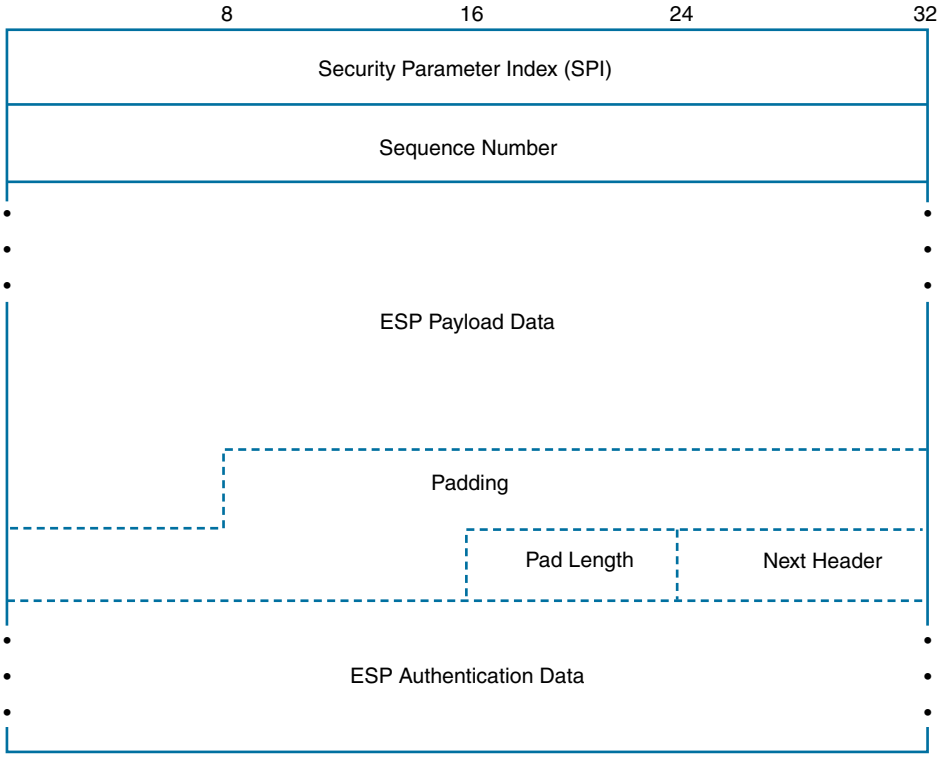


Figure 3-26 *ESP Extension Header*

Figure 3-26 illustrates the fields in the ESP extension header. ESP is viewed as end-to-end communications; in other words, it is not processed by routers along the path. Remember that ESP provides authentication, integrity, and confidentiality of the original packet. Therefore, the ESP extension header is encapsulated after the main IPv6 header and after the Hop-by-Hop, Routing, and Fragment extension headers, as shown in Figure 3-27. For IPv6, encryption covers the entire transport-level segment plus the ESP trailer and the Destination Options extension header, if it occurs after the ESP header. The Destination Options extension header could appear before the ESP, after the ESP header, or both. ESP and AH headers can be combined in a variety of modes. These options are described in RFC 4301, *Security Architecture for the Internet Protocol*.

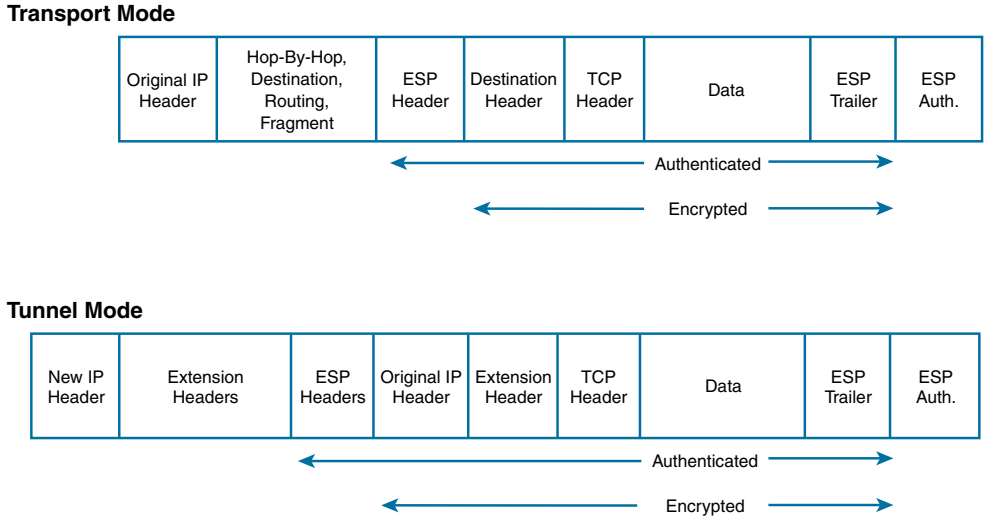


Figure 3-27 *ESP—Transport and Tunnel Modes*

The details of each field are beyond the scope of this book and require a better understanding of IPsec. If you are new to IPsec, it is understandable if some of this is a little vague. For more information about IPsec, the Cisco Press book *IPsec Virtual Private Network Fundamentals*, by James Henry Carmouche, is an excellent resource.

Authentication Header (AH) Extension Header

The Authentication Header (AH) is also a variable-length extension header. Unlike ESP, AH provides only authentication and integrity and does not use encryption to provide confidentiality. The AH extension header is identified by a Next Header value of 51 in the preceding header.

Figure 3-28 illustrates the fields in the AH extension header. Like ESP, AH is viewed as end-to-end communications. Remember that AH only provides data integrity, a feature to ensure that the participants are who they say they are, and that any alteration in the packet's content in transit is detected by the recipient. Much like ESP, the AH extension header is encapsulated after the main IPv6 header and after the Hop-by-Hop, Routing, and Fragment extension headers, as shown in Figure 3-29.

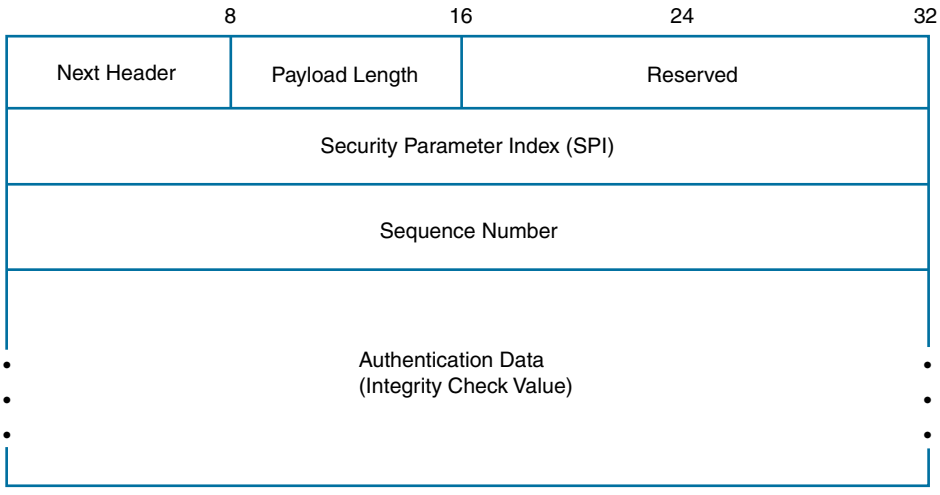


Figure 3-28 *AH Extension Header*

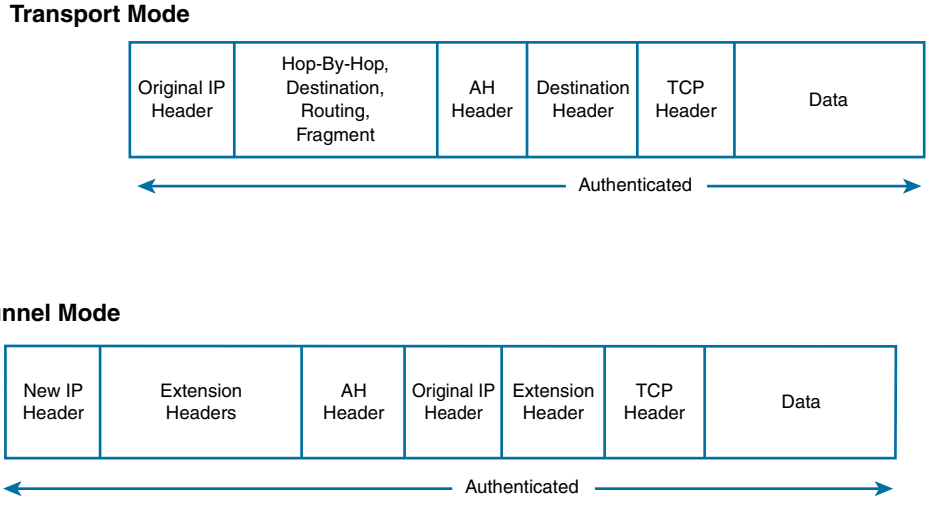


Figure 3-29 *AH—Transport and Tunnel Modes*

As with the other IPsec extension header, the details of AH and these fields are beyond the scope of this book and require a more thorough understanding of IPsec.

Destination Options Extension Header

The Destination Options header is used to carry optional information that only needs to be examined by a packet's destination node(s). The Destination Options header is the other extension header that uses options besides the Hop-by-Hop header.

The Destination Options header is identified by a Next Header value of 60 in the preceding header. As illustrated in Figure 3-30, the Destination Options header has the following format:

- **Next Header (8 bits):** Identifies the type of header immediately following the Destination Options header. This is either another extension header or the protocol of the payload.
- **Header Extension Length (8 bits):** This is the length of the Destination Options header in 8-octet units, not including the first 8 octets.
- **Options (variable-length):** This field contains one or more TLV-encoded options:
  - **Option Type (8 bits):** This is the type of option carried in this header.
  - **Option Data Length (8 bits):** This signifies the number of bytes of the Option Data field.
  - **Option Data (variable-length):** This is the data content, depending upon the use of this field.

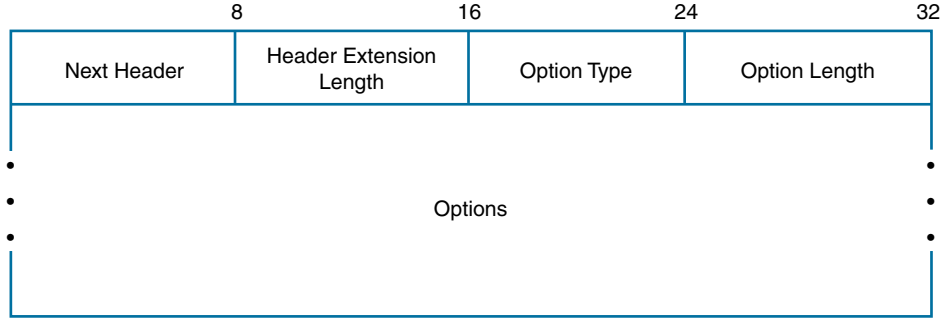


Figure 3-30 *Destination Options Extension Header*

Note One of the proposed uses for the Destination Options extension header is mobility support, as described in RFC 6275, *Mobility Support in IPv6*:

"Each mobile node is always identified by its home address, regardless of its current point of attachment to the Internet. While situated away from its home, a mobile node is also associated with a care-of address, which provides information regarding the mobile node's current location. IPv6 packets addressed to a mobile node's home address are transparently routed to its care-of address. The protocol enables IPv6 nodes to cache the binding of a mobile node's home address with its care-of address, and to then send any packets destined for the mobile node directly to it at this care-of address. To support this operation, Mobile IPv6 defines a new IPv6 protocol and a new destination option. All IPv6 nodes, whether mobile or stationary, can communicate with mobile nodes."

No Next Header

The Next Header value 59 indicates that there is no data following this header. This is just a placeholder indicating that there is nothing after this header. If the payload length indicates that there are additional bytes beyond the header, those bytes are ignored.

Comparing IPv4 and IPv6 at a Glance

When examining the details of the IPv4 and IPv6 headers, it's easy to miss some of the important differences between the two protocols. This chapter has provided a lot of information that may be difficult to digest, so this section summarizes some of the differences. As you read this section, refer to Figures 3-1 and 3-2.

These IPv4 field names are the same as those in IPv6:

- **Version (IPv4 and IPv6):** This is an easy one; the value is 4 in IPv4 and 6 in IPv6.
- **Source Address and Destination Address (IPv4 and IPv6):** Probably the most noticeable differences are the 32-bit IPv4 source and destination addresses, which have been increased to 128 bits in IPv6.

Some IPv4 field names were changed for IPv6, and in some cases the fields have functional differences:

- **Type of Service (IPv4) has been changed to Traffic Class (IPv6):** IPv4 can use either the 3-bit IP Precedence field along with another 3 bits for delay, throughput, and reliability, or the 6-bit Differentiated Services technique. IPv6 was designed to use the 6-bit Differentiated Services method.
- **Total Length (IPv4) has been changed to Payload Length (IPv6):** IPv4's Total Length field includes both the IPv4 header and the data, whereas the IPv6 Payload Length field specifies only the number of bytes of data (payload), including any extension headers, and does not include the main IPv6 header. Using the Jumbo Payload option in the Hop-by-Hop extension header, IPv6 provides jumbograms, which are IPv6 packets up to 4,294,967,295 (2^{32}–1) bytes.
- **Time to Live (IPv4) has been changed to Hop Limit (IPv6):** The IPv4 Time to Live (TTL) and IPv6 Hop Limit fields ensure that packets do not transit between networks for an indefinite period of time. These fields have the same function, with the name being more reflective of the field's actual use in IPv6.
- **Protocol (IPv4) has been changed to Next Header (IPv6):** In IPv4, this indicates the protocol being carried in the IPv4 data or payload. This same function exists in the Next Header field in IPv6 but can also indicate the existence of an extension header following the main IPv6 header.

The following fields that exist in IPv4 have been removed from IPv6:

- **Internet Header Length (IPv4):** This field is not needed in IPv6 because the main IPv6 header has a fixed length of 40 bytes. Any additional IPv6 headers are linked as indicated in the Next Header field and known as extension headers.
- **Identification (IPv4), Flags (IPv4), and Fragment Offset (IPv4):** These fields are used for fragmentation in IPv4. Fragmentation is only performed at the source in IPv6, using the Fragment extension header, and not by routers.
- **Header Checksum (IPv4):** Layer 2 technologies such as Ethernet perform their own checksum and error control. Upper-layer protocols such as TCP and UDP also have their own checksums, and therefore a checksum at Layer 3 is redundant and unnecessary. A UDP checksum, which is optional in IPv4, is mandatory in IPv6.
- **Options (IPv4):** Options in IPv4 are handled using extension headers in IPv6. Two IPv6 extension headers, Hop-by-Hop Options and Destination Options, contain their own sets of TLV options.
- **Padding (IPv4):** Because IPv6 has a fixed length of 40 bytes, it is unnecessary to extend the main IPv6 header to make sure that it falls on a 32-bit boundary. The main header falls on a 64-bit boundary.

The following are new fields in IPv6 that don't exist in IPv4:

- **Flow Label (IPv6):** This is a new field to IPv6, and the specifications of its use are still being determined by the IETF. RFC 2460 discusses using the Flow Label field to label sequences of packets that need special handling by IPv6 routers for "real-time" service. RFC 6437 contains additional details on the Flow Label field.

The following are new headers in IPv6 that don't exist in IPv4:

- **Extension Headers (IPv6):** IPv6 extension headers are used to provide flexibility and future enhancements to a fixed IPv6 header.

Other features:

- **IPv6 over Ethernet:** Ethernet II frames use the EtherType field with the hexadecimal value 86dd (0x86dd) to indicate that the payload is an IPv6 packet.
- **Fragmentation:** IPv6 routers do not perform fragmentation. When an IPv6 router receives a packet larger than the MTU of the egress interface, the router drops the packet and sends an ICMPv6 Packet Too Big message back to the source, including the MTU size of the egress link in bytes. Fragmentation can be done in IPv6 only by the source device, and it is handled using a Fragment extension header.

Summary

This chapter examines both the IPv4 and IPv6 headers. It looks at the similarities and differences between the two protocols. The IPv6 header has fewer fields, and in many respects, IPv6 is a simpler protocol. Some of the fields in IPv6 remain the same as those in IPv4, some have experienced name changes with functional differences, others have been removed completely, and a new Flow Label field has been added.

This chapter introduces extension headers, which provide more flexibility and better efficiency for IPv6. This chapter also talks about the impact of IPv6 on User Datagram Protocol (UDP) and maximum transmission unit (MTU).

The final section of this chapter provides an overview of the similarities and differences between the two protocols.

Review Questions

1. Why doesn't the IPv6 header include a field similar to the IHL in IPv4?
2. What is the difference between the IPv4 Total Length field and the IPv6 Payload Length field?
3. What is the most notable difference between the source and destination addresses in IPv6 compared to IPv4?
4. What is the difference between how IPv4 and IPv6 perform fragmentation?
5. Unlike the IPv4 header, the IPv6 header does not include a Checksum field. What effect does this have on any transport layer protocols?
6. Why does IPv6 not contain the Options and Padding fields? What does IPv6 use instead of these fields?
7. What is the value of the EtherType field in an Ethernet II frame when the payload is an IPv6 packet?

References

RFCs

RFC 791, *Internet Protocol, DARPA Internet Program Protocol Specification*, USC, www.ietf.org/rfc/rfc791.txt, September 1981.

RFC 1191, *Path MTU Discovery*, J. Mogul, Stanford University, www.ietf.org/rfc/rfc1191.txt, November 1990.

RFC 1393, *Traceroute Using an IP Option*, G. Malkin, Xylogics, Inc., www.ietf.org/rfc/rfc1393.txt, January 1993.

RFC 1700, *Assigned Numbers*, J. Reynolds, ISI, www.ietf.org/rfc/rfc1700.txt, October 1994.

RFC 1981, *Path MTU Discovery for IP Version 6*, J. McCann, Digital Equipment Corporation, www.ietf.org/rfc/rfc1981.txt, August 1996.

RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, Cisco Systems, www.ietf.org/rfc/rfc2460.txt, December 1998.

RFC 2474, *Using a Technique Called Differentiated Services (DS)*, K. Nichols, Cisco Systems, www.ietf.org/rfc/rfc2474.txt, December 1998.

RFC 2675, *IPv6 Jumbograms*, D. Borman, Berkeley Software Design, www.ietf.org/rfc/rfc2675.txt, August 1999.

RFC 3775, *Mobility Support in IPv6*, D. Johnson, Rice University, www.ietf.org/rfc/rfc3775.txt, June 2004.

RFC 6434, *IPv6 Node Requirements*, E. Jankiewicz, SRI International, www.ietf.org/rfc/rfc6434.txt, December 2011.

RFC 6437, *IPv6 Flow Label Specification*, S. Amante, Level 3, www.ietf.org/rfc/rfc6437.txt, November 2011.

Websites

IANA's list of protocol numbers, www.iana.org/assignments/protocol-numbers/protocol-numbers.xml

Wireshark, www.wireshark.org

This page intentionally left blank

Part II

IPv6 Addresses

This page intentionally left blank

Chapter 4

IPv6 Address Representation and Address Types

The most obvious and recognizable difference between IPv4 and IPv6 is the IPv6 address. An IPv4 address is 32 bits and expressed in dotted-decimal notation, whereas an IPv6 address is 128 bits in length and written in hexadecimal. However, there are many other differences between the two protocol addresses. IPv6 includes new address types as well as changes to familiar address types.

In this chapter, you will become familiar with reading IPv6 addresses. You will also learn how to represent many IPv6 addresses with fewer digits, using two simple rules.

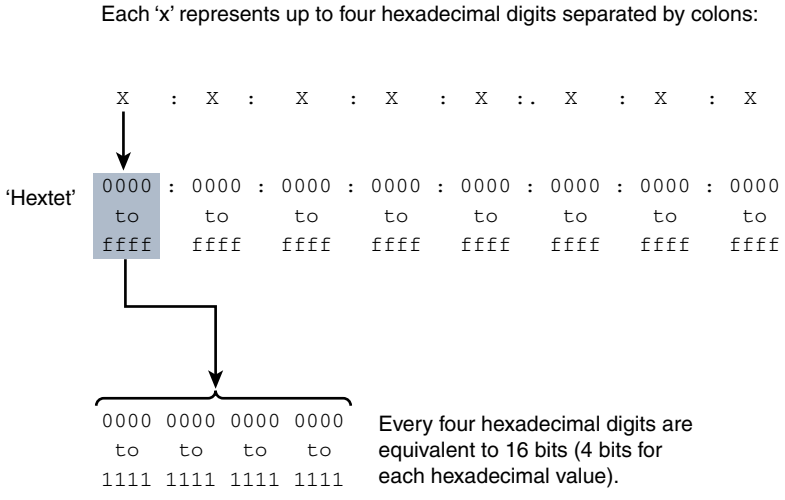
This chapter examines all the different types of IPv6 addresses in the unicast, multicast, and anycast categories. Some addresses, such as global unicast, link-local unicast, and multicast addresses, have more significance in IPv6. These addresses are examined more closely in Chapter 5, "Global Unicast Address," Chapter 6, "Link-Local Unicast Address," and Chapter 7, "Multicast Addresses."

Representation of IPv6 Addresses

IPv6 addresses are 128 bits in length and written as a string of hexadecimal digits. Every 4 bits can be represented by a single hexadecimal digit, for a total of 32 hexadecimal values (0_{16} [0000_2] through f_{16} [1111_2]). You will see later in this section how to possibly reduce the number of digits required to represent an IPv6 address. The alphanumeric characters used in hexadecimal are not case sensitive; therefore, uppercase and lowercase characters are equivalent. Although IPv6 address can be written in lowercase or uppercase, RFC 5952, *A Recommendation for IPv6 Address Text Representation*, recommends that IPv6 addresses be represented in lowercase.

Note If you are new to the hexadecimal number system, see Chapter 2, "IPv6 Primer," for information on this number system.

As described in RFC 4291, the preferred form is `x:x:x:x:x:x:x:x`. Each *x* is a 16-bit section that can be represented using up to four hexadecimal digits, with the sections separated by colons. The result is eight 16-bit sections, or hextets, for a total of 128 bits in the address, as shown in Figure 4-1. Figure 4-1 also shows an example of IPv6 addresses on a Windows host and a Mac OS host. These addresses and the format of these addresses will be explained in this chapter.




```
Windows-OS> ipconfig

Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086  ! IPv6 GUA
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11         ! IPv6 Link-Local
   IPv4 Address. . . . . . . . . . . : 192.168.1.100
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::1%11                           ! IPv6 Default Gateway
                                       192.168.1.1
-------------------------------------------------------------------------------------------------
Mac-OS$ ifconfig

en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
          ether 60:33:4b:15:24:6f
          inet6 fe80::6233:4bff:fe15:246f%en1 prefixlen 64 scopeid 0x5          ! IPv6 Link-Local
          inet 192.168.1.111 netmask 0xffffff00 broadcast 192.168.1.255
          inet6 2001:db8:cafe:1:4bff:fe15:246f prefixlen 64 autoconf            ! IPv6 GUA
          Graphics: autoselect
          status: active
```

Figure 4-1 *Preferred Form of IPv6 Address*

The longest representation of the preferred form includes a total of 32 hexadecimal values. Colons separate the groups of 4-bit hexadecimal digits.

The unofficial term for a section of four hexadecimal values is a *hextet*, similar to the term *octet* used in IPv4 addressing. An IPv6 address consists of eight hextets separated by colons. As Figure 4-1 illustrates, each hextet, with its four hexadecimal digits, is

equivalent to 16 bits. For clarity, the term *hextet* is used throughout this book when referring to individual 16-bit segments. The following list shows several examples of IPv6 addresses using the longest representation of the preferred form:

```
0000:0000:0000:0000:0000:0000:0000:0000
0000:0000:0000:0000:0000:0000:0000:0001
ff02:0000:0000:0000:0000:0000:0000:0001
fe80:0000:0000:0000:a299:9bff:fe18:50d1
2001:0db8:1111:000a:00b0:0000:9000:0200
2001:0db8:0000:0000:abcd:0000:0000:1234
2001:0db8:cafe:0001:0000:0000:0000:0100
2001:0db8:cafe:0001:0000:0000:0000:0200
```

At first glance, these addresses can look overwhelming. Don't worry, though. Later in this chapter, you will learn a technique that helps in reading and using IPv6 addresses. RFC 2373 and RFC 5952 provide two helpful rules for reducing the notation involved in the preferred format, which will be discussed next.

Rule 1: Omit Leading 0s

One way to shorten IPv6 addresses is to omit leading 0s in any hextet (that is, 16-bit section). This rule applies only to leading 0s and not to trailing 0s; being able to omit both leading and trailing 0s would cause the address to be ambiguous. Table 4-1 shows a list of preferred IPv6 addresses and how the leading 0s can be removed. The preferred form shows the address using 32 hexadecimal digits.

Table 4-1 *Examples of Omitting Leading 0s in a Hextet**

Format	IPv6 Address
Preferred	0000:0000:0000:0000:0000:0000:0000:0000
Leading 0s omitted	0: 0: 0: 0: 0: 0: 0: 0 or 0:0:0:0:0:0:0:0
Preferred	0000:0000:0000:0000:0000:0000:0000:0001
Leading 0s omitted	0: 0: 0: 0: 0: 0: 0: 1 or 0:0:0:0:0:0:0:1
Preferred	ff02:0000:0000:0000:0000:0000:0000:0001
Leading 0s omitted	ff02: 0: 0: 0: 0: 0: 0: 1 or ff02:0:0:0:0:0:0:1

Format	IPv6 Address
Preferred	fe80:**000**0:**000**0:**000**0:a299:9bff:fe18:50d1
Leading 0s omitted	fe80: 0: 0: 0:a299:9bff:fe18:50d1 or fe80:0:0:0:a299:9bff:fe18:50d1
Preferred	2001:**0**db8:1111:**000**a:**00**b0:**000**0:9000:**0**200
Leading 0s omitted	2001: db8:1111: a: b0: 0:9000: 200 or 2001:db8:1111:a:b0:0:9000:200
Preferred	2001:**0**db8:**000**0:**000**0:abcd:**000**0:**000**0:1234
Leading 0s omitted	2001: db8: 0: 0:abcd: 0: 0:1234 or 2001:db8:0:0:abcd:0:0:1234
Preferred	2001:**0**db8:aaaa:**000**1:**000**0:**000**0:**000**0:**0**100
Leading 0s omitted	2001: db8:aaaa: 1: 0: 0: 0: 100 or 2001:db8:aaaa:1:0:0:0:100
Preferred	2001:**0**db8:aaaa:**000**1:**000**0:**000**0:**000**0:**0**200
Leading 0s omitted	2001: db8:aaaa: 1: 0: 0: 0: 200 or 2001:db8:aaaa:1:0:0:0:200

* In this table, the 0s to be omitted are in bold. Spaces are retained so you can better visualize where the 0s were removed.

It is important to remember that only leading 0s can be removed; if you deleted trailing 0s the address would be incorrect. To ensure that there is only one correct interpretation of an address, only leading 0s can be omitted, as shown in the following example:

- 0s omitted:

  2001:db8:100:a:0:bc:abcd:d0b

- Incorrect (trailing 0s):

  2001:db8**0**:100**0**:a**000**:0**000**:bc**00**:abcd:d0b**0**

- Correct (leading 0s):

  2001:**0**db8:**0**100:**000**a:**000**0:**00**bc:abcd:**0**d0b

Rule 2: Omit All-0s Hextets

The second rule for shortening IPv6 addresses is that you can use a double colon (::) to represent any single, contiguous string of two or more hextets (16-bit segments) consisting of all 0s. Table 4-2 illustrates the use of the double colon.

Table 4-2 *Examples of Omitting a Single Contiguous String of All-0s Hextets**

Format	IPv6 Address
Preferred	**0000:0000:0000:0000:0000:0000:0000:0000**
(::) All-0s segments	**::**
Preferred	**0000:0000:0000:0000:0000:0000:0000:**0001
(::) All-0s segments	**::**0001
Preferred	ff02:**0000:0000:0000:0000:0000:0000:**0001
(::) All-0s segments	ff02**::**0001
Preferred	fe80:**0000:0000:0000**:a299:9bff:fe18:50d1
(::) All-0s segments	fe80**::**a299:9bff:fe18:50d1
Preferred	2001:0db8:1111:000a:00b0:**0000**:0200
(::) All-0s segments	2001:0db8:1111:000a:00b0**::**0200
Preferred	2001:0db8:**0000:0000**:abcd:0000:0000:1234
(::) All-0s segments	2001:0db8**::**abcd:0000:0000:1234
Preferred	2001:0db8:aaaa:0001:**0000:0000:0000**:0100
(::) All-0s segments	2001:0db8:aaaa:0001**::**0100
Preferred	2001:0db8:aaaa:0001:**0000:0000:0000**:0200
(::) All-0s segments	2001:0db8:aaaa:0001**::**0200

* In this table, the 0s in bold in the preferred address are replaced by the double colon.

Only a single contiguous string of all-0s segments can be represented with a double colon; otherwise, the address would be ambiguous, as shown in this example:

- Incorrect address using two double colons:

```
2001::abcd::1234
```

- Possible ambiguous choices:

```
2001:0000:0000:0000:0000:abcd:0000:1234
2001:0000:0000:0000:abcd:0000:0000:1234
2001:0000:0000:abcd:0000:0000:0000:1234
2001:0000:abcd:0000:0000:0000:0000:1234
```

As you can see, if two double colons are used, there are multiple possible interpretations, and you don't know which address is the correct one.

What happens if you have an address with more than one contiguous string of all-0s hextets—for example, 2001:0db8:0000:0000:abcd:0000:0000:1234? In that case, where should you use the single double colon (::)?

RFC 5952 states that the double colon should represent:

- The longest string of all-0s hextets.
- If the strings are of equal length, the first string should use the double colon (::) notation.

Therefore, 2001:0db8:0000:0000:abcd:0000:0000:1234 would be written 2001:0db8::abcd:0000:0000:1234. Applying both Rules 1 and 2, the address would be written 2001:db8::abcd:0:0:1234.

Note Most operating systems, including Cisco IOS and Microsoft Windows, accept the placement of a single double colon (::) in any valid location.

Combining Rule 1 and Rule 2

You can combine the two rules just discussed to reduce an address even further. Table 4-3 illustrates how this works, showing the preferred address, application of rule 1, and application of rule 2. Again, spaces are left so you can better visualize where the 0s have been removed.

Table 4-3 *Examples of Applying Both Rule 1 and Rule 2*

Format	IPv6 Address
Preferred	0000:0000:0000:0000:0000:0000:0000:0000
Leading 0s omitted	0: 0: 0: 0: 0: 0: 0: 0
(::) All-0s segments	::
Preferred	0000:0000:0000:0000:0000:0000:0000:0001
Leading 0s omitted	0: 0: 0: 0: 0: 0: 0: 1
(::) All-0s segments	::1

Format	IPv6 Address
Preferred	ff02:**000**0:**000**0:**000**0:**000**0:**000**0:**000**0:**000**1
Leading 0s omitted	ff02: 0: 0: 0: 0: 0: 0: 1
(::) All-0s segments	ff02::1
Preferred	fe80:**000**0:**000**0:**000**0:a299:9bff:fe18:50d1
Leading 0s omitted	fe80: 0: 0: 0:a299:9bff:fe18:50d1
(::) All-0s segments	fe80::a299:9bff:fe18:50d1
Preferred	2001:**0**db8:1111:**000**a:**00**b0:**000**0:9000:**0**200
Leading 0s omitted	2001: db8:1111: a: b0: 0:9000: 200
(::) All-0s segments	2001:db8:1111:a:b0::9000:200
Preferred	2001:**0**db8:**000**0:**000**0:abcd:**000**0:**000**0:1234
Leading 0s omitted	2001: db8: 0: 0:abcd: 0: 0:1234
(::) All-0s segments	2001:db8::abcd:0:0:1234
Preferred	2001:**0**db8:aaaa:**000**1:**000**0:**000**0:**000**0:**0**100
Leading 0s omitted	2001: db8:aaaa: 1: 0: 0: 0: 100
(::) All-0s segments	2001:db8:aaaa:1::100
Preferred	2001:**0**db8:aaaa:**000**1:**000**0:**000**0:**000**0:**0**200
Leading 0s omitted	2001: db8:aaaa: 1: 0: 0: 0: 200
(::) All-0s segments	2001:db8:aaaa:1::200

Table 4-4 shows the same examples as in Table 4-3, this time showing just the longest preferred form and the final compressed format after implementing both rules.

Table 4-4 *IPv6 Address Preferred and Compressed Formats*

Preferred Format	Compressed Format
0000:0000:0000:0000:0000:0000:0000:0000	::
0000:0000:0000:0000:0000:0000:0000:0001	::1
ff02:0000:0000:0000:0000:0000:0000:0001	ff02::1
fe80:0000:0000:0000:a299:9bff:fe18:50d1	fe80::a299:9bff:fe18:50d1
2001:0db8:1111:000a:00b0:0000:0000:0200	2001:db8:1111:a:b0::200
2001:0db8:0000:0000:abcd:0000:0000:1234	2001:db8::abcd:0:0:1234
2001:0db8:aaaa:0001:0000:0000:0000:0100	2001:db8:aaaa:1::100
2001:0db8:aaaa:0001:0000:0000:0000:0200	2001:db8:aaaa:1::200

Even after applying the two rules to compress the format, an IPv6 address can still look unwieldy. Don't worry! Chapter 5, "Global Unicast Address," shows a technique that I call the 3–1–4 rule. Using that rule makes IPv6 global unicast addresses (GUAs) easier to read than an IPv4 address and helps you recognize the parts of a GUA address.

Prefix Length Notation

In IPv4, the prefix (or network portion) of the address can be identified by a dotted-decimal netmask, commonly referred to as a *subnet mask*. For example, 255.255.255.0 indicates that the network portion, or prefix length, of the IPv4 address is the leftmost 24 bits. The 255.255.255.0 dotted-decimal netmask can also be written in CIDR notation as /24, indicating the 24 bits in the prefix.

IPv6 address prefixes can be represented much the same way that IPv4 address prefixes are written in CIDR notation. An IPv6 address prefix (the network portion of the address) is represented using the following format:

ipv6-address/prefix-length

The *prefix-length* is a decimal value indicating the number of leftmost contiguous bits of the address. It identifies the prefix (that is, the network portion) of the address. It is also used with unicast addresses to separate the prefix portion of the address from the Interface ID. Remember from Chapter 2 that the Interface ID is the equivalent to the host portion of an IPv4 address.

Let's look at an example using the address 2001:db8:aaaa:1111::100/64. The longest preferred form in Figure 4-2 illustrates how the /64 prefix length identifies the prefix, or network portion, of the address. The /64 prefix length leaves another 64 bits, which is the Interface ID portion of the address.

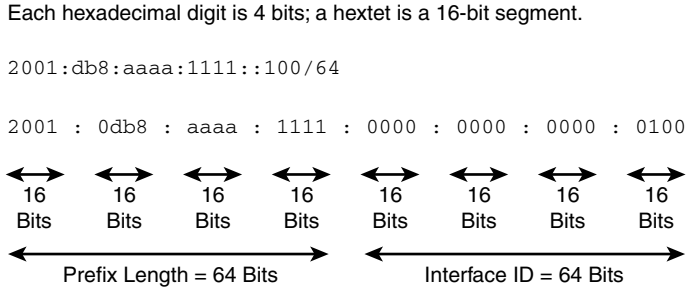


Figure 4-2 *IPv6 Prefix and Prefix Length*

In IPv6, just as in IPv4, the number of devices you can have on a network depends on the prefix length. However, due to the 128-bit length of an IPv6 address, there is no need to conserve address space as is needed with IPv4 public addresses.

In Figure 4-2, notice that the /64 prefix length results in an Interface ID of 64 bits. As we will discuss further in Chapter 5, this is a common prefix length for most end-user networks. A /64 prefix length gives us 18 quintillion devices on a single network (or subnet, if you prefer)!

Figure 4-3 shows several prefix length examples: /32, /48, /52, /56, /60, and /64. Notice that all of these examples fall on a *nibble boundary*, a multiple of 4 bits. Prefix lengths do not necessarily have to fall on a nibble boundary, although in most cases they do. Prefix lengths can also fall *within a nibble*—for example, /61, /62, or /63. We will discuss the prefix lengths, including within the nibble, more in Chapter 5 when we discuss the global unicast address, prefix allocation, and subnetting.

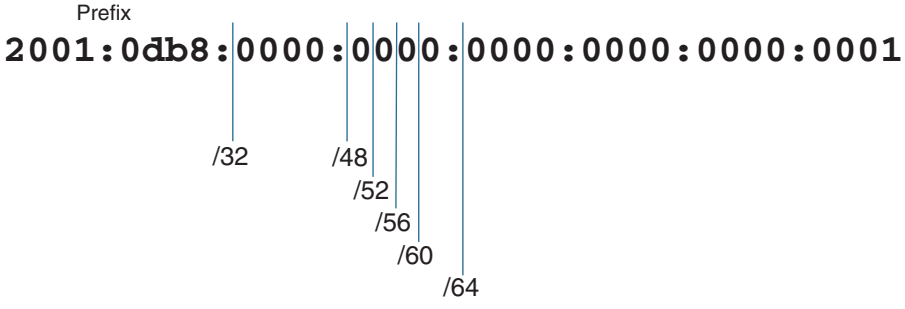


Figure 4-3 *IPv6 Prefix Length Examples*

IPv6 Address Types

We begin this section with a brief look at the IPv6 address space and how the different types of addresses are allotted within this space. Next, we examine the various addresses within three IPv6 address types: unicast, multicast, and anycast.

IPv6 address types are defined in RFC 4291, *IP Version 6 Addressing Architecture*. In this section, we examine the several types of unicast addresses, three types of multicast addresses, and the anycast address. We discuss some of these addresses in more detail than others. Global unicast addresses, link-local addresses, and multicast addresses are examined more closely in Chapters 5, 6, and 7.

Note IPv6 does not have a broadcast address. Other options exist in IPv6, such as a solicited-node multicast address and an all-IPv6 devices multicast address. Chapter 7 provides details on these types of addresses.

IPv6 Address Space

IPv4, with its 32-bit address space, provides for 4.29 billion (4,294,967,296) addresses. IPv6, with its 128-bit address space, provides for 340 undecillion addresses, or 340 trillion trillion trillion addresses. That's 340,282,366,920,938,463,463,374,607,431,768,211,456 addresses—a *lot* of addresses!

Many analogies have been made to help comprehend 340 undecillion (not all of which are completely accurate):

- "3,911,873,538,269,506,102 addresses per square meter of the surface of the planet Earth"[1]
- The number of grains of sand on Earth
- 10 nonillion addresses assigned to every person on Earth

As a disclaimer, I didn't do the math to calculate the number of square meters on the surface of Earth, and I haven't had a chance to count all the grains of sand on Earth either. And an argument can be made that this would be purely theoretical because of how addresses are allocated. Regardless, I think we can all agree that IPv6 provides an extremely large address space.

Figure 4-4 shows a chart of the powers of 10 to give a better idea of the tremendous increase in the IPv6 address space.

IPv4
4.29 Billion

IPv6
340 Undecillion

Number Name	Scientific Notation	Number of Zeros
1 Thousand	10^3	1,000
1 Million	10^6	1,000,000
1 Billion	10^9	1,000,000,000
1 Trillion	10^{12}	1,000,000,000,000
1 Quadrillion	10^{15}	1,000,000,000,000,000
1 Quintillion	10^{18}	1,000,000,000,000,000,000
1 Sextillion	10^{21}	1,000,000,000,000,000,000,000
1 Septillion	10^{24}	1,000,000,000,000,000,000,000,000
1 Octillion	10^{27}	1,000,000,000,000,000,000,000,000,000
1 Nonillion	10^{30}	1,000,000,000,000,000,000,000,000,000,000
1 Decillion	10^{33}	1,000,000,000,000,000,000,000,000,000,000,000
1 Undecillion	10^{36}	1,000,000,000,000,000,000,000,000,000,000,000,000
340,282,366,920,938,463,463,374,607,431,768,211,456		

Figure 4-4 *Powers of 10: Comparing IPv4 and IPv6 Address Space*

As mentioned in Chapter 1, "Introduction to IPv6," this means that we can now design IPv6 addressing schemas based on management and security plans, without the concern for public address depletion that we face with IPv4. (This will become even more evident in Chapter 5, when we discuss the global unicast address and subnetting.)

Table 4-5 shows the Internet Assigned Numbers Authority's (IANA's) allocation of the 128-bit IPv6 address space. Notice the allocations for global unicast, unique local unicast, link-local unicast, and multicast addresses. It may be a little difficult to visualize this using the table, so Figure 4-5 shows this same allocation in a pie chart to make it a little easier. Using the first 3 bits, the chart divides the IPv6 pie into eight slices (that is, 3 bits gives us eight possibilities). There are portions within the 000 and 111 slices used to indicate very small allocations (the chart shows them larger than the actual allocations) from this part of the address space.

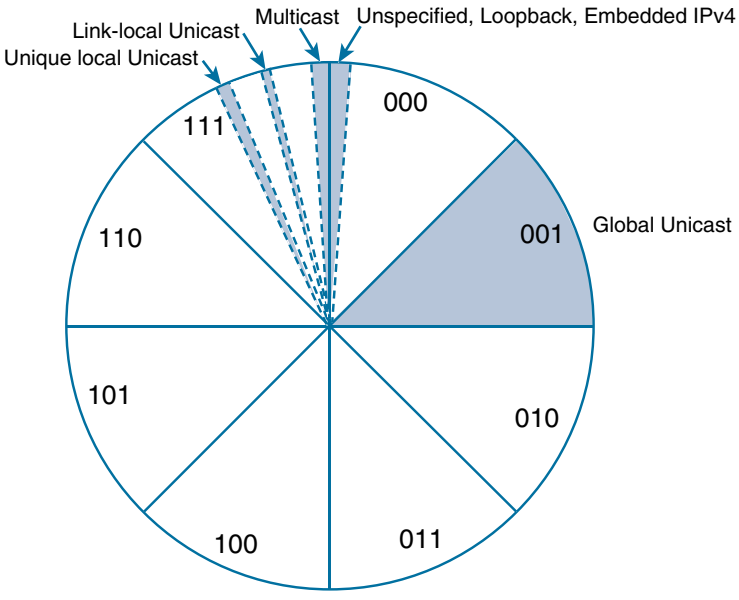
Table 4-5 *IANA's Allocation of IPv6 Address Space**

Leading Bits	Address	Range of First Hextet	Allocation	Fraction of Space
000x		**0000** **1fff**		1/8
0000 0000	0000::/8	0000 00ff	Unspecified, loopback, embedded	1/256
0000 0001 through 0001 xxxx	0000::/3	0100 1fff	Reserved by IETF	Remaining 1/8
001x	2000::/3	2000 3fff	Global unicast	1/8
010x	4000::/3	4000 5fff	Reserved by IETF	1/8
011x	6000::/3	6000 7fff	Reserved by IETF	1/8
100x	8000::/3	8000 9fff	Reserved by IETF	1/8
101x	a000::/3	a000 bfff	Reserved by IETF	1/8

Leading Bits	Address	Range of First Hextet	Allocation	Fraction of Space
110x	c000::/3	c000 dfff	Reserved by IETF	1/8
111x				1/8
1110 xxxx	e000::/4	e000 efff	Reserved by IETF	1/16
1111 0xxx	f000::/5	f000 f7ff	Reserved by IETF	1/32
1111 10xx	f800::/6	f800 fbff	Reserved by IETF	1/64
1111 110x	fc00::/7	fc00 fdff	Unique local unicast	1/128
1111 1110 0	fe00::/9	fe00 fe74	Reserved by IETF	1/512
1111 1110 10	fe80::/10	fe80 febf	Link-local unicast	1/1024
1111 1110 11	fec0::/10	fec0 feff	Reserved by IETF; previously site-local (deprecated)	1/1024
1111 1111	ff00::/8	ff00 ffff	Multicast	1/256

* In this table, the "Range of First Hextet" column does not show the complete range of the address space. For example, the actual range of the global unicast address space would be 2000:: through 3fff:ffff:ffff:ffff:ffff:ffff:ffff:ffff.

In both Table 4-5 and Figure 4-5, the IPv6 address space is divided into eighths, using the leading 3 bits (000, 001, 010, 011, 100, 101, 110, and 111). This information might be a little confusing right now, but it will become more obvious as you examine each of the IPv6 address types.



The remaining portions of IPv6 address space are reserved by IETF for future use.

Figure 4-5 *IANA's Allocation of IPv6 Address Space in 1/8 Sections*

Unicast Addresses

Figure 4-6 diagrams the three types of addresses: unicast, multicast, and anycast. We begin by looking at unicast addresses. Don't be intimidated by all the different types of unicast addresses. The most significant types are global unicast addresses, which are equivalent to IPv4 public addresses, and link-local addresses. These address types are discussed in detail in Chapters 5 and 6.

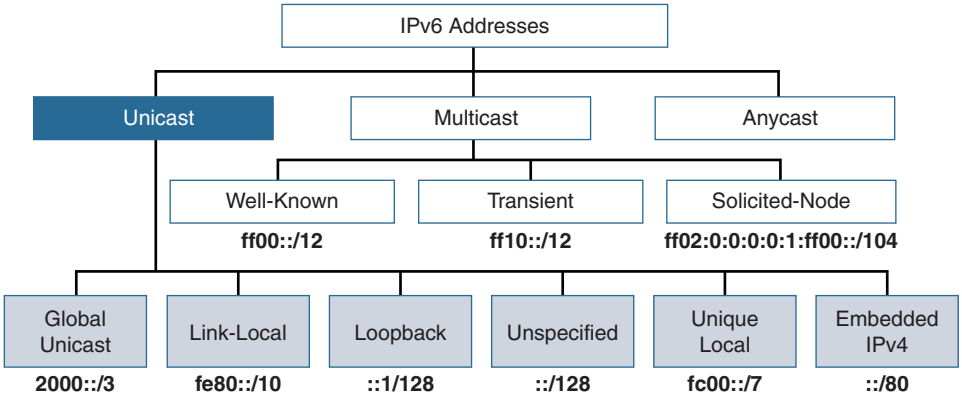


Figure 4-6 *IPv6 Address Types: Unicast Addresses*

A unicast address uniquely identifies an interface on an IPv6 device. A packet sent to a unicast address is received by the interface that is assigned to that address. Similar to IPv4, a source IPv6 addresses must be a unicast address.

Note Notice that there is no broadcast address shown in Figure 4-6. Remember that IPv6 does not include a broadcast address.

This section covers the different types of unicast addresses, as illustrated in Figure 4-6. The following is a quick preview of each type of unicast address discussed in this section:

- **Global unicast:** A routable address in the IPv6 Internet, similar to a public IPv4 address (covered in more detail in Chapter 5).
- **Link-local:** Used only to communicate with devices on the same local link (covered in more detail in Chapter 6).
- **Loopback:** An address not assigned to any physical interface that can be used for a host to send an IPv6 packet to itself.
- **Unspecified address:** Used only as a source address and indicates the absence of an IPv6 address.
- **Unique local:** Similar to a private address in IPv4 (RFC 1918) and not intended to be routable in the IPv6 Internet. However, unlike RFC 1918 addresses, these addresses are not intended to be statefully translated to a global unicast address.
- **IPv4 embedded:** An IPv6 address that carries an IPv4 address in the low-order 32 bits of the address.

Global Unicast Address

Global unicast addresses (GUAs), also known as *aggregatable global unicast addresses*, are globally routable and reachable in the IPv6 Internet. They are equivalent to public IPv4 addresses. They play a significant role in the IPv6 addressing architecture. One of the main motivations for transitioning to IPv6 is the exhaustion of its IPv4 counterpart. As you can see in Figure 4-6, a GUA address is only one of several types of IPv6 unicast addresses.

Figure 4-7 shows the generic structure of a GUA, which has three fields:

- **Global Routing Prefix:** The Global Routing Prefix is the prefix or network portion of the address assigned by the provider, such as an ISP, to the customer site.
- **Subnet ID:** The Subnet ID is a separate field for allocating subnets within the customer site. Unlike with IPv4, it is not necessary to borrow bits from the Interface ID (host portion) to create subnets. The number of bits in the Subnet ID falls between where the Global Routing Prefix ends and where the Interface ID begins. This makes subnetting simple and manageable.
- **Interface ID:** The Interface ID identifies the interface on the subnet, equivalent to the host portion of an IPv4 address. The Interface ID in most cases is 64 bits.

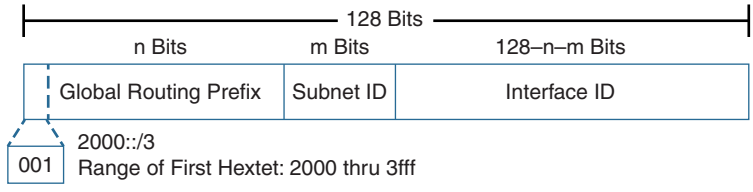


Figure 4-7 *Structure of a GUA Address*

Figure 4-7 illustrates the more general structure, without the specific sizes for any of the three parts. The first 3 bits of a GUA address begin with the binary value 001, which results in the first hexadecimal digit becoming a 2 or a 3. (We look at the structure of the GUA address more closely in Chapter 5.)

There are several ways a device can be configured with a global unicast address:

- Manually configured.
- Stateless Address Autoconfiguration.
- Stateful DHCPv6.

Example 4-1 demonstrates how to view the global unicast address on Windows and Mac OS operating systems, using the **ipconfig** and **ifconfig** commands, respectively. The **ifconfig** command is also used with the Linux operating system and provides similar output.

Note You may see multiple IPv6 global unicast addresses including one or more temporary addresses. You'll learn more about this in Chapter 9.

Example 4-1 *Viewing IPv6 Addresses on Windows and Mac OS*

```
Windows-OS> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  .  :
   ! IPv6 GUA
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086
   ! IPv6 Link-Local
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
   IPv4 Address. . . . . . . . . . . : 192.168.1.100
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   ! IPv6 Default Gateway
   Default Gateway . . . . . . . . . : fe80::1%11
                                       192.168.1.1
--------------------------------------------------------------------------------
Mac-OS$ ifconfig
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
   ether 60:33:4b:15:24:6f
   ! IPv6 Link-Local
   inet6 fe80::6233:4bff:fe15:246f%en1 prefixlen 64 scopeid 0x5
   inet 192.168.1.111 netmask 0xffffff00 broadcast 192.168.1.255
   ! IPv6 GUA
   inet6 2001:db8:cafe:1:4bff:fe15:246f prefixlen 64 autoconf
   media: autoselect
   status: active
```

This section has provided just a brief introduction to global unicast addresses. Remember that IPv6 introduced a lot of changes to IP. Devices may obtain more than one GUA address for reasons such as privacy. For a network administrator needing to manage and control access within a network, having these additional addresses that are not administered through stateful DHCPv6 may be undesirable. Chapter 11 discusses devices obtaining or creating multiple global unicast addresses and various options to ensure that devices only obtain a GUA address from a stateful DHCPv6 server.

Link-Local Unicast Address

Link-local addresses are another type of unicast address as shown in Figure 4-6. A link-local address is a unicast address that is confined to a single link, a single subnet. Link-local addresses only need to be unique on the link (subnet) and do not need to be unique beyond the link. Therefore, routers do not forward packets with a link-local address. Devices can use Duplicate Address Detection (DAD) to determine whether or not the link-local address is unique.

Note Link-local unicast addresses are discussed in detail in Chapter 6. ICMPv6 DAD is examined in Chapter 13, "ICMPv6 Neighbor Discovery."

Figure 4-8 shows the format of a link-local unicast address, which is in the range fe80::/10. Using this prefix and prefix length results in the range of the first hextet being from fe80 to febf.

Note Using a prefix other than fe80 for a link-local address can result in unexpected behaviors. Although permitted by the RFC 4291, using a prefix other than fe80 should be tested prior to usage.

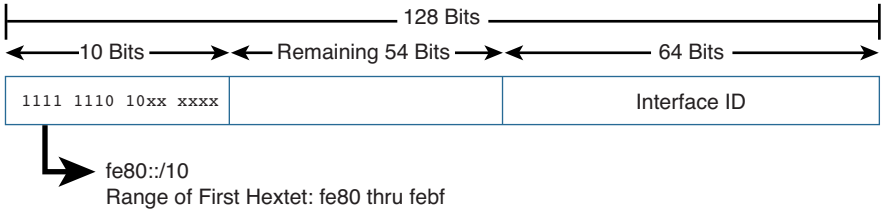


Figure 4-8 *Structure of a Link-Local Unicast Address*

In Chapter 6 we will examine the structure, uses, and configuration options for link-local addresses in much more detail. For now, here is a summary of some of the key points:

- To be an IPv6-enabled device, a device must have an IPv6 link-local address. The device doesn't have to have an IPv6 global unicast address, but it must have a link-local address.
- Link-local addresses are not routable off the link (IPv6 subnet). Routers do not forward packets with a link-local address.
- Link-local addresses only have to be unique on the link. It is very likely and sometimes even desirable for a device to use the same link-local address on different interfaces that are on different links.
- There can be only one link-local address per interface.

Configuration options for link-local addresses are (see Chapters 6 and 9 for more details):

- Devices dynamically (automatically) create their own link-local IPv6 address upon startup. This is the default on most operating systems, including Cisco IOS, Windows, Mac OS, and Linux.
- Link-local addresses can be manually configured.

The idea of a device creating its own IP address upon startup is really an amazing benefit in IPv6! Think of it. A device can create its own IPv6 link-local address completely on its own, without any kind of manual configuration or the services of a DHCP server. This means that the device can immediately communicate with any other device on its link (IPv6 subnet). A device may only need a link-local address because it only needs to communicate with other devices on its same network. Or it can use its link-local address to communicate with a device where it can obtain information for getting or creating a global unicast address, such as an IPv6 router or a DHCPv6 server. The device can then use this information to communicate with devices on other networks.

This solves the "Which came first, the chicken or the egg?" problem with IPv4. That is, "How do I ask a DHCP server for an IP address when I first need to have an IP address before I can communicate with the server to ask for one?" (DHCP for IPv4 uses a Discover message with an IPv4 source address of 0.0.0.0.) With IPv6, during startup the device automatically gives itself a link-local address that is unique on that subnet. It can then use this address to communicate with any device on the network, including an IPv6 router and, if necessary, a DHCPv6 server. Remember from Chapter 2 that an IPv6 router sends ICMPv6 Router Advertisement messages that allow the device to obtain a global unicast address, with or without the services of DHCPv6.

Example 4-1 demonstrates how to view the link-local address on Windows and Mac OS operating systems by using the **ipconfig** and **ifconfig** commands. These operating systems, as well as Linux, are enabled for IPv6 by default. So, even if the devices did not have a global unicast address, as shown in Example 4-1, you would still see the IPv6 link-local address. And as discussed in Chapter 2, this means client hosts are running IPv6 and, at a minimum, the network should be secured to prevent IPv6 attacks.

Note Notice the %11 and %en1 following the IPv6 link-local addresses in Example 4-1. These are known as *zone identifiers*, and they are used to identify the interface on the device. These are usually of little importance when referring to a link-local address, but they are highly significant for tying the address to the interface. Zone identifiers are discussed in Chapter 6.

The following are some of the ways IPv6 devices use a link-local address:

- When a device starts up, before it obtains a GUA address, the device uses its IPv6 link-local address as its source address to communicate with other devices on the network, including the local router.
- Devices use the router's link-local address as their default gateway address.
- Routers exchange IPv6 dynamic routing protocol (OSPFv3, EIGRP for IPv6, RIPng) messages from their IPv6 link-local address.
- IPv6 routing table entries populated from dynamic routing protocols use the IPv6 link-local address as the next-hop address.

This section has provided just an introduction to the link-local address. We will explore all these topics in more detail in Chapter 6.

Loopback Addresses

A loopback address is another type of unicast address (refer to Figure 4-6). An IPv6 loopback address is ::1, an all-0s address except for the last bit, which is set to 1. It is equivalent to the IPv4 address block 127.0.0.0/8, most commonly the 127.0.0.1 loopback address.

Table 4-6 shows the different formats for representing an IPv6 loopback address.

Table 4-6 *IPv6 Loopback Address Representations*

Representation	IPv6 Loopback Address
Preferred	0000:0000:0000:0000:0000:0000:0000:0001
Leading 0s omitted	0:0:0:0:0:0:0:1
Compressed	::1

The loopback address can be used by a node to send an IPv6 packet to itself, typically when testing the TCP/IP stack. Loopback addresses have the following characteristics:

- A loopback address cannot be assigned to a physical interface.
- A packet with a loopback address, source address, or destination address should never be sent beyond the device.
- A router can never forward a packet with a destination address that is a loopback address.
- The device must drop a packet received on an interface if the destination address is a loopback address.

Unspecified Addresses

An unspecified unicast address is an all-0s address (refer to Figure 4-6). An unspecified unicast address is used as a source address to indicate the absence of an address. It cannot be assigned to an interface.

One example where an unspecified address can be used is as a source address in ICMPv6 Duplicate Address Detection (DAD). DAD is a process that a device uses to ensure that its unicast address is unique on the local link (network). DAD is discussed in Chapter 14.

Table 4-7 shows the different formats for representing an IPv6 unspecified address.

Table 4-7 *IPv6 Unspecified Address Representations*

Representation	IPv6 Unspecified Address
Preferred	0000:0000:0000:0000:0000:0000:0000:0000
Leading 0s omitted	0:0:0:0:0:0:0:0
Compressed	::

Unspecified addresses have the following characteristics:

- An unspecified source address indicates the absence of an address.
- An unspecified address cannot be assigned to a physical interface.
- An unspecified address cannot be used as a destination address.
- A router will never forward a packet that has an unspecified source address.

Unique Local Addresses

Figure 4-6 shows another type of IPv6 unicast address, the unique local address (ULA), which is the counterpart of IPv4 private addresses. Unique local addresses are also known as *private IPv6 addresses* or *local IPv6 addresses* (not to be confused with link-local addresses).

ULA addresses can be used similarly to global unicast addresses but are for private use and should not be routed in the global Internet. ULA addresses are only to be used in a more limited area, such as within a site or routed between a limited number of administrative domains. ULA addresses are for devices that never need access to the Internet and never need to be accessible from the Internet.

ULA addresses are defined in RFC 4193, *Unique Local IPv6 Unicast Addresses*. Figure 4-9 illustrates the format of a unique local unicast address.

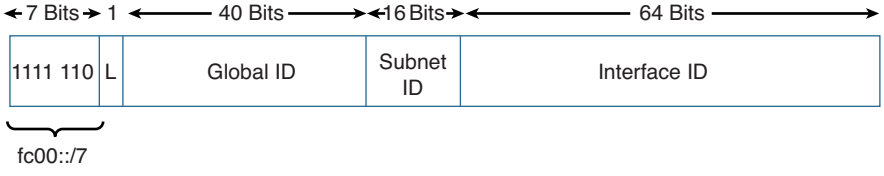


Figure 4-9 *Structure of a Unique Local Unicast Address*

Unique local addresses have the prefix fc00::/7, which results in the range of addresses from fc00::/7 to fdff::/7, as shown in Table 4-8.

Table 4-8 *Range of Unique Local Unicast Addresses*

Unique Local Unicast Address (Hexadecimal)	Range of First Hextet	Range of First Hextet in Binary
fc00::/7	fc00 to	**1111 110**0 0000 0000 to
	fdff	**1111 110**1 1111 1111

Unique local addresses have the following characteristics:

- They can be used just like global unicast addresses.
- They can be used for devices that never need access to or from the global Internet.
- They allow sites to be combined or privately interconnected without address conflicts and without requiring addressing renumbering. (Address conflicts are highly unlikely due to the large address space.)
- They are independent of any ISP and can be used within a site even without having Internet connectivity.

ULA and NAT

ULA and NAT is a bit of a tricky topic. The concept of translating a unique local address to a global unicast address is the subject of ongoing debate within the IPv6 community, and it fosters emotional opinions on both sides of the argument. The IAB published an informational RFC highlighting its thoughts on NAT and IPv6 in RFC 5902, *IAB Thoughts on IPv6 Network Address Translation*. In this RFC, the IAB summarizes the use of NAT as follows:

> Network address translation is viewed as a solution to achieve a number of desired properties for individual networks: avoiding renumbering, facilitating multihoming, making configurations homogenous, hiding internal network details, and providing simple security.

So, does this means NAT provides security, and ULA addresses can be translated to GUA addresses for this purpose? The simple answer is no. RFC 5902 goes on to state, "However, one should not confuse NAT boxes with firewalls. As discussed in [RFC 4864] Section 2.2, the act of translation does not provide security in itself."

Remember that the driving force for using NAT with IPv4 is not security but IPv4 address depletion. Although the IAB and the IETF did not intend for NAT to be used with IPv6 as it is with IPv4, NAT does provide mechanisms for translation where translation is necessary. These translation techniques include Network Prefix Translation version 6 (NPTv6), described in RFC 6296, *IPv6-to-IPv6 Network Prefix Translation*, and NAT66, described in an Internet draft RFC, *IPv6-to-IPv6 Network Address Translation* (long expired). Both of these RFCs focus on translation for address independence—and only where necessary. In RFC 6296, the IETF goes as far as stating, "For reasons discussed in [RFC 2993] and Section 5, the IETF does not recommend the use of Network Address Translation technology for IPv6."

Both NPTv6 and NAT66 are designed for address independence and not security. *Address independence* means that a site does not have to renumber its internal addresses if the ISP changes the site's external prefix or if the site changes ISPs and receives a different prefix.

NPTv6 and NAT66 are both stateless technologies, whereas NAT for IPv4 is stateful. It is the statefulness, not NAT itself, that provides the security. This means that internal devices are open to certain types of attacks that would not be possible in a NAT for IPv4 network. Without getting into the NAT-versus-security debate covered in Chapter 1, NAT for IPv4 is not security and introduces many problems and challenges.

If all this seems vague, complicated, and perhaps even contradictory, welcome to the discussion on NAT and IPv6.

Note For more information on ULA addresses with NAT66 or NPTv6, see Ed Horley's excellent articles on the topic, at www.howfunky.com. Horley has also written an excellent book, *Practical IPv6 for Windows Administrators*.

L Flag and Global ID

ULA addresses have the prefix fc00::/7, or the first 7 bits as 1111 110x. As shown in Figure 4-10, the eighth bit (x) is known as the L flag, or the local flag, and it can be either 0 or 1. This means that the ULA address range is divided into two parts:

- **fc00::/8** (1111 1100): When the L flag is set to 0, may be defined in the future.
- **fd00::/8** (1111 1101): When the L flag is set to 1, the address is locally assigned.

Because the only legitimate value for the L flag is 1, the only valid ULA addresses today are in the fd00::/8 prefix.

Another difference between ULA addresses and private IPv4 addresses is that ULA addresses can also be globally unique. This is helpful for ensuring that there won't be any conflicts when combining two sites using ULA addresses or just in case they get leaked out into the Internet.

The trick is that the global IDs must somehow be unique without being administered by a central authority. RFC 4193, *Sample Code for Pseudo-Random Global ID Algorithm*, defines a process whereby locally assigned Global IDs can be generated using a pseudorandom algorithm that gives them a very high probability of being unique. It is important that all sites generating Global IDs use the same algorithm to ensure that there is this high probability of uniqueness.

Note This section includes some information on the random Global ID algorithm for your reference. This information is not critical to your fundamental understanding of IPv6, and you can skip it if you prefer.

The algorithm defined in RFC 4193 is beyond the scope of this book, but these are the six steps from Section 3.2.2 of RFC 4193:

3.2.2. Sample Code for Pseudo-Random Global ID Algorithm

The algorithm described below is intended to be used for locally assigned Global IDs. In each case the resulting global ID will be used in the appropriate prefix as defined in Section 3.2.

1. Obtain the current time of day in 64-bit NTP format [NTP].
2. Obtain an EUI-64 identifier from the system running this algorithm. If an EUI-64 does not exist, one can be created from a 48-bit MAC address as specified in [ADDARCH]. If an EUI-64 cannot be obtained or created, a suitably unique identifier, local to the node, should be used (e.g., system serial number).
3. Concatenate the time of day with the system-specific identifier in order to create a key.
4. Compute an SHA-1 digest on the key as specified in [FIPS, SHA1]; the resulting value is 160 bits.
5. Use the least significant 40 bits as the Global ID.
6. Concatenate fc00::/7, the L bit set to 1, and the 40-bit Global ID to create a Local IPv6 address prefix.

Note The algorithm in RFC 4193 requires a /48 prefix. It does not work well if a larger prefix or contiguous prefixes are needed.

This algorithm will result in a Global ID that is reasonably unique and can be used to create a locally assigned local IPv6 address prefix. You can use the following website to generate and register your ULA address space: www.sixxs.net/tools/grh/ula.

Site-Local Addresses (Deprecated)

The original IPv6 specification allocated address space, similar to RFC 1918, *Private Address Space in IPv4*, for site-local addresses. Site-local addresses have since been deprecated (that is, made obsolete).

Site-local addresses, defined in RFC 3513, were given the prefix range fec0::/10. (You will most likely come across this prefix in older documentation.) The problem was that the term *site* was ambiguous. No one could really agree on what a site really meant. The other issue was that there was no guarantee that two sites within the same organization wouldn't end up using the same or overlapping site-local addresses, which kind of defeats the purpose of IPv6 and all this extra address space. Therefore, site-local addresses have been deprecated and replaced with unique local addresses.

IPv4 Embedded Address

The final unicast address types are IPv4 embedded addresses, as shown in Figure 4-6. IPv4 embedded addresses are IPv6 addresses used to aid the transition from IPv4 to IPv6. IPv4 embedded addresses carry an IPv4 address in the low-order 32 bits. These addresses are used to represent an IPv4 address inside an IPv6 address. RFC 4291 defines two types of IPv4 embedded addresses:

- IPv4-mapped IPv6 addresses
- IPv4-compatible IPv6 addresses (deprecated)

Special techniques such as tunnels are used to provide communications between islands of IPv6 devices over an IPv4-only network. To support this compatibility, IPv4 addresses can be embedded within an IPv6 address. This is easy to do because a 128-bit IPv6 address has plenty of room for the 32-bit IPv4 address. Basically, IPv6 just puts it at the end of the address and pads the front end. IPv4 and IPv6 packets are not compatible. Features such as NAT64 are required to translate between the two address families. See Chapter 17, "Deploying IPv6 in the Network," for more information.

IPv4-Mapped IPv6 Addresses

IPv4-mapped IPv6 addresses can be used by a dual-stack device that needs to send an IPv6 packet to an IPv4-only device. As shown in Figure 4-10, the first 80 bits are set to all 0s, and the 16-bit segment preceding the 32-bit IPv4 address is all 1s. The last 32 bits in the IPv4 address are represented in dotted-decimal notation. So, the first 96 bits are represented in hexadecimal, and the last 32 bits contain the IPv4 address in dotted-decimal notation.

With an IPv4-mapped IPv6 address, the IPv4 address does not have to be globally unique.

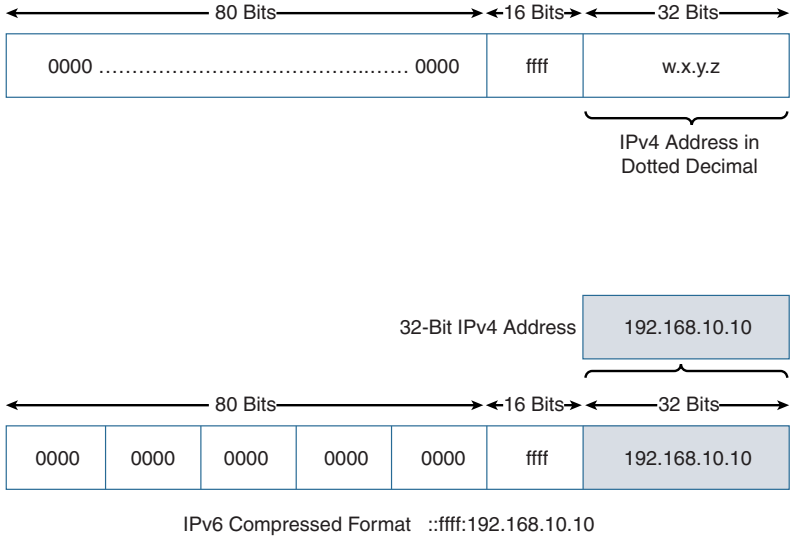


Figure 4-10 *IPv4-Mapped IPv6 Address*

Table 4-9 shows the various formats for representing an IPv4-mapped IPv6 address using the IPv4 address 192.168.10.10.

Table 4-9 *IPv4-Mapped IPv6 Address Representations*

Representation	IPv4-Mapped IPv6 Address
Preferred	0000:0000:0000:0000:0000:0000:ffff:192.168.10.10
Leading 0s omitted	0:0:0:0:0:0:ffff:192.168.10.10
Compressed	::ffff:192.168.10.10

Although there are many transition techniques available, the goal should always be native end-to-end IPv6 connectivity.

IPv4-Compatible IPv6 Addresses (Deprecated)

The deprecated IPv4-compatible IPv6 address is almost identical to an IPv4-mapped IPv6 address, except all 96 bits—including the 16-bit segment preceding the 32-bit IPv4 address—are all 0s. Another difference is that the IPv4 address used in the IPv4-compatible IPv6 address must be a globally unique IPv4 unicast address. The IPv4-compatible IPv6 address was rarely used and is now deprecated. Current IPv6 transition mechanisms no longer use this address type.

Note Chapter 18 discusses transition and coexistence strategies.

Multicast Addresses

Figure 4-11 shows the types of multicast addresses. Multicast is a technique in which a device sends a single packet to multiple destinations simultaneously (one-to-many). (Remember that a unicast address sends a single packet to a single destination [one-to-one].) Multiple destinations can actually be multiple interfaces on the same device, but they are typically different devices.

Note Figure 4-11 does not show all types of multicast addresses but is used to indicate the three multicast addresses this book focuses on.

An IPv6 multicast address defines a group of devices known as a *multicast group*. IPv6 multicast addresses use the prefix ff00::/8, shown in Table 4-10, which is equivalent to the IPv4 multicast address 224.0.0.0/4. A packet sent to a multicast group always has a unicast source address. A multicast address can never be the source address. Unlike IPv4, there is no broadcast address in IPv6. Instead, IPv6 uses multicast, including an all-IPv6 devices well-known multicast address and a solicited-node multicast address.

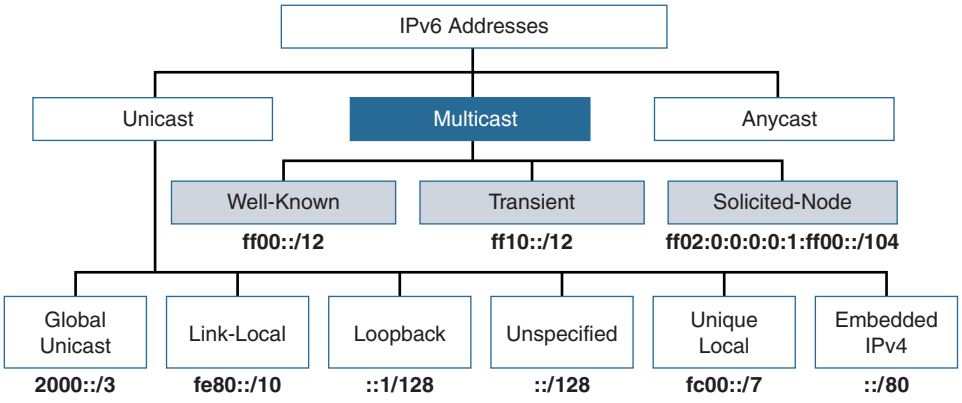


Figure 4-11 *Multicast Addresses*

Table 4-10 *IPv6 Multicast Address Representations*

Representation	IPv6 Multicast Address
Preferred	ff00:0000:0000:0000:0000:0000:0000:0000/8
Leading 0s omitted	ff00:0:0:0:0:0:0:0/8
Compressed	ff00::/8

Figure 4-12 shows the structure of an IPv6 multicast address. The first 8 bits are 1-bits (ff), followed by 4 bits allocated for flags and a 4-bit Scope field. The Scope field defines the range to which routers can forward the multicast packet. The next 112 bits represent the Group ID.

The 4 bits following 1111 1111 represent four different flags. The first three flags, 0 (reserved), R (rendezvous point), and P (network prefix), are beyond the scope of this book. The fourth flag, the least significant bit (LSB), or rightmost bit, is the transient flag (T flag). The T flag denotes the two types of multicast addresses:

- **Permanent (0):** These addresses, known as *predefined multicast addresses*, are assigned by IANA and include both well-known and solicited multicast.
- **Nonpermanent (1):** These are "transient" or "dynamically" assigned multicast addresses. They are assigned by multicast applications.

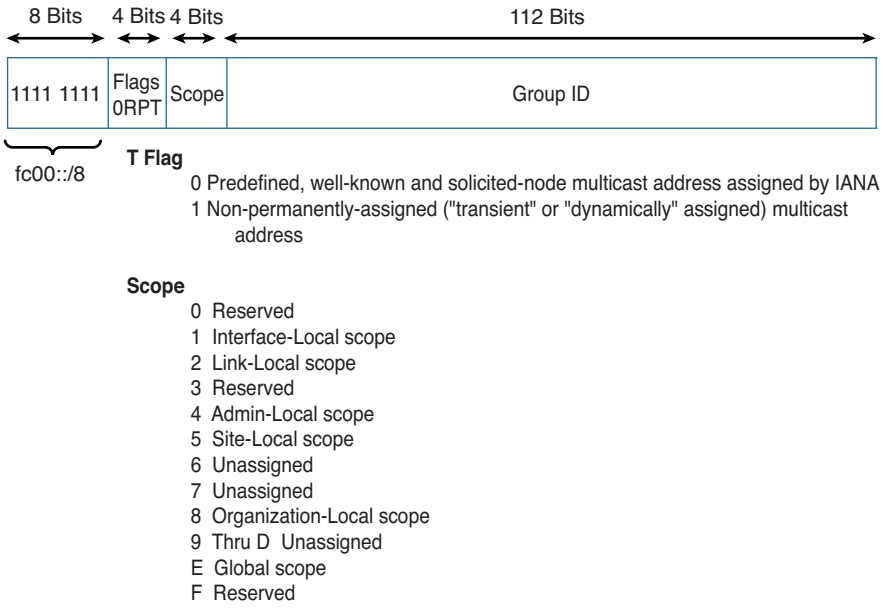


Figure 4-12 *IPv6 Multicast Address*

As shown in Figure 4-11, there are two types of predefined multicast addresses, both of which use the Flag field with a 0x0 value:

- Well-known multicast addresses
- Solicited-node multicast addresses

Note For additional information on IPv6 multicast and multicast routing, I highly suggest resources by Tim Martin, Cisco Systems, including the video *IPv6 Summit 2015: IPv6 Multicast Technologies*, at www.youtube.com/watch?v=H6bBiIPfYXM. Tim Martin also has an excellent Cisco Press LiveLessons video series, *IPv6 Design & Deployment LiveLessons* (see lesson 5).

Well-Known Multicast Addresses

Well-known multicast addresses have the prefix ff00::/12. As shown in Figure 4-12, this means that the third hexadecimal digit, the Flag field, is always set to 0. Well-known multicast addresses are predefined or reserved multicast addresses for assigned groups of devices. These addresses are equivalent to IPv4 well-known multicast addresses in the range 224.0.0.0 to 239.255.255.255. Some examples of IPv6 well-known multicast addresses include the following:

- **ff02::1:** All IPv6 devices
- **ff02::2:** All IPv6 routers
- **ff02::5:** All OSPFv3 routers
- **ff02::a:** All EIGRP (IPv6) routers

Solicited-Node Multicast Addresses

Solicited-node multicast addresses are used as a more efficient approach to IPv4's broadcast address. As discussed in Chapter 2, the solicited-node multicast is used in Layer 3-to-Layer 2 address resolution, similar to how Address Resolution Protocol (ARP) is used in IPv4. Solicited-node multicast addresses are automatically created using a special mapping of the device's unicast address with the solicited-node multicast prefix ff02:0:0:0:0:1:ff00::/104. Solicited-node multicast addresses are automatically created for every unicast address on a device.

Note Multicast addresses, the Scope field, assigned multicast, and solicited-node multicast are discussed in detail in Chapter 7.

Anycast Addresses

The last type of IPv6 address examined in this chapter is the anycast address (see Figure 4-13). An IPv6 anycast address is an address that can be assigned to more than one interface (typically different devices). In other words, multiple devices can have the same anycast address. A packet sent to an anycast address is routed to the "nearest" interface having that address, according to the router's routing table.

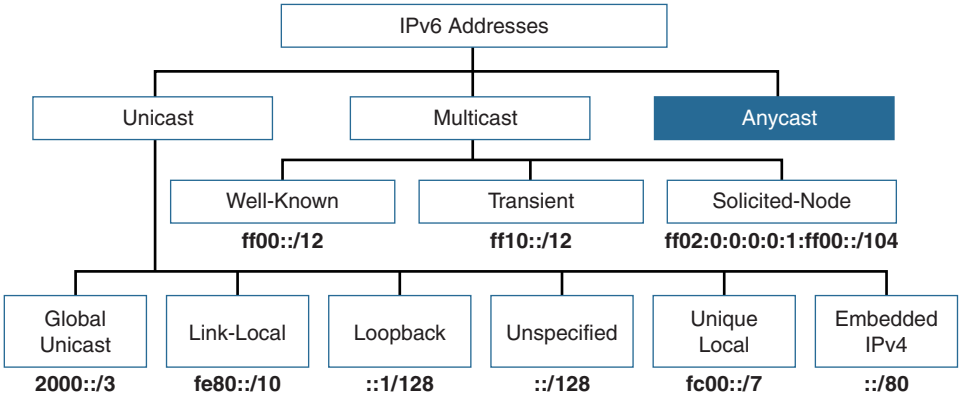


Figure 4-13 *Anycast Addresses*

Anycast addresses are available for both IPv4 and IPv6, initially defined in RFC 1546, *Host Anycasting Service*. Anycast was meant to be used for services such as DNS and HTTP but was never really implemented as designed.

There is no special prefix for an IPv6 anycast address. An IPv6 anycast address uses the same address range as global unicast addresses. Each participating device is configured to have the same anycast address. For example, servers A, B, and C in Figure 4-14 could be DHCPv6 servers with a direct Layer 3 connection into the network. These servers could advertise the same /128 address using OSPFv3. The router nearest the client request would then forward packets to the *nearest* server identified in the routing table.

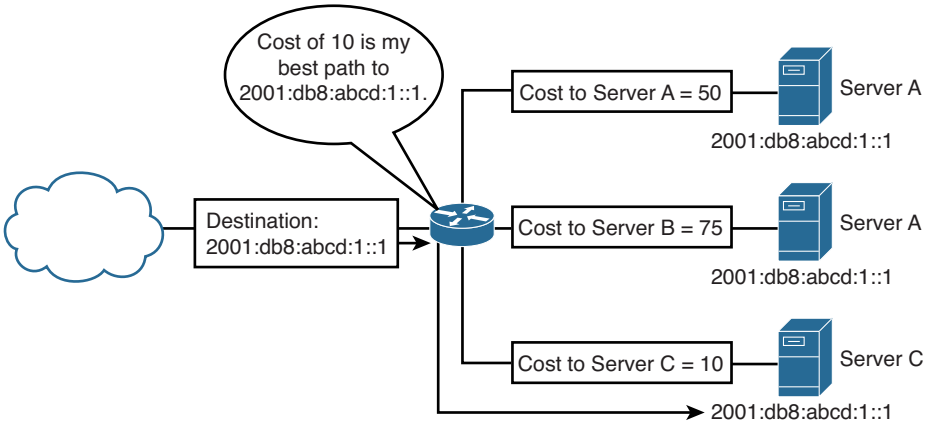


Figure 4-14 *Example of Anycast Addressing*

There are some reserved anycast address formats such as the subnet-router anycast address defined in RFC 4291 and RFC 2526. IPv6 anycast addressing is still somewhat in the experimental stages and beyond the scope of this book.

Summary

This chapter explains the basics of IPv6 addressing. The preferred format of an IPv6 128-bit address is written as eight 16-bit segments (hextets), separated by colons. The notation of the address can be reduced by omitting leading 0s and by using the double colon to replace contiguous hextets of 0s.

The IPv6 address space is extremely large. IPv6, with its 128-bit address space, provides for 340 undecillion addresses. Currently, only one-eighth of this space has been allocated for global unicast addresses, and a very small portion has been allocated for other unicast and multicast addresses.

This chapter introduces the three types of IPv6 addresses: unicast, multicast, and anycast. The following is a brief description of each of the addresses as discussed in this chapter:

- **Unicast addresses:** A unicast address uniquely identifies an interface on an IPv6 device. A source IPv6 address must be a unicast address. There are several types of unicast addresses:
  - **Global unicast addresses (GUAs):** Global unicast addresses are also known as an *aggregatable global unicast address*. These addresses are globally routable and reachable on the IPv6 Internet. They are equivalent to public IPv4 addresses. The current GUA address assignment from IANA begins with the binary value 001 or the prefix 2000::/3.
  - **Link-local addresses:** A link-local unicast address (fe80::/10) is a unicast address that is confined to a single link. The uniqueness of this address only has to be assured on that link because these packets are not routable off the link. An IPv6-enabled device must have a link-local address. Link-local unicast addresses are usually automatically created but can also be manually configured.
  - **Loopback addresses:** A loopback address is an all-0s address except for the last bit, which is set to 1. It is equivalent to the IPv4 loopback address, 127.0.0.1.
  - **Unspecified addresses:** An unspecified unicast address is an all-0s address. It cannot be assigned to an interface. An unspecified unicast address is used as a source address to indicate the absence of an address.
  - **Unique local addresses:** A unique local address (fc00::/7) is similar to the RFC 1918 private address space in IPv4. Unique local addresses should not be routable in the global Internet. They are to be used in more limited areas, such as within a site, or routed between a limited number of sites.
  - **IPv4 embedded addresses:** IPv6 addresses aid in the transition from IPv4 to IPv6. An IPv4 embedded address carries an IPv4 address in the low-order 32 bits. This type of address is used to represent an IPv4 address inside an IPv6 address. IPv4-mapped IPv6 addresses are the current type of IPv4 embedded addresses, with IPv4-compatible IPv6 addresses having been deprecated.
- **Multicast addresses:** Multicast is a technique used in which a device sends a single packet to multiple destinations simultaneously. This chapter introduces two types of multicast addresses:
  - **Well-known multicast addresses:** These multicast addresses are reserved for predefined groups of devices, such as all-IPv6 nodes and all-IPv6 routers multicast groups.
  - **Solicited-node addresses:** Every unicast address assigned to an interface also has a special multicast address known as a solicited-node multicast address. These multicast addresses are automatically created using a special mapping by prepending the solicited-node multicast prefix ff02:0:0:0:0:1:ff00::/104 to the last 24 bits of the unicast address. IPv6's solicited-node multicast address provides a way to reach every device on the link without all those devices needing to process the contents of the packet.

- **Anycast addresses:** An IPv6 anycast address is an address that can be assigned to more than one interface (typically different devices). In other words, multiple devices can have the same anycast address. A packet sent to an anycast address is routed to the "nearest" interface having that address, according to the router's routing table.

There is no broadcast address in IPv6. Instead, IPv6 uses multicast addresses such as the solicited-node multicast and all-IPv6 devices multicast.

Review Questions

1. Convert the following IPv6 address to its most compressed format, using the RFC 5952 standard for multiple strings of all-0s hextets:

   2001:0db8:cab0:0234:0034:0004:0000:0000
2. Convert the following IPv6 address to its most compressed format, using the RFC 5952 standard for multiple strings of all-0s hextets:

   2001:0db8:0cab:0000:0000:0000:0001:0000
3. Convert the following IPv6 address to its most compressed format, using the RFC 5952 standard for multiple strings of all-0s hextets:

   2001:0db8:0cab:1234:0230:1200:0034:0000
4. Convert the following IPv6 address to its most compressed format, using the RFC 5952 standard for multiple strings of all-0s hextets:

   fd00:0000:0000:0000:1234:0000:0000:0000
5. Convert the following IPv6 address to its most compressed format, using the RFC 5952 standard for multiple strings of all-0s hextets:

   2001:0db8:0000:0000:1234:0000:0000:1000
6. Convert this compressed IPv6 address to the complete address with 32 hexadecimal digits:

   2001:db8:cab::1
7. Convert this compressed IPv6 address to the complete address with 32 hexadecimal digits:

   2001:db8:0:0:234::
8. What is the prefix for the address 2001:db8:80f:f425::230/64?
9. What is the prefix for the address 2001:db8:80f:f425:250:56ff:fe83:ecc/64?
10. What is the prefix for the address fe80::250:56ff:fe83:ecc/64?
11. What is the prefix for the address 2001:db8:80f:f425:250:56ff:fe83:ecc/48?
12. What is the prefix for the address 2001:db8:80f:f425::230/48?
13. What is the prefix for the address 2001:db8:bb8a:f390::1/32?
14. What are the three fields in a global unicast address?
15. What is the range of the first hextet of a global unicast address?
16. Which type of address is required for a device to be IPv6-enabled?
17. What is the range of the first hextet of a link-local unicast address?

18. What are three characteristics of a link-local unicast address?
19. What unicast address is an all-0s address?
20. What are two characteristics of an unspecified unicast address?
21. What type of IPv6 unicast address is similar to IPv4 private addresses?
22. What is the range of the first hextet of a unique local address?
23. What is the difference between IPv6 unique local addresses and IPv4 private addresses in terms of NAT?
24. What are the first two hexadecimal digits in a multicast address?
25. What multicast address that is used in address resolution with IPv6 is similar to ARP with IPv4?

References

Endnote

1. R. Hinden, "IP Next Generation Overview," *Communications of the ACM*, Volume 39, Issue 6, June 1996, pp. 61–71.

RFCs

RFC 1546, *Host Anycasting Service*, C. Partridge, www.ietf.org/rfc/rfc1543.txt, November 1993.

RFC 1918, *Address Allocation for Private Internets*, Y. Rekhter, Cisco Systems, www.ietf.org/rfc/rfc1918.txt, February 1996.

RFC 2373, *IP Version 6 Addressing Architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc2373.txt, July 1998.

RFC 2374, *An IPv6 Aggregatable Global Unicast Address Format*, R. Hinden, Nokia, www.ietf.org/rfc/rfc2374.txt, July 1998.

RFC 2375, *IPv6 Multicast Address Assignments*, R. Hinden, Ipsilon Networks, www.ietf.org/rfc/rfc2375.txt, July 1998.

RFC 2526, *Reserved IPv6 Subnet Anycast Addresses*, D. Johnson, Carnegie Mellon University, www.ietf.org/rfc/rfc2526.txt, March 1998.

RFC 2993, *Architectural Implications of NAT*, T. Hain, Microsoft, www.ietf.org/rfc/rfc2993.txt, November 2000.

RFC 3306, *Unicast-Prefix-Based IPv6 Multicast Addresses*, B. Haberman, www.ietf.org/rfc/rfc3306.txt, August 2002.

RFC 3513, *Internet Protocol Version 6 (IPv6) Addressing Architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc3513.txt, April 2003.

RFC 3587, *IPv6 Global Unicast Address Format*, R. Hinden, Nokia, www.ietf.org/rfc/rfc3587.txt, March 2005.

RFC 4038 *Application Aspects of IPv6 Transition*, M-K Shin, ETRI/NIST, www.ietf.org/rfc/rfc4038.txt, August 2003.

RFC 4193, *Unique Local IPv6 Unicast Addresses*, R. Hinden, Nokia, www.ietf.org/rfc/rfc4193.txt, October 2005.

RFC 4291, *IP Version 6 Addressing Architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc4291.txt, February 2006.

RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*, Y. Narten, IMB, www.ietf.org/rfc/rfc4861.txt, September 2007.

RFC 4864, *Local Network Protection for IPv6*, G. Van de Velde, www.ietf.org/rfc/rfc4864.txt, May 2007.

RFC 5902, *IAB Thoughts on IPv6 Network Address Translation*, D. Thaler, www.ietf.org/rfc/rfc5902.txt, July 2010.

RFC 5952, *A Recommendation for IPv6 Address Text Representation*, S. Kawamura, NEC Biglobe, Ltd., www.ietf.org/rfc/rfc5952.txt, August 2010.

RFC 6296, *IPv6-to-IPv6 Network Prefix Translation*, M. Wasserman, Painless Security, www.ietf.org/rfc/rfc6296.txt, June 2011.

IPv6-to-IPv6 Network Address Translation (NAT66), draft-mrw-behave-nat66-02.txt, M. Wasserman, Sandstorm Enterprises, tools.ietf.org/html/draft-mrw-behave-nat66-02, November 2008.

Websites

IANA, *Internet Protocol Version 6 Address Space*, www.iana.org/assignments/ipv6-address-space/ipv6-address-space.txt

IANA, *IPv6 Global Unicast Address Assignments*, www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xml

Ed Horley's blog, www.howfunky.com, an excellent resource for IPv6.

Book

Practical IPv6 for Windows Administrators, by Ed Horley, Apress, December 2013.

This page intentionally left blank

Chapter 5

Global Unicast Address

Global unicast addresses (GUAs) are globally routable and reachable in the IPv6 Internet; they are equivalent to public IPv4 addresses. GUA addresses are also known as *aggregatable global unicast addresses*, as mentioned in Chapter 4, "IPv6 Address Representation and Address Types." The GUA address is one of several types of IPv6 unicast addresses shown in Figure 5-1.

This chapter elaborates on the following points mentioned in Chapter 4:

- GUA addresses are equivalent to public IPv4 addresses.
- The address range for GUA addresses is 2000::/3.
- There are three parts to a GUA address:
  - Global Routing Prefix
  - Subnet ID
  - Interface ID
- A device can be configured with a global unicast address in these ways:
  - Manual configuration
  - Stateless Address Autoconfiguration (SLAAC)
  - Stateful DHCPv6

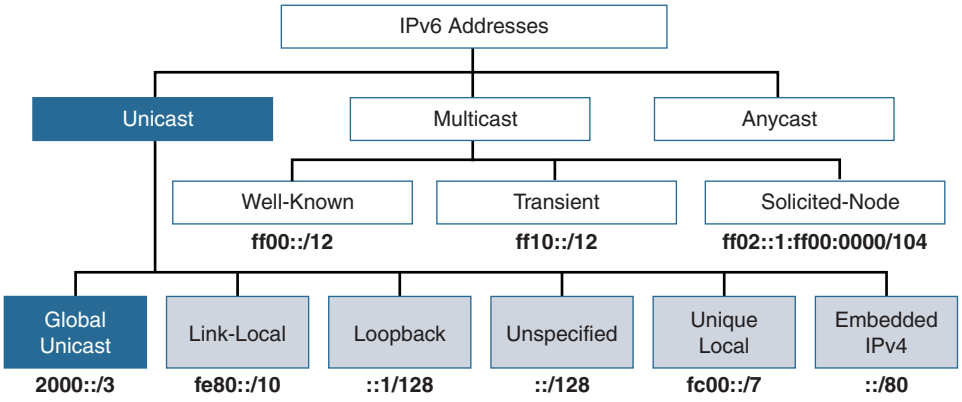


Figure 5-1 *IPv6 Global Unicast Address*

This chapter takes a closer look at these and additional topics. We will begin by examining the structure of a global unicast address and how to manually configure a GUA address on Cisco IOS, Windows, Linux, and Mac OS.

This chapter shows a simple technique you can use to easily recognize the different parts of a GUA address for /48 prefixes, which I call the 3–1–4 rule. You will see that it is much easier to differentiate the prefix (network), subnet, and Interface ID (host portion) of an IPv6 address than in an IPv4 address.

We will see how the Subnet ID makes subnetting an IPv6 GUA address much simpler than subnetting IPv4. We will also see how to subnet beyond the Subnet ID by using bits from the Interface ID.

We will discuss the role of the Global Routing Prefix and the difference between provider-aggregatable (PA) and provider-independent (PI) address space. We will see how the IPv6 general prefix option can make readdressing on Cisco IOS easier.

The end of this chapter provides a brief overview of how a device can dynamically receive or create a GUA address using Stateless Address Autoconfiguration (SLAAC) or stateful DHCPv6. SLAAC and DHCPv6 are discussed more thoroughly in later chapters.

Structure of a Global Unicast Address

Figure 5-2 shows the generic structure of a global unicast address, which includes Global Routing Prefix, Subnet ID, and Interface ID.

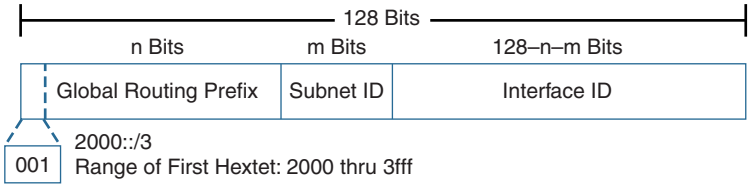


Figure 5-2 *Structure of a Global Unicast Address*

The Internet Corporation for Assigned Names and Numbers (ICANN), the operator for the Internet Assigned Numbers Authority (IANA), allocates IPv6 address blocks to the five Regional Internet Registries (RIRs). The current global unicast address assignment from IANA begins with binary value 001, or the prefix 2000::/3. This results in a range of global unicast addresses of 2000::/3 through 3fff::/16.

How do you get this range of addresses using the 2000::/3 prefix? The /3 prefix length implies that only the first 3 bits are significant in matching the prefix 2000. The first 3 bits of the first hexadecimal value 2 are **001x**. The fourth bit, x, is insignificant and can be either 0 or 1. This results in the first hextet being a 2 (**0010**) or a 3 (**0011**). The remaining 24 bits in the hextet (16-bit segment) can be 0 or 1. This is illustrated in Table 5-1.

Table 5-1 *Range of Global Unicast Addresses*

Global Unicast Address (Hexadecimal)	Range of First Hextet	Range of First Hextet in Binary
2000::/3	2000 to	0010 0000 0000 0000
	3fff	0011 1111 1111 1111

Currently, IANA limits global unicast address assignments to the range of 2000::/3, only one-eighth of the total IPv6 address space. IANA assignments from this block are registered in the IANA registry *IPv6 Global Unicast Address Assignments*; see www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xml. If you look at the registry, you will notice that the RIRs also have prefixes shorter than /23, giving them even more addresses to allocate to Internet service providers (ISPs). We examine prefix lengths in more detail in a moment.

A global unicast address is configured on an interface, which can be configured with one or multiple GUA addresses. The GUA addresses can be on the same or different subnets, and they can be configured manually or obtained dynamically. It is not unusual to see multiple GUA addresses on a client OS device such as a Windows host. Chapter 9, "Stateless Address Autoconfiguration (SLAAC)," discusses why this occurs and how to limit an interface to a single GUA if desired.

So, which address is used when an interface has multiple GUA addresses and also a link-local address? This issue is addressed in RFC 3484, *Default Address Selection for Internet Protocol Version 6 (IPv6)*, and is discussed in Chapter 9.

Note Maybe you can't wait until Chapter 9 to find out why your computer may have multiple GUA addresses. There are two possible reasons. First, your host may create its own GUA address using SLAAC and may receive a different address using stateful DHCPv6. Second, when your host operating system creates a GUA address using SLAAC, it may create an additional temporary IPv6 GUA address. Once again, Chapter 9 explores these issues and shows how to manage them.

It's important to remember that an IPv6 interface does not have to be configured with a global unicast address but it must have a link-local address. In other words, if an interface has a global unicast address, it also has a link-local address. However, if an interface has a link-local address, it does not necessarily have to have a global unicast address.

Global Routing Prefix

Remember that a global unicast address has a Global Routing Prefix, Subnet ID, and Interface ID. It all begins with the Global Routing Prefix.

The global routing prefix is the prefix or network portion of the address assigned by the provider, such as an ISP, to a customer or site. This is a site's prefix or network address, as seen by the provider (that is, the ISP). In some cases this prefix, the Global Routing Prefix, may be propagated to other ISPs or autonomous systems (ASs) throughout the Internet. In other cases, the prefix may be summarized along with other prefixes similar to IPv4 prefixes.

Although the Internet Engineering Steering Group (IESG) and Internet Architecture Board (IAB) no longer recommend specific prefix lengths for networks of different sizes, it is still common for RIRs such as ARIN to have policies for end sites. The American Registry for Internet Numbers (ARIN) has a policy that all sites, regardless of size, should have at least a 48-bit global routing prefix (/48). An *end site* in this context is any public or private organization, company, school, home, or any other institution that needs IP connectivity to the Internet.

Note A good definition of the term *site* in the context of this chapter is available in ARIN's *Number Resource Policy Manual*, Section 6.5.8.2, at www.arin.net/policy/nrpm.html. RFC 6177, *IPv6 Address Assignment to End Sites*, replaced (obsoleted) RFC 3177, *IAB/IESG Recommendations on IPv6 Address Allocations to Sites*.

Figure 5-3 shows a typical site address with a /48 Global Routing Prefix. It is highly recommended that you use a 64-bit Interface ID for most networks. This is because SLAAC uses 64 bits for the Interface ID, which leaves 16 bits for the Subnet ID.

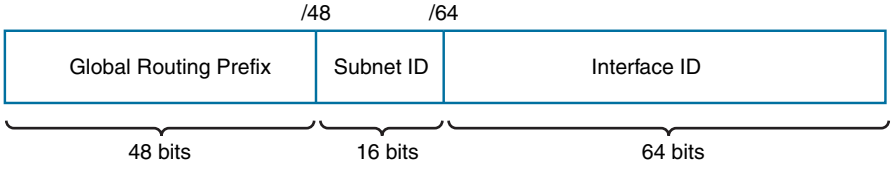


Figure 5-3 *Structure of a GUA for a Typical Site*

Note RFC 4291, *IP Version 6 Addressing Architecture*, does not specify the size of the Subnet ID. The 16-bit Subnet ID in Figure 5-3 results from a site receiving a /48 Global Routing Prefix. With a 64-bit Interface ID, this leaves 16 bits for the Subnet ID.

Later in this chapter we will examine more closely how the size of the Global Routing Prefix and a 64-bit Interface ID determine the number of bits in the Subnet ID. We will also discuss how an organization gets an IPv6 prefix (Global Routing Prefix) either from an ISP, known as provider-aggregatable (PA) address space, or from an RIR, known as provider-independent (PI) address space.

Subnet ID

A big difference between IPv4 and IPv6 addresses is the addition of a Subnet ID in an IPv6 GUA address. In IPv4, you have to borrow bits from the host portion of the address to create subnets. There are no bits designated for the subnet portion of the address because subnetting was an afterthought in IPv4. With IPv6, the Subnet ID is a separate field, apart from the host portion of the address.

As shown in Figure 5-3, a /48 Global Routing Prefix with a 64-bit Interface ID results in a 16-bit Subnet ID. This allows 65,536 individual subnets. Just in case you're wondering… yes, you can use the all-0s and the all-1s subnets. However, using the all-0s and all-1s subnets is typically not recommended. These two subnets can be confused with higher aggregation boundaries and cause headaches for desktop support. The Subnet ID makes subnetting easy! The joy of subnetting is discussed in a later section in this chapter—and you'll see that it really is easier in IPv6!

The prefix length is the total length of the prefix (network portion) of the address, which includes both the Global Routing Prefix and Subnet ID. For example, the address shown in Figure 5-3 has a /64 prefix length, also known as the subnet prefix. The subnet prefix refers to the Global Routing Prefix and the Subnet ID addressing bits.

Interface ID

The Interface ID uniquely identifies the interface on the subnet. As shown earlier, the 64-bit Interface ID allows 18,446,744,073,709,551,616 (18 quintillion) addresses for each subnet! The term *Interface ID* is used rather than *Host ID* because, as mentioned previously, a single host can have multiple interfaces, each having one or more IPv6 addresses.

In most cases, a 64-bit Interface ID should be used for LANs and other networks, including point-to-point links. (We will discuss whether /64 prefixes should be used on point-to-point links later in this chapter, when we discuss subnetting.) The reason /64 prefixes should be used for user networks (LANs) is that SLAAC uses 64 bits to automatically create an Interface ID. SLAAC allows a device to create its own GUA address without the services of DHCPv6 and is discussed in Chapter 9.

There are other reasons to use a 64-bit Interface ID; for example, to make subnetting easier. For those who have been involved with IPv4 for any number of years, it may seem strange—even wasteful—to use 64 bits for an Interface ID. Are 18 quintillion addresses really necessary on a LAN with only a few devices? Obviously no, but remember that with IPv6, the focus of an addressing plan no longer has to be conserving addresses. Instead, an addressing plan can focus on how to organize and manage subnets, without having to find the right balance of the number of subnets versus the number of hosts per subnet for a limited number of IPv4 addresses.

With a typical site having a Global Routing Prefix of /48 and a 16-bit Subnet ID this gives 65,536 subnets, each with 18 quintillion devices (per subnet)! It's okay that you won't be using all of them! You're not cutting down binary trees or anything like that. This is the benefit of a 128-bit address space with its 340 undecillion addresses.

By the way, it is possible to subnet into the Interface ID. Later in this chapter when we discuss subnetting you will see why you might want to do this and how to do it.

Another interesting difference between IPv6 and IPv4 addresses is that all-0s and all-1s addresses are legal IPv6 interface addresses. An IPv6 Interface ID can contain all 0s or all 1s. In IPv4 these addresses typically cannot be used. All 0s in the host portion of the address is reserved for the network or subnet address. All 1s in the host portion of an IPv4 address indicates a broadcast address. Remember, there is no broadcast address in IPv6.

Note In IPv4, the all-0s and all-1s addresses are used when configuring /31 prefixes for point-to-point links. This is discussed in RFC 3021, *Using 31-Bit Prefixes on IPv4 Point-to-Point Links*.

Note Although all-0s or all-1s interface addresses are permitted in IPv6, all 0s shouldn't be used to avoid confusion with prefix addresses and subnet-router anycast addresses. Packets sent to the subnet-router anycast address would be delivered to the router on the subnet. The all 1s is generally not used just so we don't get confused thinking this is a broadcast address, which doesn't exist in IPv6. With 18 quintillion addresses in the /64 subnet, we really don't need to use these two addresses.

Manual Configuration of a Global Unicast Address

In this section we will see how to manually configure an IPv6 global unicast address on Cisco IOS, Windows, Linux, and Mac OS.

Similarly to IPv4, an interface can be assigned a global unicast address manually or dynamically. Figure 5-4 shows the various configuration options. This chapter focuses on the manual configuration options for a GUA address. Besides the traditional manual

configuration, Cisco IOS provides two additional options: manual configuration using EUI-64 (EUI stands for *Extended Unique Identifier*) and IPv6 unnumbered. In later chapters, beginning in Chapter 8, we discuss dynamic configuration options.

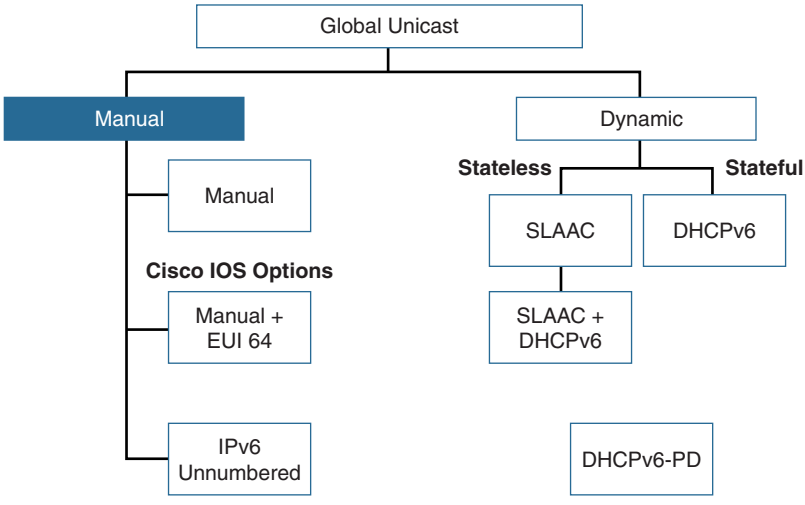


Figure 5-4 *Options for Configuring a GUA*

This section discusses the following ways to manually configure a global unicast address on Cisco IOS:

- **Manual:** This method is similar to configuring a manual or static IPv4 address. The IPv6 address and the prefix length are both configured on the interface.
- **Manual + EUI-64:** With this option, the prefix and prefix length are configured, and the Interface ID is created automatically, using the EUI-64 process.
- **IPv6 unnumbered:** IPv6 unnumbered is the same as with IPv4. It allows an interface to use the IPv6 address of another interface from the same device.

Note The end of this chapter provides an overview of the dynamic configuration options shown in Figure 5-4. SLAAC and DHCPv6 are discussed at length in Chapters 8, "Basics of Dynamic IPv6 Addressing," through 11, "Stateful DHCPv6."

Manual GUA Configuration for Cisco IOS

As with IPv4, it is typically best practice to manually configure IPv6 addresses on router interfaces and servers. The manual (static) configuration of a global unicast address on a Cisco router is similar to the manual configuration of an IPv4 address.

Table 5-2 shows the format of the **ipv6 address** interface command to statically configure a global unicast address. Additional forms and options of this command when configuring other types of addresses are addressed later in this chapter.

Table 5-2 ipv6 address *Command*

Command	Description
Router(config)# **interface** *interface-type -interface-number*	Specifies the interface type and interface number.
Router(config-if)# **ipv6 address** *ipv6-address/prefix-length*	Specifies the IPv6 address and prefix length to be assigned to the interface. To remove the address from the interface, use the **no** form of this command.

Figure 5-5 shows the topology used in this chapter. There are four IPv6 networks:

- 2001:db8:cafe:1::/64
- 2001:db8:cafe:2::/64
- 2001:db8:cafe:3::/64
- 2001:db8:cafe:4::/64

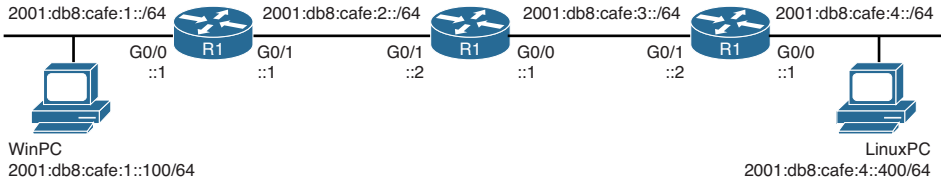


Figure 5-5 *IPv6 Topology*

Example 5-1 shows the commands for configuring the global unicast addresses for Routers R1, R2, and R3. Notice that there is not a space between the prefix and the prefix length when configuring an IPv6 address. Most of the commands that are implemented in IPv6 are similar to those in IPv4, but with **ipv6** substituted for **ip**.

Example 5-1 *Configuring Global Unicast Addresses on Routers R1, R2, and R3*

```
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 address 2001:db8:cafe:1::1/64
R1(config-if)# no shutdown
R1(config-if)# exit
R1(config)# interface gigabitethernet 0/1
R1(config-if)# ipv6 address 2001:db8:cafe:2::1/64
R1(config-if)# no shutdown
--------------------------------------------------------------------------------
```

```
R2(config)# interface gigabitethernet 0/1
R2(config-if)# ipv6 address 2001:db8:cafe:2::2/64
R2(config-if)# no shutdown
R2(config-if)# exit
R2(config)# interface gigabitethernet 0/0
R2(config-if)# ipv6 address 2001:db8:cafe:3::1/64
R1(config-if)# no shutdown
--------------------------------------------------------------------------------
R3(config)# interface gigabitethernet 0/1
R3(config-if)# ipv6 address 2001:db8:cafe:3::2/64
R3(config-if)# no shutdown
R3(config-if)# exit
R3(config)# interface gigabitethernet 0/0
R3(config-if)# ipv6 address 2001:db8:cafe:4::1/64
R1(config-if)# no shutdown
```

It is important to note that we have not yet configured the **ipv6 unicast-routing** global configuration command on any of the routers. This command will be used and explained shortly.

Example 5-2 shows the running configuration (running-config) for router R1. The **no ip address** output refers to the lack of an IPv4 address. Even if you do not use an abbreviated IPv6 address when configuring the interface, the running-config displays the compressed format. You can have both an IPv6 address and an IPv4 address configured on the same interface. This is known as *dual stack*.

Example 5-2 **show running-config** *Command on Router R1*

```
R1# show running-config
<output omitted for brevity>
!
interface GigabitEthernet0/0
 no ip address
 duplex auto
 speed auto
 ipv6 address 2001:DB8:CAFE:1::1/64
!
interface GigabitEthernet0/1
 no ip address
 duplex auto
 speed auto
 ipv6 address 2001:DB8:CAFE:2::1/64
!
```

Note Although RFC 5952 recommends the use of lowercase characters for representing IPv6 addresses, notice that Cisco IOS currently uses uppercase.

Example 5-3 shows the output from the **show ipv6 interface brief** command. This output is similar to that of the **show ip interface brief** command used for IPv4. Notice that the line protocol and status for each of the interfaces is **up/up**.

Example 5-3 **show ipv6 interface brief** *Command on Router R1*

```
R1# show ipv6 interface brief
GigabitEthernet0/0     [up/up]
    FE80::5AAC:78FF:FE93:DA00     ! Link-local address
    2001:DB8:CAFE:1::1            ! Global unicast address
GigabitEthernet0/1     [up/up]
    FE80::5AAC:78FF:FE93:DA01     ! Link-local address
    2001:DB8:CAFE:2::1            ! Global unicast address
R1#
```

Although only one IPv6 address is configured per interface, each interface has two IPv6 addresses. The addresses that begin with fe80: are link-local unicast addresses. A link-local address is automatically configured on the interface whenever a global unicast address is assigned. Remember, an IPv6 interface must have a link-local address and the link-local address can only communicate with other devices on the same link.

Note Cisco IOS uses the EUI-64 method to create the Interface ID and appends it to the fe80::/64 prefix to create the link-local address. The link-local address looks a little ugly to read, but Chapter 6, "Link-Local Unicast Address," shows how to manually configure a link-local address on router interfaces to make them easier to recognize and remember.

Example 5-4 shows the output from the **show ipv6 interface gigabitethernet 0/0** command. Notice the global unicast address was manually configured along with a link-local address that was automatically included. Also, notice that there is a group of other addresses that begin with ff02:, below the header Joined group address(es). These are multicast addresses that the router's interface is automatically a member when the global unicast address was configured. (Multicast addresses are explored in Chapter 7, "Multicast Addresses.") There are also several lines that refer to ND, or Neighbor Discovery, which is part of ICMPv6 ND. (The output from this command is discussed in more detail in Chapters 8 through 11 when discussing dynamic addressing in IPv6. ICMPv6 and ICMPv6 Neighbor Discovery are covered in Chapters 12 and 13.)

Example 5-4 show ipv6 interface gigabitethernet 0/0 *Command on R1*

```
R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::5AAC:78FF:FE93:DA00
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
  Joined group address(es):
    FF02::1
    FF02::FB
    FF02::1:FF00:1
    FF02::1:FF93:DA00
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 30000)
  ND NS retransmit interval is 1000 milliseconds
  Default router is FE80::662:73FF:FE5E:F961 on GigabitEthernet0/1
R1#
```

Manual GUA Configuration with EUI-64 for Cisco IOS

An alternative method to manually configuring a GUA address on Cisco IOS is the EUI-64 option, shown in Example 5-5. With the EUI-64 option, the prefix (network portion) of the address is configured manually, while the EUI-64 process is used to automatically assign the Interface ID.

Note The majority of the time, router interfaces are manually configured with an IPv6 GUA address, without the EUI-64 option. I cover this option here to be comprehensive.

The **ipv6 address** interface command is used with the **eui-64** option as shown here:

```
Router(config-if)# ipv6 address ipv6-prefix/prefix-length eui-64
```

Notice that only the prefix is specified, along with the prefix length. The Interface ID is determined by the EUI-64 process.

The modified EUI-64 process uses the interface's 48-bit Ethernet MAC address and another 16 bits to generate a 64-bit Interface ID. Because the Interface ID is 64 bits, the address requires a /64 subnet prefix.

EUI-64 will be explained in detail in Chapter 6, "Link-Local Unicast Address," where we discuss the link-local address, and again in Chapter 9. For now, remember that the

EUI-64 method uses the interface's 48-bit Ethernet MAC address and inserts 16 more bits (fffe) in the middle. The seventh bit gets flipped, which changes the value of the second hexadecimal digit in the Interface ID.

Example 5-5 shows the configuration and verification of a GUA address with the EUI-64 option. The **show interface gigabitethernet 0/0** command displays the Ethernet MAC address 58ac.7893.da00. Next, we configure R1's gigabitethernet 0/0 interface using the EUI-64 format. The command only requires the prefix 2001:db8:cafe:1:: and the prefix length /64, followed by the **eui-64** option.

Note In Example 5-5, if the **eui-64** option were not included, the Interface ID would consist of all 0s. Although that would be a valid IPv6 interface address, it is not recommended.

Example 5-5 *Configuring a GUA Address with the EUI-64 Option*

```
R1# show interface g 0/0
GigabitEthernet0/0 is up, line protocol is up
  Hardware is CN Gigabit Ethernet, address is 58ac.7893.da00 (bia 58ac.7893.da00)
<output omitted for brevity>

R1(config)# interface g 0/0
R1(config-if)# ipv6 address 2001:db8:cafe:1::/64 ?
  anycast  Configure as an anycast
  eui-64   Use eui-64 interface identifier
  <cr>

R1(config-if)# ipv6 address 2001:db8:cafe:1::/64 eui-64
R1(config-if)# end

R1# show ipv6 interface g 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::5AAC:78FF:FE93:DA00
  No Virtual link-local address(es):
  Global unicast address(es):

    2001:DB8:CAFE:1:5AAC:78FF:FE93:DA00, subnet is 2001:DB8:CAFE:1::/64 [EUI]
<output omitted for brevity>
```

The Interface ID portion was created with the EUI-64 format, which uses the interface's 48-bit Ethernet MAC address 58ac.7893.da00. Notice that the Interface ID has fffe inserted in the middle. This is the additional 16 bits to make the Interface ID a total of 64 bits. The second digit of the Interface ID has been changed from 8 to A. This is due to the seventh bit being flipped as part of the EUI-64 process. (Don't worry; this is explained in Chapter 6.)

To summarize, the subnet prefix was manually configured, and the **eui-64** option was used to create the Interface ID. Prepending the manually configured subnet prefix (2001:db8:cafe:1::/64) to the EUI-64-generated Interface ID (5aac:78ff:fe93:da00) results in the global unicast address 2001:db8:cafe:1:5aac:78ff:fe93:da00. To remove the manual EUI-64 address, you use the interface command **no ipv6 address 2001:db8:cafe:1::/64 eui-64**.

Manual GUA Configuration with IPv6 Unnumbered for Cisco IOS

Another form of manual configuration for global unicast addresses is the **ipv6 unnumbered** command, which enables IPv6 processing on an interface by assigning it a GUA address from another interface.

The **ipv6 unnumbered** *interface* command is used to specify the source address that the interface will use when originating IPv6 packets. The syntax of the **ipv6 unnumbered** command is shown here:

```
Router(config-if)# ipv6 unnumbered interface-type interface-number
```

Note The **ip unnumbered** command in IPv4 is used to help conserve IPv4 address space. Obviously, address space is not a concern in IPv6 networks, so this command will have more limited uses.

Example 5-6 demonstrates using the **ipv6 unnumbered** command on a router that is not part of the topology. The router's serial 0/0/1 interface is configured as unnumbered. IPv6 packets originated by the router that are sent on serial 0/0/1 will use the IPv6 address of gigabitethernet 0/0 as their source address.

Example 5-6 *Example of Using the* **ipv6 unnumbered** *Command*

```
Router(config)# interface gigabitethernet 0/0
Router(config-if)# ipv6 address 2001:db8:abcd:1234::1/64
Router(config)# interface serial 0/0/1
Router(config-if)# ipv6 unnumbered gigabitethernet 0/0
```

Manual GUA Configuration for Windows, Linux, and Mac OS

Continuing with the topology shown in Figure 5-5, let's examine how to manually configure IPv6 global unicast addresses on Windows, Linux, and Mac OSX hosts. The Windows host is running Windows 7 (but it is the same for Windows 8 and 10). The Linux host is running Ubuntu Linux.

Example 5-7 shows the default interface configuration for both WinPC and LinuxPC. We have not yet configured IPv6 global unicast addresses on these hosts, but notice that both hosts have IPv6 link-local addresses. This is because Windows, Linux, and Mac OSX operating systems are IPv6-enabled by default.

Note Notice that the link-local address on the Linux host used EUI-64 to create its Interface ID, while the link-local address on the Windows host used a random 64-bit value. LinuxPC's Interface ID includes its MAC address, with fffe inserted in the middle and a change in the second hexadecimal digit due to the seventh bit being flipped. Although the MAC address for the Windows device is not shown, we can deduce that the Interface ID of the link-local address was randomly created because there is no fffe in the middle of the Interface ID.

Example 5-7 *Viewing the IPv6 Configuration on WinPC and LinuxPC*

```
WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
   Autoconfiguration IPv4 Address  . : 169.254.112.134
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :
------------------------------------------------------------------------
LinuxPC$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:af:14:1b
          inet6 addr: fe80::250:56ff:feaf:141b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:466475604 errors:0 dropped:0 overruns:0 frame:0
          TX packets:403172654 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2574778386 (2.5 GB)  TX bytes:1618367329 (1.6 GB)
          Interrupt:16
```

As mentioned previously, the global configuration command **ipv6 unicast-routing** has not yet been configured on any of the routers. This is significant because this command enables the router as an IPv6 router, and the router will begin sending ICMPv6 Router Advertisement messages. Most host operating systems (Windows, Linux, and Mac OS) by default will obtain their IPv6 addresses dynamically. This means that if you had previously configured the **ipv6 unicast-routing** command on routers R1 and R3, the WinPC and LinuxPC would have created their own global unicast addresses automatically, using SLAAC, which is discussed in Chapter 9.

Since the **ipv6 unicast-routing** command had not been configured on R1 or R3, WinPC and LinuxPC only have link-local addresses. Figure 5-6 shows the manual configuration of IPv6 addressing on WinPC, which includes the following:

- **IPv6 address: 2001:db8:cafe:1::100** – This is the global unicast address for WinPC.
- **Subnet prefix length: /64** – This is the length of the subnet prefix for the GUA address.

- **Default gateway: 2001:db8:cafe:1::1** – This is the local router's GUA address. Although the router's GUA address can be used, it is more common to use the router's link-local address—in this case fe80::5aac:78ff:fe93:da00.

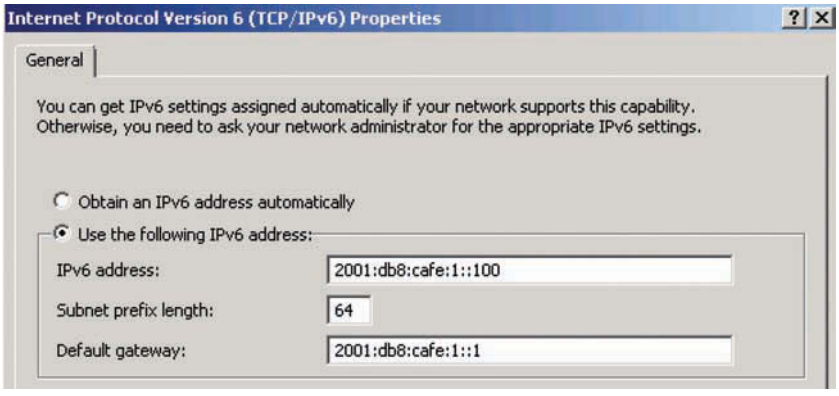


Figure 5-6 *WinPC IPv6 Configuration*

The **ipconfig** command in Example 5-8 verifies the configuration on WinPC. Besides the link-local address, WinPC now has a GUA address and an IPv6 default gateway address.

Example 5-8 *IPv6 Configuration on WinPC*

```
WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1::100
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
   Autoconfiguration IPv4 Address  . : 169.254.112.134
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : 2001:db8:cafe:1::1
```

Example 5-9 shows the configuration and verification of the GUA address and IPv6 default gateway on the LinuxPC, using the Linux shell. Similar to the WinPC configuration, the default gateway address for the LinuxPC is the GUA address of router R3 instead of R3's link-local address.

Note Most implementations of Linux also offer a GUI-based method for configuring network information, including the IPv6 GUA address and default gateway.

Example 5-9 *IPv6 Configuration and Verification on LinuxPC*

```
Configuring the IPv6 global unicast address
LinuxPC$ ifconfig eth0 inet6 add 2001:db8:cafe:4::400/64

Configuring the IPv6 default gateway address
LinuxPC$ route -A inet6 add default gw 2001:db8:cafe:4::1

Verifying the IPv6 global unicast address
LinuxPC$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:af:14:1b
          inet6 addr:0.0.0.6  Bcast:255.255.255.255  Mask:0.0.0.0
          inet6 addr: 2001:db8:cafe:4::400/64 Scope:Global
          inet6 addr: fe80::250:56ff:feaf:141b/64 Scope:Link
<output omitted>

Verifying the IPv6 default gateway
LinuxPC$ ip -6 route show
<output omitted>
default via 2001:db8:cafe:4::1 dev eth0  metric 1
```

Although not part of the topology, Figure 5-7 shows an example of configuring the IPv6 address information on Mac OSX. Mac OSX is a UNIX operating system and for those who prefer the command-line interface, the shell commands are similar to the Linux commands in Example 5-9.

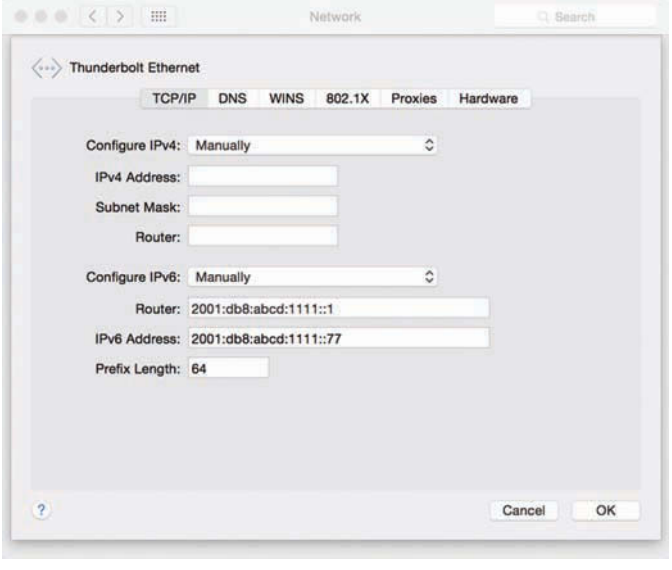


Figure 5-7 *Mac OSX IPv6 Configuration Example*

Implementing Static Routing and Verifying Connectivity with Ping

The final step in this section is to verify reachability between WinPC and LinuxPC in the topology shown in Figure 5-5. But first our network with three routers and four IPv6 networks requires some sort of routing to be implemented.

Example 5-10 shows static routing implemented on routers R1, R2, and R3. To establish reachability, routers R1, R2, and R3 have been configured as follows:

- **ipv6 unicast-routing:** The command **ipv6 unicast-routing** enables the router as an IPv6 router. This command allows the router to forward IPv6 packets, enables static and dynamic routing, and sends ICMPv6 Router Advertisement messages. The **ipv6 unicast-routing** command is discussed in more detail in later chapters, beginning with Chapter 8, "Basics of Dynamic IPv6 Addressing."
- **Static routing:** Each router has been configured with the appropriate static routes. For now, it is not necessary to understand the syntax for the static routes, although you probably notice the similarity to an IPv4 static route. The route to ::/0 is the default route in IPv6. Static routing is covered in Chapter 14, "IPv6 Routing Table and Static Routes."

Example 5-10 *IPv6 Routing Configuration on R1, R2, and R3*

```
R1(config)# ipv6 unicast-routing
R1(config)# ipv6 route ::/0 2001:db8:cafe:2::2

------------------------------------------------------------------
R2(config)# ipv6 unicast-routing
R2(config)# ipv6 route 2001:db8:cafe:1::/64 2001:db8:cafe:2::1
R2(config)# ipv6 route 2001:DB8:cafe:4::/64 2001:db8:cafe:3::2

------------------------------------------------------------------
R3(config)# ipv6 unicast-routing
R3(config)# ipv6 route ::/0 2001:db8:cafe:3::1
```

Note The **ipconfig** (Windows) or **ifconfig** (Linux) command would now show an additional IPv6 address created by SLAAC. The **ipv6 unicast-routing** command also enabled the router to begin sending ICMPv6 Router Advertisement messages every 200 seconds or upon receipt of a Router Solicitation message. After this command is configured, WinPC and LinuxPC will each use the information in the RA message to create an additional GUA address.

Our network now has complete reachability, as verified with the **ping** command on router R1, WinPC, and LinuxPC in Example 5-11. Notice that both Cisco IOS and Windows OS use the same **ping** command used for IPv4. However, Linux requires the use of the **ping6** command for pinging an IPv6 address.

Example 5-11 *Verifying Connectivity on Router R1, WinPC, and LinuxPC*

```
R1# ping 2001:db8:cafe:4::400
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:4::400, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
R1#

------------------------------------------------------------------
WinPC> ping 2001:db8:cafe:4::400
Pinging 2001:db8:cafe:4::400 with 32 bytes of data:
Reply from 2001:db8:cafe:4::400: time=8ms
Reply from 2001:db8:cafe:4::400: time=1ms
Reply from 2001:db8:cafe:4::400: time=1ms
Reply from 2001:db8:cafe:4::400: time=1ms
Ping statistics for 2001:db8:cafe:4::400:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 8ms, Average = 2ms

------------------------------------------------------------------
LinuxPC$ ping6 2001:db8:cafe:1::100
PING 2001:db8:cafe:1::100(2001:db8:cafe:1::100) 56 data bytes
64 bytes from 2001:db8:cafe:1::100: icmp_seq=1 ttl=61 time=6.07 ms
64 bytes from 2001:db8:cafe:1::100: icmp_seq=2 ttl=61 time=1.11 ms
64 bytes from 2001:db8:cafe:1::100: icmp_seq=3 ttl=61 time=1.14 ms
64 bytes from 2001:db8:cafe:1::100: icmp_seq=4 ttl=61 time=1.15 ms
^C
4 packets transmitted, 4 received, 0% packet loss, time 3003ms rtt min/avg/max/
  mdev = 1.114/2.374/6.078/2.138 ms
```

Recognizing the Parts of a GUA Address and the 3–1–4 Rule

Similar to IPv4, the IPv6 prefix length determines the number of subnets and devices available for the network. End sites, as defined previously, receive a prefix and prefix length from their provider—an ISP or an RIR.

It is common for end sites receiving their IPv6 address from an ISP to get a /48 prefix. However, as mentioned previously, an end site may receive a prefix length of any size, as determined by the provider. ARIN, the RIR for North America, has a current policy of recommending that in most cases end sites should receive a minimum /48. Therefore, this section uses the /48 prefix as a starting point to show how easy it is to break down and understand the different parts of an IPv6 global unicast address.

IPv6 global unicast addresses sometimes look complicated, and it can be difficult to recognize all the parts. A simple technique that I use to quickly break down a /48 prefix is the 3–1–4 rule, as shown in Figure 5-8. (This is not any type of official rule, but something I created.) Each number refers to the number of hextets, or 16-bit segments, of that portion of the address. Perhaps an easy way to remember these numbers is to call it the *pi rule* (pi = 3.14).

- **3:** This indicates the three hextets, or 48 bits, of the Global Routing Prefix. In this example, the Global Routing Prefix is the first three hextets, 2001:db8:cafe.
- **1:** This indicates the one hextet, or 16 bits, of the Subnet ID. The Subnet ID in this example is 0001, or 1. A 16-bit Subnet ID results in 65,536 subnets.
- **4:** This indicates the four hextets, or 64 bits, of the Interface ID. A 64-bit Interface ID is recommended for most end user networks to accommodate SLAAC and make the addressing plan easier to manage. A 64-bit Interface ID gives us 18 quintillion devices per subnet. The Interface ID in this example is 0000:0000:0000:0100, or ::1, with the double colon (::) representing the three all-0s hextets.

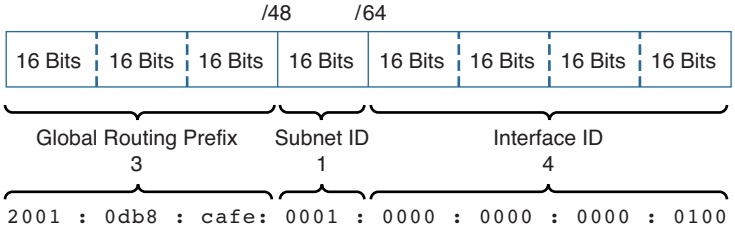


Figure 5-8 *Global Unicast Addresses and the 3–1–4 Rule*

Table 5-3 shows several examples of /48 global unicast addresses using the 3–1–4 technique. The addresses are shown in preferred and compressed formats to show the parts more clearly. Although the double colon compresses the notation of the address, it can sometimes make it more difficult to recognize the three parts of the address. Sometimes it can be easier to start from the Interface ID or work from both ends toward the Subnet ID in the middle.

Table 5-3 *Examples of /48 Global Unicast Addresses with the 3–1–4 Technique*

/48 Global Unicast Address	Global Routing Prefix (3)	Subnet ID (1)	Interface ID (4)
2001:0db8:cafe:0001:0000:0000:0000:0001	2001:db8:cafe	0001	0000:0000:0000:0001
2001:0db8:cafe:0004:0000:0000:0000:0400	2001:db8:cafe	0004	0000:0000:0000:0400
2001:0db8:1234:0001:0000:0000:0000:0100	2001:db8:1234	0001	0000:0000:0000:0100
2001:db8:aaaa:9:0:0:0:a	2001:db8:aaaa	0009	0000:0000:0000:000a

/48 Global Unicast Address	Global Routing Prefix (3)	Subnet ID (1)	Interface ID (4)
2001:db8:aaaa:1::0200	2001:db8:aaaa	0001	0000:0000:0000:0200
2001:db8:aaaa::200	2001:db8:aaaa	0000	0000:0000:0000:0200
2001:db8::abc:0	2001:db8:0000	0000	0000:0000:0abc:0000
2001:db8:abc:1::	2001:db8:abc	0001	0000:0000:0000:0000
2001:db8:deed:450:0123:4567:89ab:cdef	2001:db8:deed	0450	0123:4567:89ab:cdef
2001:db8:deed:5:ffff:ffff:ffff:ffff	2001:db8:deed	0005	ffff:ffff:ffff:ffff

Notice that both of the following addresses are legal interface (host) addresses in IPv6:

- **All-0s address:** 2001:db8:abc:1::, or 2001:0db8:0abc:0001:0000:0000:0000:0000
- **All-1s address:** 2001:db8:deed:5:ffff:ffff:ffff:ffff

Obviously the 3–1–4 rule works only for /48 prefixes, a common prefix allocated to sites by an ISP.

Examining Other Prefix Lengths

Because prefixes are almost always allocated on a nibble (hexadecimal digit) boundary, with IPv6 it is easy to differentiate the Subnet ID from the Interface ID. Figure 5-9 shows examples of different prefix allocations. All these examples use a 64-bit Interface ID, which is recommended for most end-user subnets.

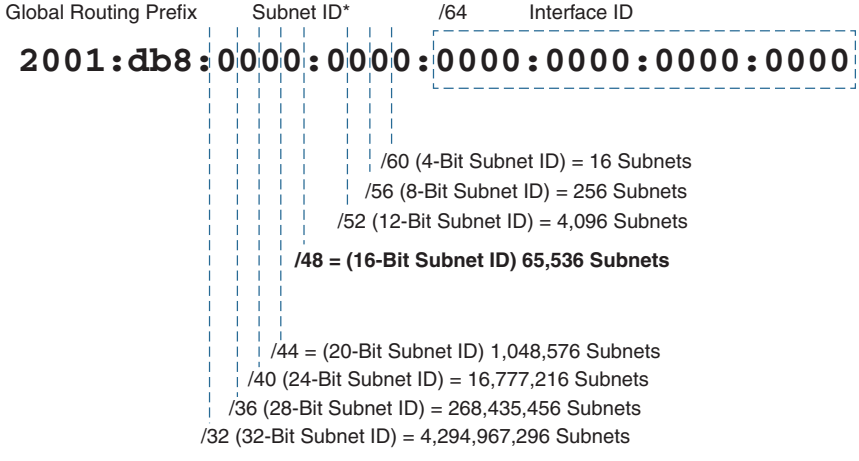


Figure 5-9 *Prefix Lengths and Subnet IDs*

Let's look at the /32 prefix. A site may receive a /32 prefix from a provider or, more typically, directly from an RIR. A /32 prefix results in the following:

- **Global Routing Prefix:** This is the first 32 bits, or first two hextets. The network address or prefix as seen from the provider would be 2001:db8::/32.
- **Subnet ID:** With a 64-bit Interface ID this leaves 32 bits, the next two hextets, for the Subnet ID. Showing all the digits in the Subnet ID, the subnets would range from 2001:db8:0000:0000::/64 to 2001:db8:ffff:ffff::/64. A 32-bit Subnet ID gives us 4.29 billion subnets, each subnet having 18 quintillion devices! This single site would have as many subnets as the total number of IPv4 addresses in the Internet!

This may seem like an extraordinarily large network that only very large organizations might receive, but that is not necessarily the case. For example, the University of California Santa Cruz received a /32 prefix from ARIN.

Figure 5-9 shows several other examples of the Subnet ID falling on a nibble (hexadecimal digit) boundary. This makes it very easy to differentiate the subnet portion of an address from the Global Routing Prefix and Interface ID. (We examine what happens when we subnet within a nibble in the next section, on subnetting.)

Just to make sure this is clear, let's take a look at the /52 prefix 2001:db8:face:1000::/52, shown in Figure 5-10.

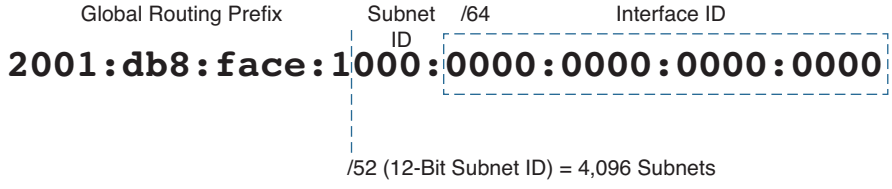


Figure 5-10 *Example of a /52 Prefix*

A /52 Global Routing Prefix with a 64-bit Interface ID leaves 3 hexadecimal digits for the Subnet ID. The provider would see this site in its routing table as 2001:db8:face:1000::/52. Internally, the site would have up to 4096 /64 subnets, ranging from 2001:db8:face:1000::/64 to 2001:db8:face:1fff::/64.

Subnetting IPv6

Depending on the size of the environment, developing an IPv6 addressing plan can require substantial planning. However, basic subnetting of an IPv6 address is very straightforward. In many ways, it is much simpler than subnetting an IPv4 address. With IPv4, unless you are subnetting on a natural octet boundary, the specific subnets are not usually obvious.

RFC 5375, *IPv6 Unicast Address Assignment Considerations*, offers guidelines for subnet prefix considerations. (IPv6 address plans are discussed in Chapter 17,

"Deploying IPv6 in the Network.") RIRs and other sources also offer assistance in developing an IPv6 addressing plan:

- ARIN's *IPv6 Addressing Plans*, visit www.getipv6.info/index.php/IPv6_Addressing_Plans.
- RIPE's *Preparing an IPv6 Addressing Plan*, see labs.ripe.net/Members/steffann/preparing-an-ipv6-addressing-plan.
- IPv6 Forum's *Preparing an IPv6 Addressing Plan*, go to www.ipv6forum.org/dl/presentations/IPv6-addressing-plan-howto.pdf.

Remember that the term *Subnet ID* refers to the contents of the field used to allocate individual subnets and can vary in size, depending on the size of the prefix and Interface ID. The *Subnet prefix* (or sometimes just *prefix*) shown in Figure 5-11 refers to the Global Routing Prefix and the Subnet ID addressing bits.

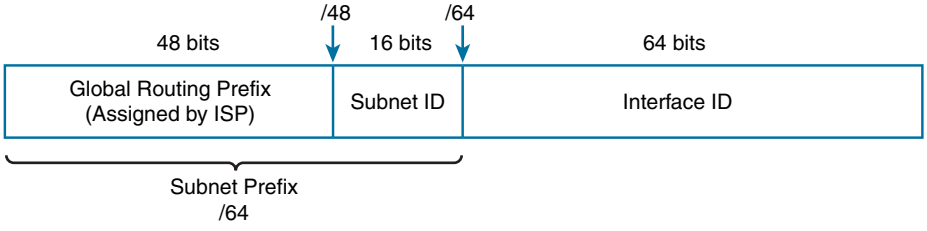


Figure 5-11 *Subnet Prefix*

A common IPv6 site prefix is /48, assigned by the provider–usually an ISP but can also be an RIR. This creates a 16-bit Subnet ID, allowing 2^{16}, or 65,536, subnets. Remember that the all-0s and all-1s subnets are valid subnets. This leaves 64 bits for the Interface ID, yielding 2^{64}, or 18 quintillion, devices per subnet, as shown in Figure 5-12.

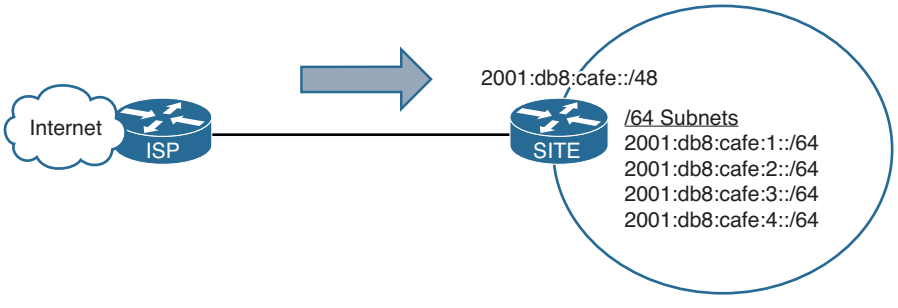


Figure 5-12 *Example of a /48 Site Subnetted with Four /64 Subnets*

Using the topology in Figure 5-12, the 2001:db8:cafe::/48 network has been subnetted internally using four /64 subnets, as shown in Table 5-4. The all-0s subnet can be used, but this example starts with 0001 for clarity.

Table 5-4 *IPv6 Address Chart for 2001:db8:cafe::/64*

Subnet Prefix (Global Routing Prefix and Subnet ID)		
Global Routing Prefix	**Subnet ID**	**Compressed Format**
2001:db8:cafe:	0001	2001:db8:cafe:1::/64
2001:db8:cafe:	0002	2001:db8:cafe:2::/64
2001:db8:cafe:	0003	2001:db8:cafe:3::/64
2001:db8:cafe:	0004	2001:db8:cafe:4::/64

With a 16-bit Subnet ID, the values can range from 0000 to ffff, which provides 65,536 total subnets. Subnetting is easy because we start with 0000 and just increment by 1 in hexadecimal. Remember, after 0009, the next Subnet ID value would be 000a. The simplicity of subnetting using the 16-bit Subnet ID is illustrated in Table 5-5.

Table 5-5 *Subnetting Using the 16-Bit Subnet ID*

Range	Subnet ID
First 16 subnets	0000
	0001
	0002
	…
	0009
	000a
	000b
	000c
	000d
	000e
	000f
Next 16 subnets	0010
	0011
	0012
	…
	001f
Next 16 subnets	0020
	0021
	0022
	…

Range	Subnet ID
	002f
…	0030
	0031
	…

Extending the Subnet Prefix

Subnetting is not limited to a 16-bit Subnet ID. The Subnet ID can be variable in length. In other words, any number of bits after the Global Routing Prefix can be chosen for the Subnet ID. Just as with IPv4, if you want to extend the number of subnets or—more likely in IPv6—reduce the number of hosts per subnet, you must borrow bits from the Interface ID. It is important to note that best practice dictates that this should be done only on network infrastructure links. Any segment that includes end systems should have a /64 prefix, which is required for supporting SLAAC.

Figure 5-13 shows an example of a /112 prefix, extending the original /48 prefix by 64 bits (four hextets). Figure 5-14 shows the range of subnet prefixes using a /112 prefix length.

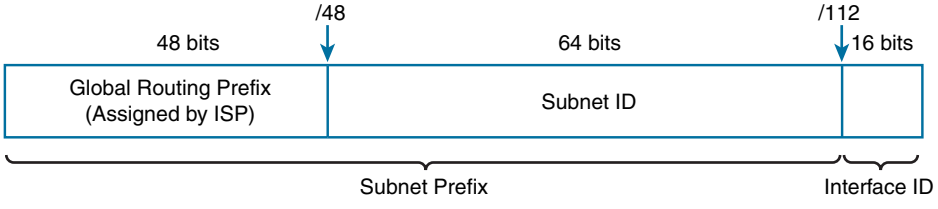


Figure 5-13 */112 Subnet Prefix*

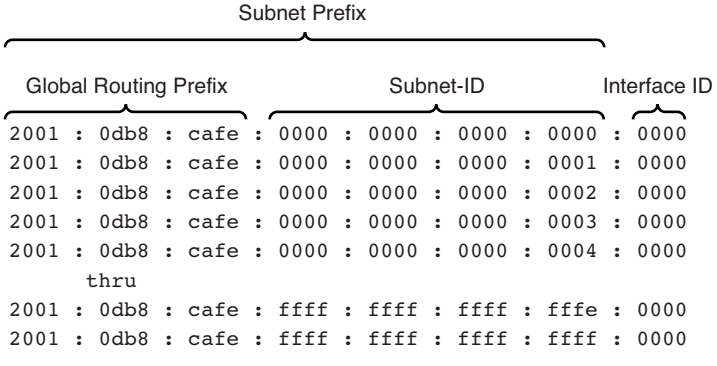


Figure 5-14 *Range of* /112 *Subnet Prefixes*

Even with the extension of the Subnet ID, subnetting is very straightforward, as long as you subnet on a nibble boundary.

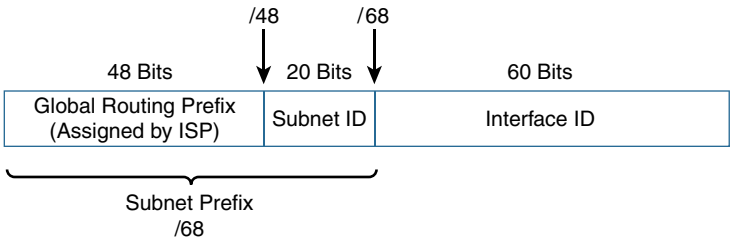
Subnetting on a Nibble Boundary

If you are going to extend the Subnet ID, which means using bits from the Interface ID, it is best practice to subnet on a nibble boundary. A nibble is 4 bits, as shown in Table 5-6.

Table 5-6 *Decimal, Hexadecimal, and Binary Chart*

Decimal	Hexadecimal	Binary (Nibble)
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	a	1010
11	b	1011
12	c	1100
13	d	1101
14	e	1110
15	f	1111

In Figure 5-15, the /64 subnet prefix is extended by 4 bits, or one nibble, to a /68. This increases the Subnet ID from 16 bits to 20 bits. This in turn allows more subnets but reduces the size of the Interface ID. In this case, there isn't any practical reason for doing this except to illustrate the concept. By extending the subnet prefix by 4 bits, or one full nibble, we are implementing the best practice of subnetting on a nibble boundary. Using 20 bits, a factor of 4 bits, makes it very easy to list the subnets (see Figure 5-15).



Subnetting on a nibble (4 bit) boundary makes it easier to list the subnets.

```
2001:db8:cafe:0000:0000::/68
2001:db8:cafe:0000:1000::/68
2001:db8:cafe:0000:2000::/68
  thru
2001:db8:cafe:ffff:f000::/68
```

Figure 5-15 *Subnetting on a Nibble Boundary*

Subnetting Within a Nibble

For most customer networks, subnetting within a nibble is not recommended. It provides few, if any, benefits and makes implementation and troubleshooting more difficult. However, in some cases, subnetting on a nibble is potentially wasteful, and it is beneficial to subnet within the 4-bit nibble. The addressing plans discussed previously help address some of those instances.

Subnetting within a nibble is a little more challenging. Figure 5-16 shows an example of subnetting within a nibble using a /70 prefix. Because it is extended by only 2 bits instead of a nibble (4 bits), the conversion is a little more troublesome. Of course, it is perfectly valid; it is just cumbersome to convert.

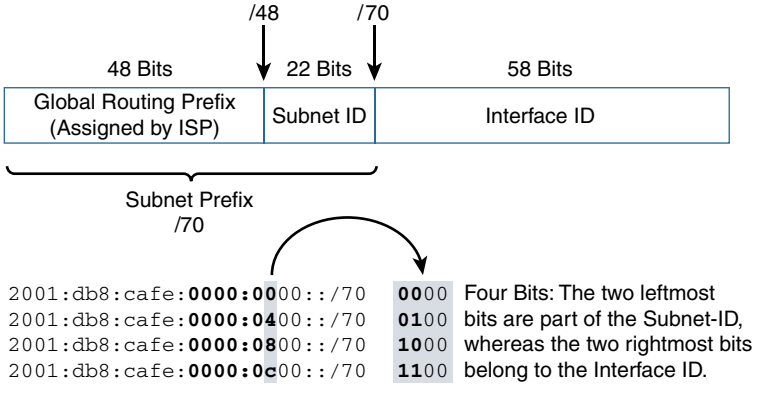


Figure 5-16 *Subnetting Within a Nibble*

The first subnet is easy enough to figure out, but right away, you can see that the second subnet requires a little more thought. IPv6 addresses use hexadecimal values to represent each 4 bits. With a /70 subnet prefix, the first half of the last hexadecimal digit belongs to the Subnet ID and the other half belongs to the Interface ID. So, only the first 2 bits of the last digit of the Subnet ID are modified, as shown in Table 5-7.

Table 5-7 *Subnetting Within a Nibble*

Subnetting Within a Nibble	Last Digit of Subnet ID Binary to Hexadecimal
2001:db8:cafe:**0000:00**00::/70	**00**00 = 0
2001:db8:cafe:**0000:04**00::/70	**01**00 = 4
2001:db8:cafe:**0000:08**00::/70	**10**00 = 8
2001:db8:cafe:**0000:0c**00::/70	**11**00 = c

Subnetting /127 Point-to-Point Links

Although IPv6 address space is plentiful, there can be reasons for limiting the size of the Interface ID within a network infrastructure. There is some debate about whether limiting the size of the Interface ID on some networks is necessary. This section includes a brief explanation of the topic.

NDP Exhaustion Attack

Jeff Wheeler's paper "IPv6 NDP Table Exhaustion Attack" brought to light a potential problem with /64 subnets and having such a large IPv6 address space on a single network. For example, an attacker could send a continuous stream of packets to a router from hundreds of millions of fake source IPv6 addresses. This would cause the device to create a neighbor cache entry for each address and consume large amounts of memory. Depending on the type of packet sent, it might cause the device to respond with a large number of Neighbor Solicitation packets that will never receive replies, thereby consuming large amounts of memory and processing. This is the equivalent to filling a device's ARP cache in IPv4.

Note Jeff Wheeler's presentation can be downloaded from inconcepts.biz/~jsw/IPv6_NDP_Exhaustion.pdf.

A 64-bit Interface ID allows an abundance of interface addresses—more than 18 quintillion interfaces (hosts) per subnet. An IPv4 host has an Address Resolution Protocol (ARP) cache to maintain a list of IPv4 addresses and their relative Layer 2 MAC addresses. In IPv6, there is something similar: Neighbor Cache.

There are mitigation techniques for dealing with an NDP exhaustion attack. Cisco routers internally manage their Neighbor Cache that defends against this kind of attack. Cisco IOS IPv6 destination guard is another mitigation technique that blocks data traffic from an unknown source and filters IPv6 traffic based on the destination address.

RFC 3756, *IPv6 Neighbor Discovery (ND) Trust Models and Threats*, proposes some techniques to address the NDP exhaustion attack issue, such as Secure Neighbor Discovery (SEND or SeND). SEND has not been widely implemented or supported by vendors. There is a draft RFC, *Mitigating IPv6 Neighbor Discovery DoS Attack Using Stateless Neighbor Presence Discovery*, that also addresses this issue.

The details of the NDP exhaustion attack and mitigation techniques are beyond the scope of this book. I have included several links in the Reference section at the end this chapter about this issue.

/127 Subnetting on Point-to-Point Links

One way to defend against NDP exhaustion attack is by using a /127 prefix on point-to-point links. RFC 6164, *Using 127-Bit IPv6 Prefixes on Inter-Router Links*, recommends employing 127-bit IPv6 prefixes (/127) on inter-router point-to-point links for security and other reasons.

Subnetting using a /127 prefix is similar to using a 31-bit (/31) prefix in IPv4. To conserve IPv4 address space, a /30 or a /31 prefix can be used on point-to-point links. It has been a long-time practice on many networks to use a /30 prefix, which gives you four total addresses with two host addresses, a network address, and a broadcast address.

A /31 subnet in effect doubles the number of point-to-point links over /30 subnets. Cisco IOS has supported /31 subnets since 12.2(2)T; see RFC 3021, *Using 31-Bit Prefixes on IPv4 Point-to-Point Links*.

Note Both ends of a /31 point-to-point link are configured using the network address for the interface IPv4 address. For more information and configuration examples, go to www.cisco.com.

An IPv6 /127 subnet is similar to an IPv4 /31 subnet except that the all-0s and all-1s Interface ID (the host portion of the address) is a legitimate address, and there is no broadcast address in IPv6. That said, Figure 5-17 shows an example of an IPv6 /127 prefix, using a 48-bit Global Routing Prefix. The sizes of the three fields in the figure are:

- **Global Routing Prefix:** First 48 bits
- **Subnet ID:** 79 bits
- **Interface ID:** Last 1 bit

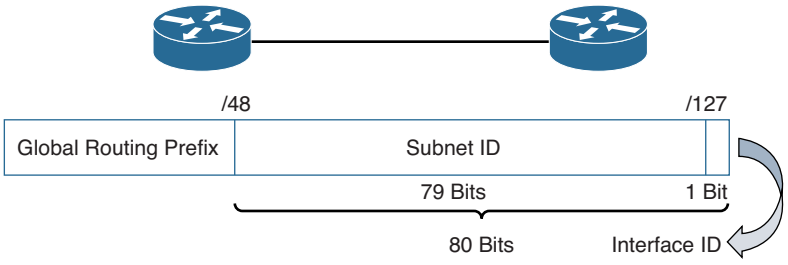


Figure 5-17 */127 Prefix*

Unlike /30 or /31 in IPv4, the purpose of an IPv6 /127 prefix is not to conserve addresses but for security and other reasons. To make address management easier, it is recommended that we allocate an entire /64 subnet for each /127 subnet.

For example, let's use the 2001:db8:cafe::/48 site prefix, which has several point-to-point links. A 16-bit Subnet ID provides 65,536 subnets. We can reserve the Subnet ID ffxx for point-to-point /127 prefixes. Therefore, the first hextet of the /127 networks would be ff00, ff01, ff02, through ffff. The xx gives us 256 /127 subnets.

Table 5-8 shows an example of allocating an entire /64 subnet for each /127 subnet. At first this may seem wasteful, but remember that the /48 Global Routing Prefix gives us 65,536 /64 subnets! The Subnet ID is shown in preferred format for clarity.

Table 5-8 *Allocating a /64 Subnet for Each /127 Subnet*

First Hextet	/64 Subnet Reserved for Each /127 Subnet
ff00	2001:db8:cafe:ff00:0000:0000:0000:0000/64 (Reserved)
	2001:db8:cafe:ff00:0000:0000:0000:0000/127 (First host)
	2001:db8:cafe:ff00:0000:0000:0000:0001/127 (Second host)
ff01	2001:db8:cafe:ff01:0000:0000:0000:0000/64 (Reserved)
	2001:db8:cafe:ff01:0000:0000:0000:0000/127 (First host)
	2001:db8:cafe:ff01:0000:0000:0000:0001/127 (Second host)
ff02	2001:db8:cafe:ff02:0000:0000:0000:0000/64 (Reserved)
	2001:db8:cafe:ff02:0000:0000:0000:0000/127 (First host)
	2001:db8:cafe:ff02:0000:0000:0000:0001/127 (Second host)
.	
.	
.	
ffff	2001:db8:cafe:ffff:0000:0000:0000:0000/64 (Reserved)
	2001:db8:cafe:ffff:0000:0000:0000:0000/127 (First host)
	2001:db8:cafe:ffff:0000:0000:0000:0001/127 (Second host)

Notice in Table 5-8 that the last hextet of the first host address ends in a double colon (::), similar to a network address. This does not present any problems in IPv6 but may be confusing. One alternative is to modify the last 3 bits of the Subnet ID so that neither address has the final hextet ending in a double colon (::).

Figure 5-18 shows an example of modifying the last 3 bits of the Subnet ID to 101 so that neither of the two addresses ends in a double colon (::). The last 3 bits of the Subnet ID (101) plus the last bit, representing the Interface ID (0 or 1), results in the last hextet being either a or b.

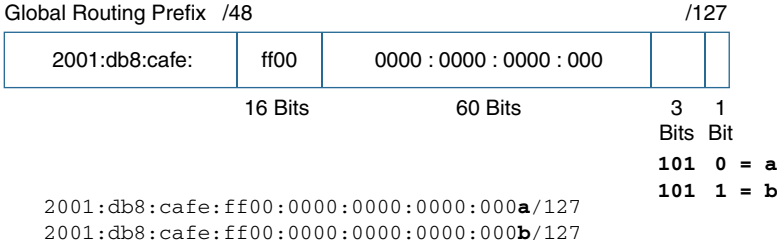


Figure 5-18 */127 Prefix with Last 3 Bits Modified in Subnet ID*

Table 5-9 shows the same /64 subnets with different /127 subnets—this time with 101 as the last 3 bits of the Subnet ID. Care must be taken in which addresses are paired with each other. The addresses 2001:db8:cafe:ff00::a/127 and 2001:db8:cafe:ff00::b/127 are on the same subnet, but 2001:db8:cafe:ff00::b/127 and 2001:db8:cafe:ff00::c/127 are on different subnets. Once again, the Subnet ID is shown in preferred format for clarity.

Example 5-12 shows the /127 configuration on two adjacent routers, R1 and R2.

Table 5-9 */127 Subnets with Modified Last Hextet*

First Hextet	/64 Subnet Reserved for /127 Subnet
ff00	2001:db8:cafe:ff00:0000:0000:0000:0000/64 (Reserved)
	2001:db8:cafe:ff00:0000:0000:0000:000a/127 (First host)
	2001:db8:cafe:ff00:0000:0000:0000:000b/127 (Second host)
ff01	2001:db8:cafe:ff01:0000:0000:0000:0000/64 (Reserved)
	2001:db8:cafe:ff01:0000:0000:0000:000a/127 (First host)
	2001:db8:cafe:ff01:0000:0000:0000:000b/127 (Second host)
ff02	2001:db8:cafe:ff02:0000:0000:0000:0000/64 (Reserved)
	2001:db8:cafe:ff02:0000:0000:0000:000a/127 (First host)
	2001:db8:cafe:ff02:0000:0000:0000:000b/127 (Second host)

First Hextet	/64 Subnet Reserved for /127 Subnet
.	
.	
.	
ffff	2001:db8:cafe:ffff:0000:0000:0000:0000/64 (Reserved)
	2001:db8:cafe:ffff:0000:0000:0000:000a/127 (First host)
	2001:db8:cafe:ffff:0000:0000:0000:000b/127 (Second host)

Example 5-12 *Configuring and Verifying the /127 Subnet*

```
R1(config)# interface g 0/1
R1(config-if)# ipv6 add 2001:db8:cafe:ff02::a/127

---------------------------------------------------------------------
R2(config)# interface g 0/1
R2(config-if)# ipv6 add 2001:db8:cafe:ff02::b/127
R2(config-if)# end
R2# ping 2001:db8:cafe:ff02::a
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:FF02::A, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R2#
```

ipv6gen: An IPv6 Subnetting Tool

As you have seen, subnetting IPv6 is easy—certainly much easier than subnetting IPv4—and there is a nice tool to make it even easier. ipv6gen generates IPv6 subnets for a given prefix length. It generates a list of IPv6 prefixes of a given length from a larger prefix.

Note ipv6gen generates prefixes according to RFC 3531, *A Flexible Method for Managing the Assignment of Bits of an IPv6 Address Block*.

The **ipv6gen** command contains several options. The basic command syntax is:

ipv6gen *prefix/prefix-length subnet-bits*

Example 5-13 shows two examples of using ipv6gen. The first example displays the /64 subnets for the 2001:db8:cafe::/48. Obviously, not all 65,536 subnets are displayed.

Example 5-13 *Using* **ipv6gen** *to Display IPv6 Subnets*

```
MacOS$ ipv6gen.pl 2001:db8:cafe::/48 64
2001:0DB8:CAFE:0000::/64
2001:0DB8:CAFE:0001::/64
2001:0DB8:CAFE:0002::/64
2001:0DB8:CAFE:0003::/64
2001:0DB8:CAFE:0004::/64
2001:0DB8:CAFE:0005::/64
<output omitted for brevity>
2001:0DB8:CAFE:FFFD::/64
2001:0DB8:CAFE:FFFE::/64
2001:0DB8:CAFE:FFFF::/64
MacOS$

--------------------------------------------------------------------------------
MacOS$ ipv6gen.pl 2001:db8:cafe:1000::/52 54
2001:0DB8:CAFE:1000::/54
2001:0DB8:CAFE:1400::/54
2001:0DB8:CAFE:1800::/54
2001:0DB8:CAFE:1C00::/54
MacOS$
```

The second example in Example 5-13 shows the results of subnetting within a nibble. The 2001:db8:cafe:1000::/52 prefix is subnetted using a /54 Subnet ID. This results in the second hexadecimal digit, the first 0 in 1000, being split between the Subnet ID and Interface ID. The 2-bit Subnet ID results in four subnets.

To help you see the subnetting within the nibble, the following list shows the fourth hextet in both hexadecimal and binary:

- Hexadecimal: 1000, Binary: 0001 **00**00 0000 0000
- Hexadecimal: 1400, Binary: 0001 **01**00 0000 0000
- Hexadecimal: 1800, Binary: 0001 **10**00 0000 0000
- Hexadecimal: 1c00, Binary: 0001 **11**00 0000 0000

Note ipv6gen is written in Perl and can be downloaded from code.google.com/archive/p/ipv6gen/downloads.

Prefix Allocation

In IPv6, just as in IPv4, the number of subnets and devices available on a network depends on the prefix length allocated by the provider.

As the IPv4 Internet grew, the limited IPv4 address space quickly became more depleted. Today, customer requests for IPv4 address space are much more scrutinized by the providers. Most sites rely heavily on NAT to accommodate the number of internal IPv4 hosts in their networks.

For example, at the time of this writing the Corporation for Education Network Initiatives in California (CENIC), the ISP for California's educational institutions, includes the following in its policy for allocating IPv4 addresses:

- "Because the number of available IP addresses on the Internet is limited, the utilization rate of address space will be a key factor in network number assignment."
- "CENIC customers may be allocated space no larger than a /27 (32 usable addresses)."
- "Utilization of 85% or greater must be demonstrated before additional space will be allocated."
- "All allocations are subject to CENIC Engineering review."
- "Allocations may be smaller than /27 if this review concludes that the smaller allocation will meet the customer's needs."
- "Customers with requirements for host addresses exceeding the capacity of a /27 should investigate implementing Network Address Translation (NAT)."[1]

Note CENIC's complete IPv4 allocation policy can be viewed at cenic.org/network/ipv4-allocation, and its IPv6 policy is available at cenic.org/network/address-allocation.

With IPv6 there is more than sufficient address space, so there is no need for this type of scrutiny or restriction. Many network administrators have become accustomed to the limited IPv4 address space and using techniques such as NAT for their internal network and /30 prefixes on point-to-point links. This is not a concern with IPv6, and for some can be a difficult habit to break.

IPv6 address allocation to end sites allows for more than enough subnets and devices per subnet. One of the goals is to make it easy for sites to manage their IPv6 addressing schema.

So, what size IPv6 network does an end site receive? The Internet Architecture Board (IAB) at one time recommended that in most cases, providers should assign /48 prefixes to end sites (see RFC 3177, *IAB Recommendations on IPv6 Address Allocations to Sites*). This recommendation has since been rescinded in RFC 6177, *IPv6 Address Assignment to End Sites*.

There are several reasons the IAB initially recommended /48 prefixes to end sites, including to ensure sufficient address space and to simplify the site's address plan. With RFC 6177, /48 prefix allocations to end sites is still an option, but it's no longer a requirement of the IPv6 architecture. Instead, providers can be more flexible in the size of the prefix allocations to end sites. This is keeping in mind that a /64 is needed for a single subnet and that end sites will need multiple subnets.

Even with this change, it is still common for providers to allocate /48 prefixes to their customers. As mentioned previously, ARIN, the RIR for North America, has a current policy recommending that in most cases end sites should receive a minimum /48 from their provider.

Provider-Aggregatable (PA) and Provider-Independent (PI) Address Space

There are two types of addresses that can be assigned to an end-user organization:

- Provider-aggregatable (PA)
- Provider-independent (PI)

Provider-Aggregatable Address Space

Provider-aggregatable (PA) address space is a block of addresses assigned by an RIR to an ISP. This allows the ISP to aggregate (summarize) its address space for more efficient routing. These addresses are then assigned by the ISP to its customers. PA address space has the following characteristics:

- The address belongs to the ISP and is assigned to a customer.
- Cost can be one of the advantages of PA address space. Since the address belongs to the provider, it usually comes with the service, and there is no additional charge.
- The disadvantage is that if an end site changes service providers, the address cannot migrate with the end user.

An example of an end site receiving an IPv6 prefix from PA address space is Cabrillo College in Aptos, California. Cabrillo College received a /48 IPv6 prefix from its ISP, CENIC. CENIC has a policy of allocating /48 prefixes to its IPv6 end sites. Again, a /48 prefix provides 65,536 subnets, each subnet having 18 quintillion devices. However, if Cabrillo College were to change service providers, it would receive a different IPv6 prefix. This could present a problem in terms of renumbering devices, editing ACLs, updating DNS, changing switch and router interfaces, modifying ingress filtering, modifying routing protocols, and more.

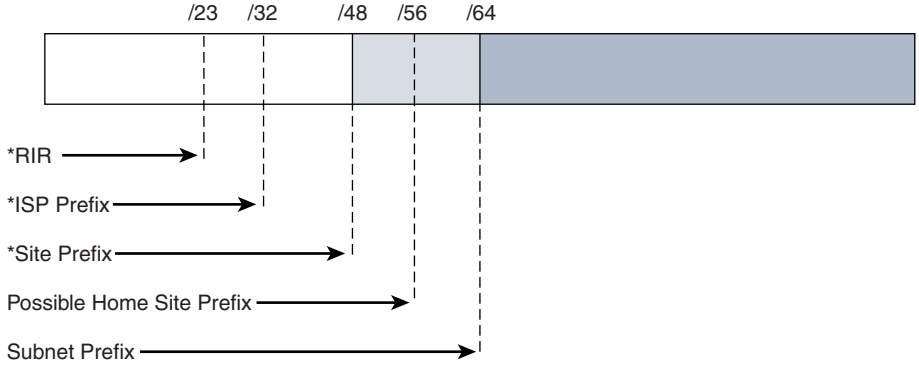
These RFCs provide help in renumbering from one IPv6 prefix to another:

- RFC 4192, *Procedures for Renumbering an IPv6 Network with a Flag Day*
- RFC 6879, *IPv6 Enterprise Networking Renumbering Scenarios, Considerations, and Methods*

Cisco IOS also provides a general prefix option (explained in the next section) that makes renumbering on Cisco devices easier.

Figure 5-19 shows an example of how address space can be allocated to the RIRs, ISPs, and end sites. These examples are minimum allocations, which means an RIR will get a /23

or shorter, an ISP will get a /32 or shorter, and a site will get a /48 or shorter. A shorter prefix length allows more available address space. For example, a site could get a /40 instead of a /48, giving it more addresses if that can be justified to its ISP.



*This is a minimum allocation. The prefix length may be less if it can be justified.

Figure 5-19 *Global Routing Prefix Sizes*

Provider-Independent Address Space

The end customer can also choose a provider-independent address space by going straight to the RIR. PI address space is assigned by an RIR directly to the end-user organization. Provider-independent address space has the following characteristics:

- Address space is assigned by the RIR.
- The site can usually get larger address space.
- PI facilitates an organization using multiple providers that may or may not be multihomed. (Multihoming is the practice of connecting a network to multiple ISPs.)
- The address remains with the customer if the customer changes providers, without requiring prefix renumbering.
- The RIR typically charges for PI address space.

One of the advantages of provider-independent address space is that it allows organizations to change service providers without obtaining new address space. The end site must contract with its ISP or ISPs to ensure the routing of its prefix.

Referring back to the prefix lengths in Figure 5-9, end sites may receive a prefix from /48 to /32, depending on the policies of the RIR. An example of a site getting PI address space is the University of California Santa Cruz (UCSC). UCSC applied to ARIN for IPv6 PI address space and received a /32 prefix. This gives UCSC 4.29 billion subnets, each with 18 quintillion devices. The 32 bits for subnetting allows UCSC to create an addressing plan that is easy to manage and implement.

General Prefix Option

The IPv6 general (or generic) prefix option simplifies network renumbering and allows for automatic prefix definition in Cisco IOS. A *prefix-name* is given to a general (or generic) IPv6 prefix—for example, a /48 prefix. The *prefix-name* is a placeholder for the general IPv6 prefix and can be used instead of the actual prefix when assigning IPv6 addresses to interfaces.

The general prefix is configured in global configuration mode using the following syntax:

```
ipv6 general-prefix prefix-name ipv6-prefix/prefix-length
```

In Example 5-14, the general prefix with the prefix name MyGUA is configured as follows:

```
Router(config)# ipv6 general-prefix MyGUA 2001:db8:cafe::/48
```

MyGUA is a placeholder for the prefix 2001:db8:cafe::/48. The MyGUA *prefix-name* can then be used as a shortcut or an alias when assigning addresses. When using the MyGUA prefix name to configure the **gigabitethernet 0/0** and **0/1** addresses, notice that the first 48 bits are all-0s (::). These 48 bits are substituted with the 48 bits in the *prefix-name* MyGUA.

Example 5-14 *Configuring Addresses with the IPv6 General Prefix Option*

```
Router(config)# ipv6 general-prefix ?
  WORD  General prefix name
Router(config)# ipv6 general-prefix MyGUA 2001:db8:cafe::/48
Router(config)# interface gigabitethernet 0/0
Router(config-if)# ipv6 address MyGUA ::88:0:0:0:1/64
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface gigabitethernet 0/1
Router(config-if)# ipv6 address MyGUA ::99:0:0:0:1/64
Router(config-if)# no shutdown
Router(config-if)# end
Router# show ipv6 interface brief
GigabitEthernet0/0     [up/up]
    FE80::7EAD:74FF:FECC:5380
    2001:DB8:CAFE:88::1
GigabitEthernet0/1     [[up/up]
    FE80::7EAD:74FF:FECC:5381
    2001:DB8:CAFE:99::1
<output omitted>

Router# show ipv6 general-prefix
IPv6 Prefix MyGUA, acquired via Manual configuration
  2001:DB8:CAFE::/48 Valid lifetime infinite, preferred lifetime infinite
Router#
```

The **show ipv6 interface brief** command in Example 5-14 verifies that both interface addresses begin with the first 48 bits, 2001:db8:cafe, the 48 bits from the MyGUA *prefix-name*. The **show ipv6 general-prefix** command displays information for all general prefixes configured.

Example 5-15 illustrates the advantage of using **general-prefix** for prefix renumbering. First, the general prefix name MyGUA and associated prefix are removed. Next, using the same prefix name MyGUA, this general prefix is reconfigured using a new /48 prefix, 2001:db8:beef. The **show ipv6 interface brief** command verifies the renumbering of the interfaces with a new prefix without requiring reconfiguration of every interface. Notice that the running-config file shows the interface configuration using **general-prefix**.

Example 5-15 *Renumbering Using the IPv6* **general-prefix** *Option*

```
! General prefix-name GUA is removed
Router(config)# no ipv6 general-prefix MyGUA 2001:db8:cafe::/48

! General prefix-name GUA is configured with a new prefix
Router(config)# ipv6 general-prefix MyGUA 2001:db8:beef::/48
Router(config-if)# end

! Verify that the interface addresses have the new prefix
Router# show ipv6 interface brief
GigabitEthernet0/0     [up/up]
    FE80::7EAD:74FF:FECC:5380
    2001:DB8:BEEF:88::1
GigabitEthernet0/1     [[up/up]
    FE80::7EAD:74FF:FECC:5381
    2001:DB8:BEEF:99::1
<output omitted>
Router# show running-config
<partial output>
ipv6 general-prefix MyGUA 2001:DB8:BEEF::/48
!
interface GigabitEthernet0/0
 ipv6 address MyGUA ::88:0:0:0:1/64
!
interface GigabitEthernet0/1
 ipv6 address MyGUA ::99:0:0:0:1/64
!
```

Note For more information on the **general-prefix** command, go to www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6_basic/configuration/15-mt/ip6b-15-mt-book/ip6-generic-prefix.html.

Dynamic Addressing Methods with SLAAC and DHCPv6

This section provides a brief overview of dynamic addressing methods for IPv6 global unicast addresses. Dynamic addressing is discussed starting in Chapter 8.

As mentioned in Chapter 2, "IPv6 Primer," IPv6 has a different approach to dynamic address allocation than IPv4. One difference is that IPv6 uses the ICMPv6 Router Advertisement (RA) message to suggest to devices how to obtain their IPv6 addressing information.

The RA message can suggest two ways a device can be configured dynamically with a global unicast address:

- **Stateless Address Autoconfiguration (SLAAC):** A device uses the information in the RA message, including the prefix, to create its own global unicast address. The Interface ID is automatically generated either by using a random 64-bit value or the EUI-64 process, which includes the Ethernet MAC address.
- **Stateful DHCPv6:** Stateful DHCPv6 is similar to DHCP for IPv4. The stateful DHCPv6 server allocates the global unicast address to the client device. The only difference is that the DHCPv6 server does not include a default gateway address. The default gateway address can only be obtained dynamically from the RA message.

Again, both SLAAC and DHCPv6 are discussed at length beginning in Chapter 8.

Summary

This chapter focuses on the IPv6 global unicast address (GUA), subnetting, and prefix allocation. It examines the structure of a global unicast address and how to manually configure a GUA address on Cisco IOS, Windows, Linux, and Mac OSX.

A simple technique for easily recognizing the different parts of a /48 GUA address is the 3–1–4 rule.

The Subnet ID makes subnetting an IPv6 GUA address much simpler than with IPv4. Where necessary, you can subnet beyond the Subnet ID by using bits from the Interface ID. It is always best to subnet on a nibble boundary, but subnetting within a nibble is also allowed. Because of concerns regarding the NDP exhaustion attack, some network administrators prefer to use a /127 prefix on point-to-point links. This chapter provides examples of how the software tool **ipv6gen** can help with subnetting IPv6.

This chapter compares the provider-aggregatable (PA) and provider-independent (PI) address spaces. One of the disadvantages of PA address space is prefix renumbering if the site changes providers. Cisco IOS provides the IPv6 **general-prefix** option to make readdressing on Cisco IOS easier.

This chapter ends with a brief overview of how a device can dynamically receive or create a GUA address using Stateless Address Autoconfiguration (SLAAC) or stateful DHCPv6. SLAAC and DHCPv6 are discussed more thoroughly in Chapters 8 through 11.

Review Questions

1. What are the three parts of a global unicast address?
2. Why is it highly recommended that end user networks (LANs) use a /64 subnet prefix?
3. What three parts of a /48 GUA address does the 3–1–4 rule represent?
4. Identify the Global Routing Prefix, Subnet ID, and Interface ID for the following /48 GUA addressess:
   a. 2001:db8:cafe:1:a:b:c:d
   b. 2001:db8:cafe:a100::2:d
   c. 2001:db8:cafe:a:37::9
   d. 2001:db8:cafe::a:b:c:d
   e. 2001:db8:cafe:1::100
   f. 2001:db8::100:a:b:c:d
   g. 2001:db8::100
5. Subnet the 2001:db8:cafe::/48 prefix using /52. How many subnets does this create? List the subnets.
6. Subnet the 2001:db8:cafe::/48 prefix using /56. How many subnets does this create? List the first three subnets and the last three subnets.
7. With a /64 Interface ID, what length Global Routing Prefix would provide 256 subnets?
8. With a /64 Interface ID, what length Global Routing Prefix would provide 16,777,216 subnets?
9. Which two of the following /127 addresses would be on the same subnet and could be used for a point-to-point link?
   a. 2001:db8:face:b00c::a/127
   b. 2001:db8:face:b000::b/127
   c. 2001:db8:face:b00c::/127
   d. 2001:db8:face:b00c::2/127
   e. 2001:db8:face:b00c::3/127
   f. 2001:db8:face:b00c::4/127
10. What is an advantage of provider-independent address space?

References

Endnote

1. CENIC, *Address Allocation*, cenic.org/network/ipv4-allocation.

RFCs

Mitigating IPv6 Neighbor Discovery DoS Attack Using Stateless Neighbor Presence Discovery, https://tools.ietf.org/id/draft-smith-6man-mitigate-nd-cache-dos-slnd-06.html.

RFC 2374, *An IPv6 Aggregatable Global Unicast Address Format*, R. Hinden, Nokia, www.ietf.org/rfc/rfc2374.txt, July 1998.

RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, Cisco Systems, www.ietf.org/rfc/rfc2460.txt, December 1998.

RFC 3021, *Using 31-Bit Prefixes on IPv4 Point-to-Point Links*, A. Retana, Cisco Systems, www.ietf.org/rfc/rfc3021.txt, December 2000.

RFC 3177, *IAB/IESG Recommendations on IPv6 Addresses*, IAB, www.ietf.org/rfc/rfc3177.txt, September 2001.

RFC 3531, *A Flexible Method for Managing the Assignment of Bits of an IPv6 Address Block*, M. Blanchet, www.ietf.org/rfc/rfc3531.txt, April 2003.

RFC 3587, *IPv6 Global Unicast Address Format*, R. Hinden, Nokia, www.ietf.org/rfc/rfc3587.txt, August 2003.

RFC 3756, *IPv6 Neighbor Discovery (ND) Trust Models and Threats*, P. Nikander, Ericsson Research Nomadic Lab, www.ietf.org/rfc/rfc3756.txt, May 2004.

RFC 4192, *Procedures for Renumbering an IPv6 Network Without a Flag Day*, F. Baker, Cisco Systems, www.ietf.org/rfc/rfc4192.txt, September 2005.

RFC 4291, *IP Version 6 Addressing Architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc4291.txt, February 2006.

RFC 5375, *IPv6 Unicast Address Assignment Considerations*, G. Van de Velde, www.ietf.org/rfc/rfc5375.txt, December 2008.

RFC 5453, *Reserved IPv6 Interface Identifiers*, S. Krishnan, www.ietf.org/rfc/rfc5453.txt, February 2009.

RFC 6164, *Using 127-Bit IPv6 Prefixes on Inter-Router Links*, M. Kohno, Juniper Networks, www.ietf.org/rfc/rfc6164.txt, April 2011.

RFC 6177, *IPv6 Address Assignment to End Sites*, IAB, www.ietf.org/rfc/rfc6177.txt, March 2011.

RFC 6879, *IPv6 Enterprise Network Renumbering Scenarios, Considerations, and Methods*, S. Jiang, www.ietf.org/rfc/rfc6879.txt, February 2013.

Websites

ARIN, *Number Resource Policy Manual*, www.arin.net/policy/nrpm.html

ARIN, *IPv6 Addressing Plans*, www.getipv6.info/index.php/IPv6_Addressing_Plans

Jeff S. Wheeler, *IPv6 NDP Table Exhaustion Attack*, inconcepts.biz/~jsw/IPv6_NDP_Exhaustion.pdf

IPv6 Neighbor Cache Exhaustion Attacks—Risk Assessment & Mitigation Strategies, Part 1, www.insinuator.net/2013/03/ipv6-neighbor-cache-exhaustion-attacks-risk-assessment-mitigation-strategies-part-1/

RIPE, *Preparing an IPv6 Addressing Plan*, labs.ripe.net/Members/steffann/preparing-an-ipv6-addressing-plan

IPv6 Forum, *Preparing an IPv6 Addressing Plan*, www.ipv6forum.org/dl/presentations/IPv6-addressing-plan-howto.pdf

IANA, *IPv6 Global Unicast Address Assignments*, www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xml

This page intentionally left blank

Chapter 6

Link-Local Unicast Address

Link-local addresses (LLA) are unicast addresses that are confined to a single link. The term *link* refers to a network segment or subnet. Therefore, link-local addresses are local only to a particular link or subnet and not routable off the link. Link-local addresses need only be unique on that link because these packets are not routable off the link. This is true of both source and destination link-local addresses. In other words, routers should not forward any packets with link-local source or destination addresses.

Figure 6-1 illustrates link-local addresses as one type of unicast address.

Chapter 4, "IPv6 Address Representation and Address Types," introduces link-local addresses and discusses the following characteristics:

- To be an "IPv6-enabled" device, a device must be able to self-generate a link-local address for each interface that will participate in IPv6. The device doesn't have to have an IPv6 global unicast address, but it must have a link-local address. The link-local address allows the device to have an IPv6 address that it can use to communicate with any other device on its network without the services of DHCP.
- A link is a subnet.
- Link-local addresses are not routable off the link (that is, the IPv6 subnet). Routers must not forward any packets with a source or destination link-local address to other links.
- Link-local addresses only have to be unique on the link. It is very likely and sometimes even desirable for a device to use the same link-local address on different interfaces that are on different links.
- There can be only one link-local address per interface.
- A link-local address can be a source or destination address.

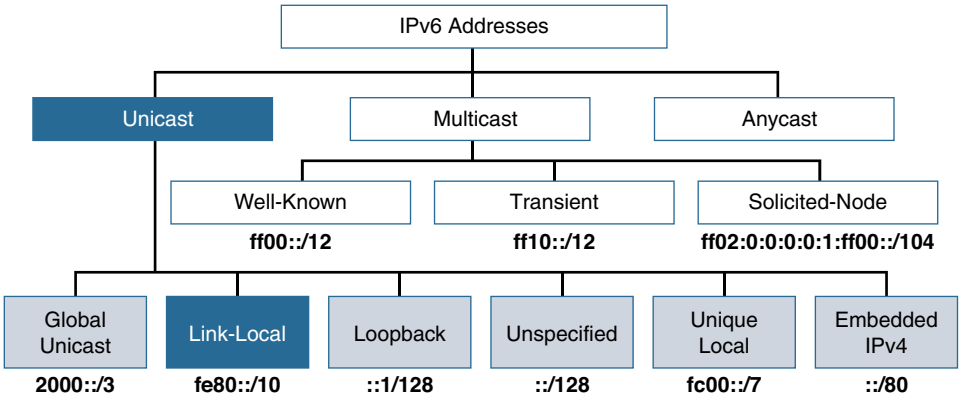


Figure 6-1 *Link-Local Unicast*

The link-local address gives IPv6 unique features that IPv4 doesn't offer. For example, a device can give itself its own IPv6 link-local address upon startup, without a DHCPv6 server. This address can then be used to communicate with any devices on the local subnet, including the local router or a DHCPv6 server, to obtain a global unicast address.

A device could have just a link-local address and no global unicast address. This would be the case with a device that you only want to have communicate with other devices on its own subnet. This may be desirable for things such as printers, entertainment systems, and IoT devices that you only want to be accessible on the local subnet.

As we mentioned in Chapter 1, "Introduction to IPv6," because host operating systems like Windows run IPv6 by default (which means they have a link-local address), it is important to secure networks from potential MITM and DoS attacks based on IPv6 transport. These are similar to rogue DHCP and ARP spoofing threats in IPv4. It's important to secure networks for IPv6 even if IPv6 isn't formally implemented in them. Remember that the client operating systems are already running IPv6, which makes them vulnerable to these types of attacks.

Note Cisco IOS provides IPv6 Router Advertisement Guard, DHCPv6 Guard, and other tools to protect against these types of threats.

As we will see in this chapter and in later chapters, IPv6 link-local addresses have several uses, including the following:

- Before getting a global unicast address, a device uses its link-local address as its source IPv6 address to communicate with the local router and DHCPv6 server to obtain a GUA address.
- Devices dynamically receive their default gateway addresses from the router's ICMPv6 Router Advertisements. The default gateway address is the router's link-local address on that subnet.

- Routers running protocols such as Enhanced Interior Gateway Routing Protocol (EIGRP) for IPv6 and Open Shortest Path First version 3 (OSPFv3) use their link-local addresses when sending routing messages and to establish adjacencies.
- IPv6 routing tables with prefixes learned from dynamic routing protocols use the link-local address as their next-hop address.

In this chapter we will take a closer look at these characteristics and other topics as well. We will begin with closer examination of the structure of a link-local address. You will see how the link-local address is created dynamically or can be configured manually. You will see how to verify and ping a link-local address using Cisco IOS, Windows, Linux, and Mac operating systems. You will also see how a device automatically receives the router's link-local address, which it then uses as its default gateway address.

Structure of a Link-Local Unicast Address

Figure 6-2 shows the format of a link-local unicast address. Link-local unicast addresses are in the range of fe80::/10. The least-significant (rightmost) 64 bits are used as the Interface ID.

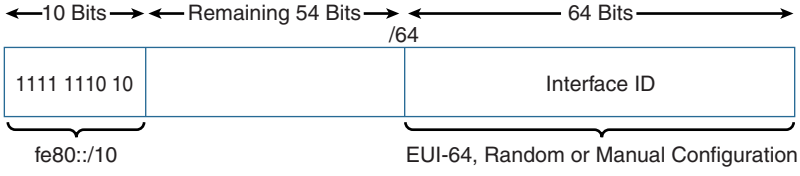


Figure 6-2 *Link-Local Unicast Address*

Using the fe80::/10 prefix results in a range of addresses from fe80::/10 to febf::/10, as illustrated in Table 6-1. The /10 indicates that the first 10 bits are the most significant (left-most) bits, the bits that must match the prefix. The prefix is fe80::/10, or 1111 1110 10 in binary. RFC 4291, *IP Version 6 Addressing Architecture*, doesn't give specific guidance on the remaining 54 bits prior to the Interface ID, but it is best practice for these 54 bits to be all 0s. And although the fe80::/10 prefix allows for the first hextet to be any value in the range of fe80 through febf, in order not to cause potential problems with some operating systems, it is also best practice to use fe80 as the prefix. In other words, it is best practice to use fe80 as the /10 prefix, followed by 54 bits of 0, and any value for the 64-bit Interface ID.

Table 6-1 *Range of Link-Local Unicast Addresses*

Link-Local Unicast Address (Hexadecimal)	Range of First Hextet	Range of First Hextet in Binary
fe80::/10	fe80	**1111 1110 10**00 0000
	febf	**1111 1110 10**11 1111

Automatic Configuration of a Link-Local Address

An IPv6-enabled device must have a link-local address. Windows, Linux, and Mac OS X host operating systems are IPv6-enabled by default and therefore automatically create an IPv6 link-local address upon startup. Cisco IOS automatically creates a link-local address on an interface whenever an IPv6 global unicast address or unique local unicast address is configured on that interface.

Note When a Cisco IOS interface is configured with a global unicast address or unique local unicast address, it automatically creates a link-local address. If the global unicast address or unique local unicast address is deleted, the link-local address is also deleted. The only exception to this is if the `ipv6 enable` interface command has been configured, in which case the link-local address will not be deleted from the interface.

By default, devices automatically create their own link-local unicast addresses. The prefix is typically fe80::/64, followed by a 64-bit Interface ID that is automatically generated in one of two ways:

- **EUI-64 generated:** Used by Cisco IOS, Mac OS X, and Ubuntu Linux (the Linux version used in our examples)
- **Randomly generated:** Used by Windows

Note Cryptographically Generated Addresses (CGA) is a third option and is beyond the scope of this book. The Mac OS examples in this book use Mac OS 10.11. Mac OS 10.12 (Sierra) now uses Cryptographically Generated Addresses (CGA), RFC 3972, to generate the Interface ID. To disable CGA add the command **net.inet6.send.opmode=0** to the /etc/sysctl.conf file and reboot. For more information about CGA, I recommend the book IPv6 Security, by Eric Vynke.

Note As discussed later in this chapter, link-local addresses can also be configured manually.

EUI-64 Generated Interface ID

IEEE defined the Extended Unique Identifier, or *modified EUI-64*, process using the interface's Ethernet MAC address to generate a 64-bit Interface ID. When EUI-64 is used to create a link-local address, the fe80::/64 prefix is prepended to EUI-64 generated Interface ID.

Using the topology in Figure 6-3, we will examine how Cisco IOS uses the EUI-64 process to create link-local addresses on router interfaces.

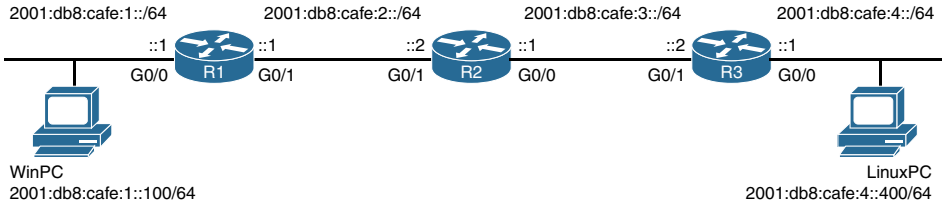


Figure 6-3 *Topology for Link-Local Addresses Example*

Example 6-1 shows Router R1's Ethernet MAC address on GigabitEthernet 0/0, using the **show interface** command. The **show ipv6 interface gigabitethernet 0/0** and the **show ipv6 interface brief gigabitethernet 0/0** commands display the link-local address on GigabitEthernet 0/0. The link-local address was created automatically using the EUI-64 format. You might notice the similarity between the MAC address in the **show interface** command and the link-local address in the **show ipv6 interface** command. This is because the MAC address of GigabitEthernet 0/0 was used to generate the Interface ID of the link-local address using EUI-64.

Example 6-1 *Displaying the Link-Local Address on Router R1*

```
R1# show interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  Hardware is CN Gigabit Ethernet, address is 58ac.7893.da00 (bia 58ac.7893.da00)
<output omitted for brevity>

R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::5AAC:78FF:FE93:DA00
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
<output omitted for brevity>

R1# show ipv6 interface brief gigabitethernet 0/0
GigabitEthernet0/0     [up/up]
    FE80::5AAC:78FF:FE93:DA00
    2001:DB8:CAFE:1::1
R1#
```

Let's take a closer look at the EUI-64 process and how the Ethernet MAC address is used to generate the Interface ID.

An Ethernet MAC address is 48 bits. Router R1's GigabitEthernet 0/0 Ethernet MAC address is shown in Figure 6-4. An Ethernet MAC address is 48 bits, a combination of a 24-bit Organizationally Unique Identifier (OUI) and a 24-bit Device Identifier, written in hexadecimal. Manufacturers of Ethernet network interface cards (NICs) have one or more OUI, or vendor, codes. The 24-bit Device Identifier uniquely identifies the Ethernet NIC

for a given OUI. In other words, appending a 24-bit Device Identifier to a 24-bit OUI uniquely identifies the Ethernet NIC.

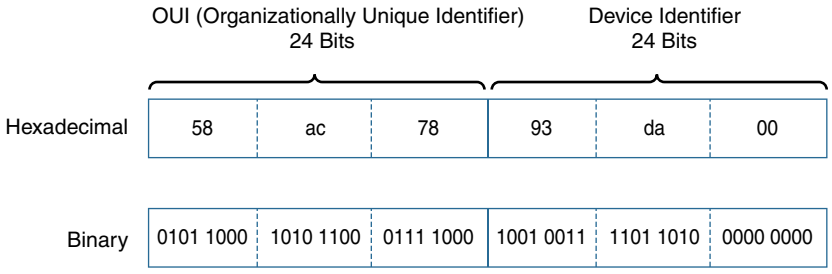


Figure 6-4 *Ethernet MAC Address on GigabitEthernet 0/0*

Note For a list of IEEE OUI codes, go to http://standards.ieee.org/develop/regauth/oui/oui.txt.

The modified EUI-64 process is an insertion of the 16-bit value of fffe between the 24-bit OUI and the 24-bit Device Identifier, with the U/L (Universal/Local) bit *flipped*. This process is accomplished in three simple steps, as illustrated in Figure 6-5:

Step 1. **Spit the MAC Address:** After converting the MAC address to binary, the MAC address is split in the middle, with the 24-bit OUI on the left and the 24-bit Device Identifier on the right.

Step 2. **Insert fffe:** 16 bits, 1111 1111 1111 1110 (fffe in hexadecimal), are inserted between the OUI and the Device Identifier. fffe is an IEEE-reserved value which indicates that the EUI-64 address was generated from a 48-bit MAC address.

Step 3. **Flip the seventh bit:** The Universal/Local (U/L) bit, also known as the Local/Global bit, is the seventh bit of the first byte and is used to determine whether the address is universally or locally administered. IEEE recommends that the U/L bit be 0 for "unique" and 1 for "locally administered."

If the U/L bit is 0, IEEE, through the designation of a unique company ID, has administered the address. If the U/L bit is a 1, the address is locally administered. This means that the network administrator has overridden the manufactured address and specified a different address.

There is historical significance to the U/L bit, and some debate how relevant or useful it is at this point. You can search online for more information on the topic if you are curious about how it all came about. The bottom line is that EUI-64 does flip the U/L bit, although it doesn't have much significance anymore.

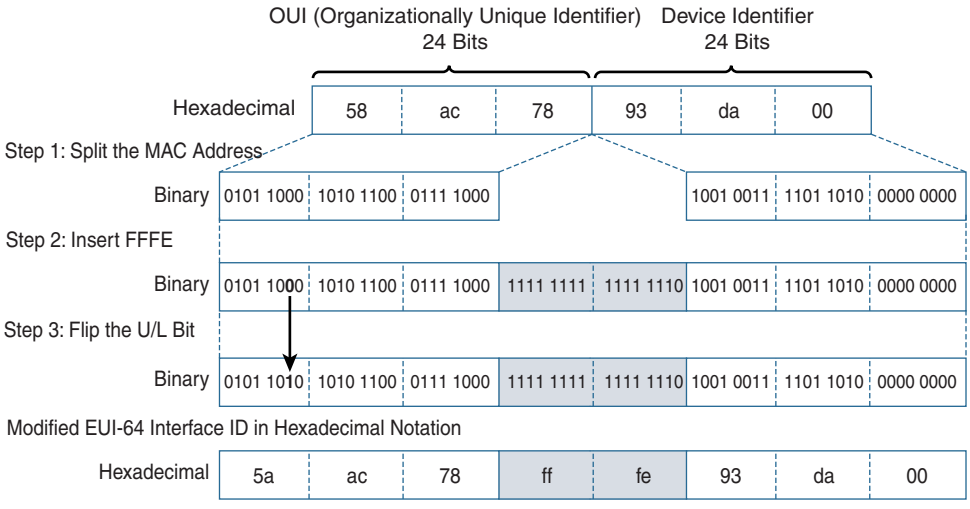


Figure 6-5 *Modified EUI-64 Format on GigabitEthernet 0/0*

There seems to be some conflict as to whether this bit should be flipped from 0 to 1 or from 1 to 0. Some documentation states that the U/L bit should only be modified if it is a 0, flipping it to a 1. However, it seems that Cisco devices will flip this bit regardless of its value. As shown in Figure 6-5, flipping the U/L bit modifies the second hexadecimal value in the Interface ID.

Note RFC7042, *IANA Considerations and IETF Protocol Usage for IEEE 802 Parameters*, discusses the logic behind flipping the U/L bit. The decision was to invert the U/L bit to make it easier for network operators to type in local-scope identifiers.

As shown in Example 6-1, the MAC address of R1's GigabitEthernet 0/0 interface is 58ac.7893.da00. Using modified EUI-64, fffe is inserted between the OUI and the Device Identifier of the MAC address. The seventh bit is flipped from 0 to 1, which changes the second hexadecimal value from 8 to A. As a result, the Interface ID is assigned the 64-bit value 5a-ac-78-ff-fe-93-da-00. The link-local prefix fe80::/64 is prepended to the Interface ID, resulting in a link-local address for R1's GigabitEthernet 0/0 interface of fe80::5aac:78ff:fe93:da00.

Note The IPv6 Interface ID is 64 bits, which also accommodates longer 64-bit MAC addresses. The modified EUI-64 format is used when the interface has a 48-bit MAC address, which is the current standard.

Verifying the Router's Link-Local Address on Ethernet and Serial Interfaces

In Example 6-1 we saw how the **show interface** and **show ipv6 interface** commands verified that the link-local addresses were created using the EUI-64 process for the 64-bit Interface ID. So, how does Cisco IOS create a link-local address using EUI-64 on an interface that doesn't have an Ethernet MAC address, such as a serial interface?

Example 6-2 shows the same router R1, but this time including a serial 0/0/0 interface. Notice that serial 0/0/0 uses the Ethernet MAC address of the GigabitEthernet 0/0 interface to create its link-local address using EUI-64.

Example 6-2 **show ipv6 interface brief** *Command with Serial Interface on Router R1*

```
R1# show ipv6 interface brief
GigabitEthernet0/0     [up/up]
    FE80::5AAC:78FF:FE93:DA00      ! Link-local address
    2001:DB8:CAFE:1::1             ! Global unicast address
GigabitEthernet0/1     [up/up]
    FE80::5AAC:78FF:FE93:DA01      ! Link-local address
    2001:DB8:CAFE:2::1             ! Global unicast address
Serial0/0/0            [up/up]
    FE80::5AAC:78FF:FE93:DA00      ! Link-local address
    2001:DB8:CAFE:99::1            ! Global unicast address
R1#
```

Wait, isn't it problematic to have two interfaces with the same link-local address? No. Remember that link-local addresses only have to be unique on the link. You will see later in this chapter that it is certainly possible to manually configure the same link-local address on all of the router's interfaces. This can make it easier to remember and recognize the link-local address. This can be helpful in recognizing default gateway addresses, analyzing routing table entries, and examining routing protocol messages.

Note Using the same link-local address on all of the router's interfaces is practical when the router has only distribution links and is not used as a default gateway address for devices. This chapter uses a common link-local address for each router for simplicity. In Chapter 17, "Deploying IPv6 in the Network," you will see how to use the VLAN ID as part of the link-local address, which is also used by devices as the default gateway address.

Another example of a link-local address and EUI-64 is on the LinuxPC in Example 6-3. The Linux host, running Ubuntu Linux, uses EUI-64 for generating the Interface ID of the link-local address (at least the version used here; these things are always subject to change). How do you know the link-local address was created using EUI-64? Notice the ff:fe in the middle of the Interface ID. The chances that this was created randomly in this specific location are slim (though of course it could happen). When in doubt, you can search the vendor's documentation to find a definitive answer. Figure 6-6 shows how the EUI-64 process generated the Interface ID for the link-local address on the LinuxPC.

Example 6-3 *Viewing the Link-Local Address on the LinuxPC*

```
LinuxPC$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:af:14:1b
          inet6 addr:0.0.0.6  Bcast:255.255.255.255  Mask:0.0.0.0
          inet6 addr: 2001:db8:cafe:4::400/64 Scope:Global
          inet6 addr: fe80::250:56ff:feaf:141b/64 Scope:Link
<output omitted>
```

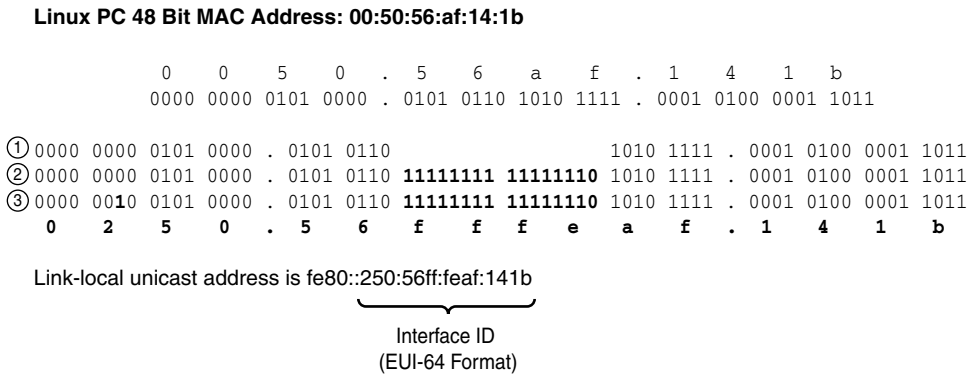


Figure 6-6 *Modified EUI-64 Format on LinuxPC*

Randomly Generated Interface ID

Using EUI-64 is a convenient technique for automatically creating a 64-bit Interface ID from a 48-bit MAC address. However, it introduces a concern for some users: the ability to trace an IPv6 address to the actual physical device using the 48-bit MAC address. To alleviate this privacy concern, devices can use randomly generated 64-bit Interface IDs.

Note Chapter 9, "Stateless Address Autoconfiguration (SLAAC)," discusses privacy concerns in addressing. For more on privacy extensions, see RFC 5375, *IPv6 Unicast Address Assignment Considerations*, and RFC 4941, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*.

Whether a device uses EUI-64 or randomly generated Interface IDs is dependent upon the operating system. You saw that Cisco IOS uses EUI-64, as does Ubuntu Linux. Mac OS X uses EUI-64 for the link-local address. Windows operating systems later than XP use a random number for the Interface ID. Earlier Windows operating systems used EUI-64.

Note The behavior of Windows, Linux, and Mac OS X operating systems in generating their Interface IDs using EUI-64 or a randomized identifier can be modified, as discussed in Chapter 9.

The **ipconfig /all** command in Example 6-4 shows the link-local addresses and the MAC address for WinPC in the topology. Windows hosts generated their own link-local addresses using fe80::/64 and a 64-bit random number for the Interface ID. Two factors that indicate the link-local Interface ID was not generated using EUI-64:

- The link-local address does not contain the Ethernet MAC address (physical address) in the Interface ID.
- The link-local address does not contain fffe in the middle of the Interface ID.

Example 6-4 *IPv6 Configuration on WinPC*

```
WinPC> ipconfig /all
Windows IP Configuration
<output omitted for brevity>

Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   Description: Intel<R> PRO/1000 MT Network Connection
   Physical Address: 00-50-56-AF-97-68
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled:. . . . : Yes
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1::100
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
<output omitted for brevity>
```

Zone ID (%) on Link-Local Interfaces

You might have noticed in Example 6-4 that %11 follows the link-local address. This is known as the Zone ID, or the Scope ID or Interface Scope. Operating systems such as Windows, Linux, and Mac OS use the Zone ID to associate a link-local address with a specific interface. The Zone ID helps determine which interface to use when sending packets destined for a link-local address. This is particularly important when a device has multiple interfaces, each on a separate link (subnet). The Zone ID is only locally significant.

Remember that link-local addresses have to be unique only on that link (subnet). A computer with two interfaces will have two link-local addresses. When sending a packet to a link-local address, the device needs to know on which interface to send that packet. This is where the Zone ID is used to specify the proper exit interface.

Example 6-5 illustrates the use of the Zone ID on a Windows host. This Windows host has two interfaces, an Ethernet NIC and a wireless NIC, both enabled for IPv6. The **ipconfig** command displays the Windows host's two link-local addresses in two separate zones:

- **%11:** The Zone ID for the Ethernet LAN
- **%12:** The Zone ID for the wireless LAN

Notice that both interfaces are on the same link (subnet) and have the same default gateway address. Although interfaces are on the same link (subnet), their Zone IDs differ. The **netsh interface ipv6 show interfaces** command shows the internal index used as the Zone ID. Again, notice the Zone ID of 11 for the Ethernet Local Area Connection and 12 for the wireless network.

Example 6-5 *Windows Host Link-Local Address and Zone ID*

```
Windows-Host> ipconfig

Windows IP Configuration

Wireless LAN adapter Wireless Network Connection:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:face:1::aaaa
   Link-local IPv6 Address . . . . . : fe80::6c51:4f86:ff70:67f5%12
   Default Gateway . . . . . . . . . : fe80::481d:70ff:fe6f:9503%12

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:face:1::bbbb
   Link-local IPv6 Address . . . . . : fe80::9d23:50de:14ce:c8ab%11
   Default Gateway . . . . . . . . . : fe80::481d:70ff:fe6f:9503%11

Windows-Host> netsh interface ipv6 show interfaces

Idx     Met         MTU          State                Name
---  ----------  ----------  ------------  ---------------------------
  1          50  4294967295  connected     Loopback Pseudo-Interface 1
 12          10        1500  connected     Wireless Network Connection
 25          50        1280  disconnected  isatap
 11          10        1500  connected     Local Area Connection
 16          50        1280  disconnected  Teredo Tunneling Pseudo-Interface
Windows-Host>
```

Note Linux and Mac OS X devices use %eth (Linux) and %en (Mac) followed by the Zone ID or Interface Scope value.

Example 6-6 shows examples of this same Windows host pinging the link-local address of the default gateway. Notice that the first two examples are successful pings using the Zone IDs %11 and %12 appended to the link-local address of the destination, the default gateway. The third example is a successful ping to the same address but without including

the Zone ID. The fourth example is a ping to the default gateway's link-local address, but the pings are unsuccessful because of a wrong Zone ID.

Note Linux and Mac OS X both require the use of the Zone ID when pinging a link-local address. This is demonstrated later in this chapter.

Example 6-6 *Windows Host Pinging the Default Gateway Using the Zone ID*

```
Windows-Host> ping fe80::481d:70ff:fe6f:9503%11

Pinging fe80::481d:70ff:fe6f:9503%11 with 32 bytes of data:
Reply from fe80::481d:70ff:fe6f:9503%11: time=2ms
Reply from fe80::481d:70ff:fe6f:9503%11: time=1ms
<output omitted for brevity>

Windows-Host> ping fe80::481d:70ff:fe6f:9503%12

Pinging fe80::481d:70ff:fe6f:9503%12 with 32 bytes of data:
Reply from fe80::481d:70ff:fe6f:9503%12: time=13ms
Reply from fe80::481d:70ff:fe6f:9503%12: time=4ms
<output omitted for brevity>

Windows-Host> ping fe80::481d:70ff:fe6f:9503

Pinging fe80::481d:70ff:fe6f:9503 with 32 bytes of data:
Reply from fe80::481d:70ff:fe6f:9503: time=4ms
Reply from fe80::481d:70ff:fe6f:9503: time=4ms
<output omitted for brevity>

Windows-Host> ping fe80::481d:70ff:fe6f:9503%16

Pinging fe80::481d:70ff:fe6f:9503%16 with 32 bytes of data:
Request timed out.
Request timed out.
<output omitted for brevity>
```

So, when do you need to use the Zone ID? Only when there are multiple interfaces and you are communicating to a link-local address. It also depends on your operating system.

Note For more information regarding the Zone ID, see RFC 4007, *IPv6 Scoped Address Architecture*. A Zone ID is not necessary when there are multiple global unicast address because of the default source address selection process, which is discussed in Chapter 9.

Later in this chapter, you will see an example of pinging a link-local address from a Cisco IOS router. You will see that both the link-local address and exit interface are needed, much like the Zone ID with Windows. You will also see examples of pinging a link-local address from a Windows, Linux, and Mac OS host.

Manual Configuration of a Link-Local Address

Dynamically assigned link-local addresses are ideal for most devices, such as hosts, including clients and servers. So in the majority of cases, there is no need to change the default behavior of having the link-local address automatically created. It might be important to some to use a randomly generated Interface ID rather than EUI-64 for privacy reasons, but there is typically no need to be able to remember or identify the link-local address for end systems such as PCs, servers, temperature sensors, etc.

However, there are certain devices, such as routers, where it is helpful to identify the link-local address to a device. Dynamic routing protocols such as EIGRP for IPv6 and OSPFv3 use the link-local address as the next-hop address in the routing table, for establishing adjacencies, and for other messages. The disadvantage of an automatically generated link-local address is that the long Interface ID (up to 16 hexadecimal digits) is difficult to recognize. It's much easier to use a simpler, manually configured link-local address that is easier to identify.

The command for configuring a static link-local unicast address is as follows:

```
Router(config-if)# ipv6 address ipv6-address link-local
```

Table 6-2 shows the command for configuring a static link-local unicast address. Notice that the **link-local** parameter is required after the *ipv6-address*.

Table 6-2 *Configuring a Static Link-Local Unicast Address*

Command	Description
Router(config)# **interface** *interface-type interface-number*	Specifies the interface type and interface number.
Router(config-if)# **ipv6 address** *ipv6-address* **link-local**	Specifies the IPv6 link-local address. The **link-local** parameter is required.

Figure 6-7 shows the topology with static link-local addresses assigned to each interface. Example 6-7 shows the configuration of these addresses on routers R1, R2, and R3. For each router, the same link-local address is configured for each interface. This makes it easy to recognize the link-local addresses for that router. As mentioned earlier, this is done to keep things simple. It is a practical method on routers with point-to-point links, but there are other options for routers with client-facing links.

In this simple topology, R1 has the Interface ID ::1 on all its interfaces, R2 has the Interface ID ::2 on all its interfaces, and Router R3 has the Interface ID ::3 on all its interfaces. Remember that a link-local address has to be unique only for that link because it is not routable off the link.

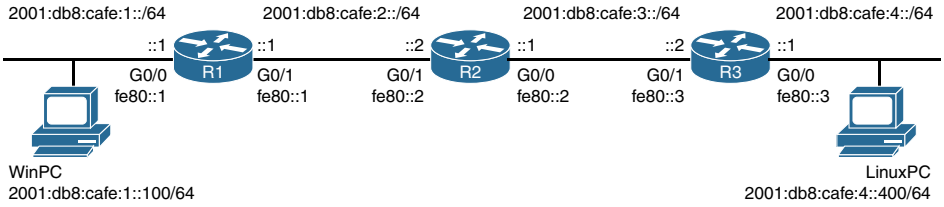


Figure 6-7 *Topology Used to Configure Static Link-Local Addresses*

Example 6-7 *Configuring Static Link-Local Unicast Addresses on R1, R2, and R3*

```
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 address fe80::1 ?
  link-local  Use link-local address

R1(config-if)# ipv6 address fe80::1 link-local
R1(config-if)# exit
R1(config)# interface gigabitethernet 0/1
R1(config-if)# ipv6 address fe80::1 link-local
----------------------------------------
R2(config)# interface gigabitethernet 0/0
R2(config-if)# ipv6 address fe80::2 link-local
R2(config-if)# exit
R2(config)# interface gigabitethernet 0/1
R2(config-if)# ipv6 address fe80::2 link-local
----------------------------------------
R3(config)# interface gigabitethernet 0/0
R3(config-if)# ipv6 address fe80::3 link-local
R3(config-if)# exit
R3(config)# interface gigabitethernet 0/1
R3(config-if)# ipv6 address fe80::3 link-local
```

In Example 6-8, the configuration is verified using the **show ipv6 interface brief** command. Again, notice that for each router, the same link-local address is configured on each interface.

Example 6-8 *Verifying the Static Link-Local Unicast Addresses on R1, R2, and R3*

```
R1# show ipv6 interface brief
GigabitEthernet0/0      [up/up]
    FE80::1
    2001:DB8:CAFE:1::1
GigabitEthernet0/1      [up/up]
    FE80::1
    2001:DB8:CAFE:2::1
```

```
R1#
-------------------------------------------
R2# show ipv6 interface brief
GigabitEthernet0/0     [up/up]
    FE80::2
    2001:DB8:CAFE:3::1
GigabitEthernet0/1     [up/up]
    FE80::2
    2001:DB8:CAFE:2::2
R2#
-------------------------------------------
R3# show ipv6 interface brief
GigabitEthernet0/0     [up/up]
    FE80::3
    2001:DB8:CAFE:4::1
GigabitEthernet0/1     [up/up]
    FE80::3
    2001:DB8:CAFE:3::2
R3#
```

Manual configuration makes it easier to recognize the link-local address and identify the appropriate device. Example 6-9 shows a sample IPv6 routing table. Notice how the manually configured link-local addresses make it easy to identify the next-hop routers fe80::1 for router R1 and fe80::3 for router R3.

Manual configuration of any address moves this address space into the managed address area. These addresses are assigned and need to be tracked by the organization, including ensuring that the addresses don't overlap with addresses that are dynamically assigned.

Note OSPFv3 is used here only to demonstrate the advantage of recognizable link-local addresses. OSPFv3 is discussed in Chapter 16, "OSPFv3."

Example 6-9 *R1's IPv6 Routing Table*

```
R2# show ipv6 route ospf
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity>
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2

O   2001:DB8:CAFE:1::/64 [110/2]
     via FE80::1, GigabitEthernet0/1
O   2001:DB8:CAFE:4::/64 [110/2]
     via FE80::3, GigabitEthernet0/0
R2#
```

Link-Local Address and Duplicate Address Detection

How does a device know whether the IPv6 address it just created for itself is unique? IPv6 uses Duplicate Address Detection (DAD) to ensure the uniqueness on the link. DAD, which is described in RFC 4862, *IPv6 Stateless Address Autoconfiguration*, uses ICMPv6 Neighbor Solicitation messages, which are similar to IPv4 ARP Requests.

DAD is explored in detail in Chapter 13, "ICMPv6 Neighbor Discovery", but the process is fairly straightforward. DAD is nothing more than a device sending its own unicast address to everyone on the link to see whether anyone else is using it. This is similar to a gratuitous ARP Request in IPv4.

In Figure 6-8 WinPC needs to determine whether the link-local address it just created for itself is unique on the link (subnet). WinPC sends a Neighbor Solicitation (NS) message—similar to an IPv4 ARP request—with its own link-local address fe80::d0f8:9ff6:4201:7086 as the target address.

If another device has this address, it responds with a Neighbor Advertisement (NA) message, similar to an ARP reply in IPv4. The NS message is sent to a *solicited node multicast* address. The purpose of a solicited node multicast address is similar to that of a broadcast address, but a solicited node multicast address is much more efficient. (Solicited node multicast addresses are introduced in Chapter 2, "IPv6 Primer," and discussed in more detail in Chapter 7, "Multicast Addresses.")

How a DAD failure (duplicate address) is handled is up to the operating system. If a duplication occurs with a link-local address deriving its Interface ID using EUI-64, it can render the interface unusable. Windows uses a randomized Interface ID. After a DAD failure, it generates a new address. If for some reason (perhaps because a spoofing attack occurs) the DAD failure continues to occur, after a few attempts at generating a new address, Windows displays the network error message "Windows has detected an IP address conflict." Typically with any duplicate address detected, operating systems log the DAD failure and issue a duplicate address detected message.

If WinPC does not hear a response to its Neighbor Solicitation message in the form of a Neighbor Advertisement, it can rest assured that its link-local address is unique.

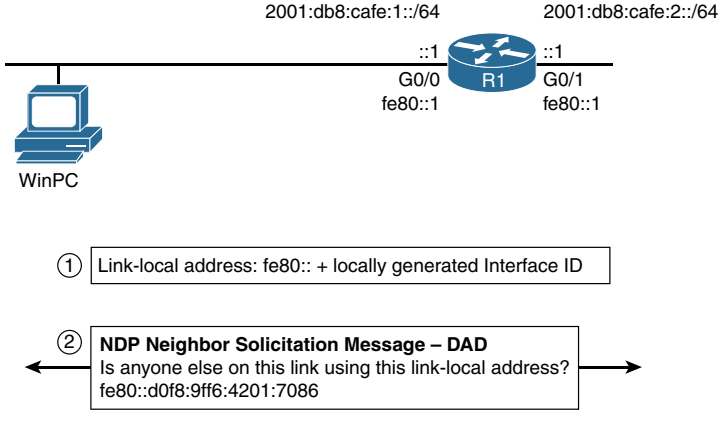


Figure 6-8 *Link-Local Addresses and Duplicate Address Detection*

A device uses Duplicate Address Detection to see whether another device on the link is using a unicast address it is about to use. Duplicate Address Detection is required to be performed for all unicast addresses (global unicast addresses, link-local unicast addresses, etc.) before the address is assigned to an interface, regardless of whether it was obtained through SLAAC, DHCPv6, or manual configuration.

Note There are some exceptions to this behavior, as discussed in RFC 4862 and in RFC 4429, *Optimistic Duplicate Address Detection (DAD) for IPv6*. DAD is not performed on anycast addresses.

If a router is configured with a link-local address that already exists on the link, Cisco IOS displays a warning message that includes the link-local address and the interface with the duplicate address, as shown here:

```
*%IPV6-4-DUPLICATE: Duplicate address FE80::1 on GigabitEthernet0/1
```

The **show ipv6 interface** command provides information about the duplicate address, along with information about whether the IPv6 interface is still available, as shown in this partial output:

```
R2# show ipv6 interface gigabitethernet 0/1
GigabitEthernet0/1 is up, line protocol is up
  IPv6 is stalled, link-local address is FE80::1 [DUP]
<output omitted for brevity>
```

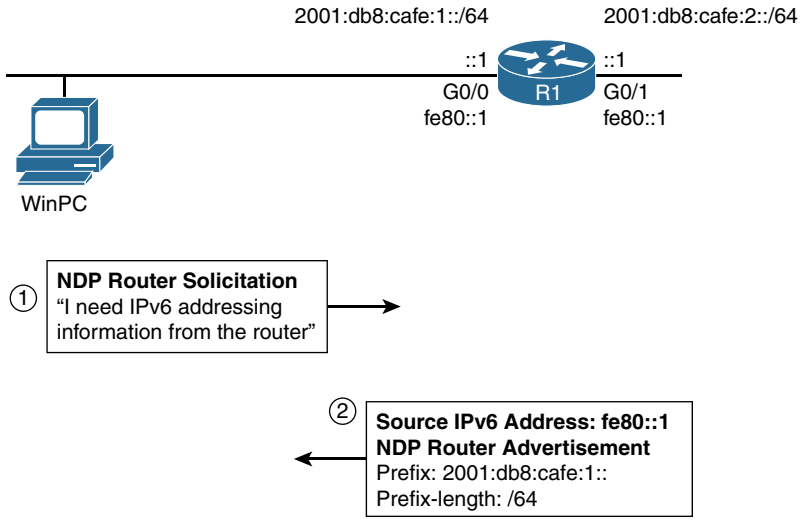
Although the address is still accepted, the interface detects that the address it wants to use is a duplicate and deems the address unusable. (Duplicate Address Detection is discussed in more detail in Chapter 13.)

Link-Local Addresses and Default Gateways

Using ICMPv6 Neighbor Discovery Protocol Router Solicitation and Router Advertisement messages, hosts determine how to obtain their IPv6 addressing information dynamically. If SLAAC is used, the host can automatically obtain IPv6 addressing information, such as prefix, prefix length, and a default gateway address. Stateless Address Autoconfiguration (SLAAC) is introduced in Chapter 2, and discussed in detail in Chapter 9.

Cisco routers send Router Advertisement messages every 200 seconds by default or after they receive a Router Solicitation message. The default gateway address is a link-local address, the IPv6 source address of the Router Advertisement message.

In Figure 6-9, WinPC has been configured to obtain its IPv6 address automatically. WinPC sends out an ND Router Solicitation (RS) message, and router R1 responds with a Router Advertisement (RA). WinPC uses a random number for its Interface ID and prepends the prefix from the RA message.



WinPC> **ipconfig**

Ethernet adapter Local Area Connection:

```
Connection-specific DNS Suffix  . :
IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086
Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
Default Gateway . . . . . . . . . : fe80::1%11
```

Figure 6-9 *WinPC* **ipconfig** *with Link-Local Address for Default Gateway*

Of interest here is the default gateway address. Router R1 sends the RA message using its link-local address fe80::1 as the source IPv6 address. WinPC uses this address as its default gateway address. WinPC also used the 2001:db8:cafe:1::/64 prefix to create its own GUA address, using SLAAC.

As mentioned previously, the default gateway address can only be obtained dynamically from the Router Advertisement message. DHCPv6 does not provide the default gateway address.

ipv6 enable: Isolated Link-Local Address

An IPv6 link-local address is created automatically whenever an IPv6 global unicast or unique local unicast is configured on the interface. A link-local address can also be created automatically without one of these unicast addresses, using the **ipv6 enable** interface command. The **no ipv6 enable** command is used to remove this option. The interface command for configuring **ipv6 enable** is as follows:

```
Router(config-if)# ipv6 enable
```

Example 6-10 demonstrates the use of this command to create a link-local address without configuring another unicast address. After this command is configured on an interface, the router immediately creates a link-local address using the EUI-64 format. This is verified using the **show ipv6 interface brief** command in Example 6-10. The GigabitEthernet 0/1 interface only has the EUI-64 generated link-local address, fe80::20c:30ff:fe10:92e1.

Example 6-10 ipv6 enable *Command*

```
Router(config)# interface gigabitethernet 0/1
Router(config-if)# ipv6 enable
Router(config-if)# end
Router# show ipv6 interface brief g 0/1
GigabitEthernet0/1            [up/up]
    FE80::20C:30FF:FE10:92E1
Router#
```

A global unicast or unique local address can still be configured on the interface when using the **ipv6 enable** command. However, when using the **ipv6 enable** command, if one of these unicast addresses is later removed, the interface will still continue to have the link-local address.

The **ipv6 enable** command isn't required to have just a link-local address on a router interface. Example 6-11 illustrates that a link-local address can be configured manually, without a global unicast address and without using the **ipv6 enable** command.

Example 6-11 *Configuring an Interface with Only a Link-Local Address*

```
Router(config)# interface gigabitethernet 0/0
Router(config-if)# ipv6 address fe80::99 link-local
Router(config-if)# end
Router# show ipv6 interface brief g 0/0
GigabitEthernet0/0            [up/up]
    FE80::99
Router#
```

To summarize, a Cisco IOS interface has an IPv6 link-local address in the following situations:

- When an IPv6 global unicast address or unique local unicast address is configured manually or dynamically on the interface
- When an IPv6 link-local address is configured manually on the interface
- When configuring an interface using the **ipv6 enable** command

Pinging a Link-Local Address

As you have seen, communicating with a link-local address is different than communicating with a global unicast address—for two reasons:

- The destination link-local address must be on the same link (subnet) as the source device.
- The same link-local address can exist on different networks. Link-local addresses only have to be unique on the link.

When pinging a link-local address, not only must the destination link-local address be on the same link, the device must know which link (interface) to use to send the packets. As illustrated in Figure 6-10, router A needs to send a ping to fe80::2, but it needs to know which output interface to use.

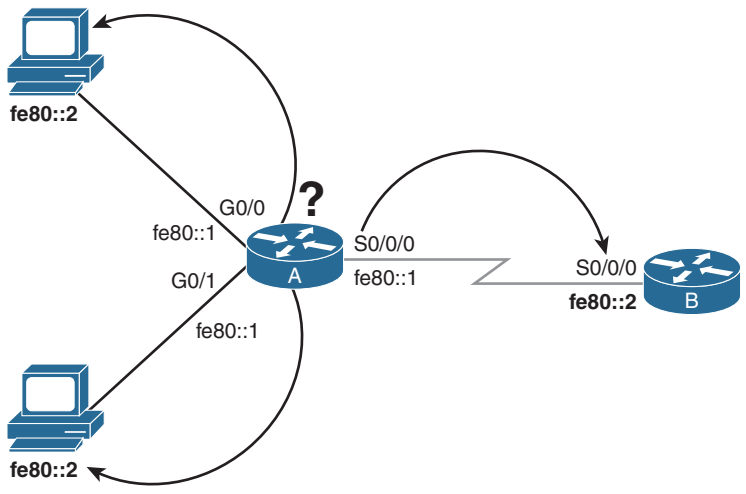


Figure 6-10 *Same Link-Local Address on Different Networks*

Using the topology in Figure 6-11, let's look at pinging a link-local address from a Cisco router, a Windows host (WinPC), a Linux host (LinuxPC), and a Mac OSX host (not shown in the topology).

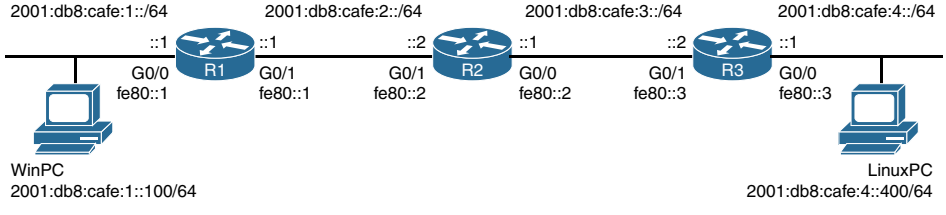


Figure 6-11 *Topology Used to Verify Link-Local Addresses*

Example 6-12 shows pinging a link-local address using Cisco IOS. Notice that you are prompted for the output interface so the router knows which link to use to send the ping. Also notice that Cisco IOS doesn't accept the shortened name for an interface, such as g0/1. Instead, you have to use the full interface name, without spaces.

Example 6-12 *Pinging a Link-Local Address Using Cisco IOS*

```
R2# ping fe80::1
Output Interface: g0/1
% Invalid interface. Use full interface name without spaces (e.g. Serial0/1)
Output Interface: gigabitethernet0/1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FE80::1, timeout is 2 seconds:
Packet sent with a source address of FE80::2%GigabitEthernet0/1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R2#
```

Example 6-13 shows an example of pinging the local router's link-local address from a Windows host (WinPC). The Zone ID can be used but is not required at the end of the link-local address, as Windows defaults to the wired interface. To find the Zone ID, use the **netsh interface ipv6 show interfaces** command, also shown in Example 6-13.

Example 6-13 *Pinging a Link-Local Address from Windows OS*

```
WinPC> ping fe80::1

Pinging fe80::1 with 32 bytes of data:
Reply from fe80::1: time=2ms
Reply from fe80::1: time=1ms
<output omitted for brevity>

WinPC> netsh interface ipv6 show interfaces

Idx     Met         MTU          State                Name
---  ----------  ----------  ------------  ---------------------------
  1          50  4294967295  connected     Loopback Pseudo-Interface 1
<output omitted for brevity>
 11          10        1500  connected     Local Area Connection
 13          50        1280  disconnected  Teredo Tunneling Pseudo-Interface

WinPC> ping fe80::1%11

Pinging fe80::1%11 with 32 bytes of data:
Reply from fe80::1%11: time=1ms
Reply from fe80::1%11: time=1ms
<output omitted for brevity>
```

Example 6-14 shows a Linux host (LinuxPC) pinging the local default gateway. Example 6-15 shows similar output for Mac OS (not shown in the topology in Figure 6-11). Both Linux and Mac OS use the **ping6** command when pinging an IPv6 address. Notice that the Zone ID (or Interface Scope) is required for Linux and Mac OS. The first ping (**ping6**) command failed because the Zone ID wasn't specified. The **ifconfig** command displays the interface configuration information, including the name of the interface required for the **ping6** command. Examples 6-14 and 6-15 show two options using the **ping6** command to ping a link-local address: using *%interface* at the end of the address or the **–I** *interface* parameter.

Example 6-14 *Pinging a Link-Local Address from Linux OS*

```
LinuxPC$ ping6 fe80::3
Connect: Invalid argument

LinuxPC$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:af:14:1b
          inet6 addr:0.0.0.6  Bcast:255.255.255.255  Mask:0.0.0.0
          inet6 addr: 2001:db8:cafe:4::400/64 Scope:Global
          inet6 addr: fe80::250:56ff:feaf:141b/64 Scope:Link
<output omitted>

LinuxPC$ ping6 fe80::3%eth0
PING fe80::3%eth0(fe80::3) 56 data bytes
64 bytes from fe80::3: icmp_seq=0 ttl=64 time=0.552 ms
64 bytes from fe80::3: icmp_seq=1 ttl=64 time=0.429 ms
<output omitted for brevity>

LinuxPC$ ping6 -I eth0 fe80::3
PING fe80::3%eth0(fe80::3) 56 data bytes
64 bytes from fe80::3: icmp_seq=0 ttl=64 time=0.552 ms
64 bytes from fe80::3: icmp_seq=1 ttl=64 time=0.551 ms
<output omitted for brevity>
```

Example 6-15 *Pinging a Link-Local Address from Mac OS*

```
MacOS$ ping6 fe80::1
ping6: sendmsg: No route to host
<output omitted for brevity>

MacOS$ ifconfig
en4: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    options=10b<RXCSUM,TXCSUM,VLAN_HWTAGGING,AV>
    ether a8:20:66:2c:9d:97
```

```
   inet6 fe80::aa20:66ff:fe2c:9d97%en4 prefixlen 64 scopeid 0x9
    inet6 2001:db8:cafe:1::200 prefixlen 64

MacOS$ ping6 fe80::1%en4
PING6(56=40+8+8 bytes) fe80::aa20:66ff:fe2c:9d97%en4 --> fe80::1%en4
16 bytes from fe80::1%en4, icmp_seq=0 hlim=64 time=5.205 ms 16 bytes from
  fe80::1%en4, icmp_seq=1 hlim=255 time=1.676 ms
<output omitted for brevity>

MacOS$ ping6 -I en4 fe80::1
PING6(56=40+8+8 bytes) fe80::aa20:66ff:fe2c:9d97%en4 --> fe80::1%en4
16 bytes from fe80::1%en4, icmp_seq=0 hlim=64 time=1.772 ms
16 bytes from fe80::1%en4, icmp_seq=1 hlim=255 time=1.086 ms
<output omitted for brevity>
```

Summary

This chapter focuses on the IPv6 link-local unicast address (LLA). A link-local address has the following characteristics:

- A device must have an IPv6 link-local address in order to be IPv6-enabled.
- A device doesn't have to have an IPv6 global unicast address, but it must have a link-local address.
- Link-local addresses are not routable off the link (subnet).
- Routers do not forward a packet with a source or destination link-local address.
- Link-local addresses have to be unique only on the link.
- There can be only one link-local address per interface.
- A link-local address can be a source or destination address.
- The range of link-local addresses is from fe80::/10 to febf::/10, but it is best to use fe80 for practical reasons.

Windows, Linux, and Mac OS hosts create a link-local unicast address at startup. Cisco IOS automatically generates a link-local address on an interface when it has a global unicast address or unique local unicast address, or when using the **ipv6 enable** command.

A link-local address is dynamically created. The prefix is typically fe80::/64, followed by a 64-bit Interface ID that is automatically generated either using EUI-64 or randomly.

The EUI-64 process converts the 48-bit Ethernet MAC address to a 64-bit Interface ID by doing the following:

- Inserting fffe in the middle of the MAC address, between the OUI and the Device Identifier.
- Flipping the seventh bit.

Link-local addresses can also be manually configured on Cisco IOS. Manual configuration makes it easier to recognize the link-local address and identify the appropriate device.

The Zone ID, also known as the Scope ID or Interface Scope, is used by a device to associate a link-local address with an interface. It may be required on some host operating systems when communicating using the link-local address. When pinging a link-local address with Cisco IOS, you are prompted for the output interface.

To ensure the uniqueness of a link-local address on the subnet, Duplicate Address Detection (DAD) is used. DAD is similar to a gratuitous ARP in IPv4. DAD is used for all unicast addresses, regardless of how they were configured or obtained.

Devices dynamically obtain their default gateway address from an ICMPv6 Router Advertisement message. The default gateway address is the link-local address of the local router.

You can use the **ipv6 enable** command to automatically create a link-local address on a router interface without configuring another unicast address.

Review Questions

1. True or false: An IPv6-enabled device must have either a link-local address or a global unicast address.
2. True or false: Routers do not forward packets with link-local addresses.
3. What is the range of the first hextet for a link-local address?
4. What are the two ways the Interface ID of the link-local address can be dynamically created?
5. How does the EUI-64 process convert a 48-bit Ethernet MAC address to a 64-bit Interface ID?
6. What method does Cisco IOS use for the Interface ID when automatically assigning a link-local address to an interface?
7. What is the purpose of the Zone ID, also known as the Scope ID or Interface Scope?
8. What is the advantage of manually configuring a link-local address on a router interface?
9. How does a device ensure that its link-local address is unique on the link (subnet)?
10. How does a device dynamically receive its default gateway address?

11. Does DHCPv6 provide a default gateway address?
12. What is the purpose of the **ipv6 enable** command?
13. What does Cisco IOS require when pinging a link-local address? Why?

References

RFCs

RFC 2373, *IP Version 6 Addressing Architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc2373.txt, July 1998.

RFC 4007, *IPv6 Scoped Address Architecture*, S. Deering, Cisco Systems, www.ietf.org/rfc/rfc4007.txt, March 2005.

RFC 4291, *IP Version 6 Addressing Architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc4291.txt, February 2006.

RFC 4429, *Optimistic Duplicate Address Detection (DAD) for IPv6*, N. Moore, Monash University CTIE, www.ietf.org/rfc/rfc4429.txt, April 2006.

RFC 4862, *IPv6 Stateless Address Autoconfiguration*, S. Thomson, Cisco Systems, www.ietf.org/rfc/rfc4862.txt, September 2007.

RFC 5342, *IANA Considerations for IETF Protocol Usage for IEEE 802 Parameters*, D. Eastlake 3rd, Eastlake Enterprises, www.ietf.org/rfc/rfc5342.txt, September 2008.

RFC 5375, *IPv6 Unicast Address Assignment Considerations*, G. Van de Velde, www.ietf.org/rfc/rfc5375.txt, December 2008.

This page intentionally left blank

Chapter 7

Multicast Addresses

This chapter discusses IPv6 multicast addresses (see Figure 7-1). Multicast is a technique in which a device sends a single packet to multiple destinations simultaneously (one-to-many), in contrast to a unicast address, which sends a single packet to a single destination (one-to-one). Multiple destinations can actually be multiple interfaces on the same device, but they are typically different devices. Multicast is not new to IPv6; it has been available in IPv4 since 1988.

A multicast address is used to send a single packet or, more likely, a single stream of packets to multiple devices simultaneously. This is much more efficient than duplicating a stream of packets as a separate unicast transmission for each destination.

Multicast can be a better option than broadcast when the recipients are only a subset of all the devices on a subnet. IP multicast packets have the potential of being filtered by the Ethernet switch and the NIC. Filtering multicast frames at the switch, which is similar to filtering unicast frames, can be accomplished by implementing either of the following:

- IGMP (Internet Group Management Protocol) IPv4 multicasting
- MLD (Multicast Listener Discovery) snooping for IPv6 multicasting

An IPv6 multicast address has the prefix ff00::/8, which defines a group of devices known as a *multicast group*. It is the IPv4 equivalent of 224.0.0.0/4. A packet sent to a multicast group always has a unicast source address. A multicast address can only be a destination address and can never be a source address.

IPv6 multicast addresses are introduced in Chapter 4, "IPv6 Address Representation and Address Types." This chapter begins with a review of the concepts introduced there. Table 7-1 shows the various representations of an IPv6 multicast address.

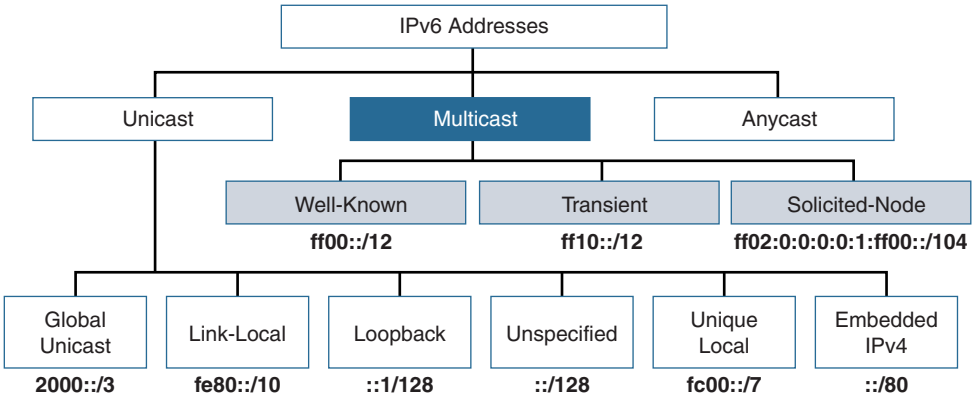


Figure 7-1 *Multicast Addresses*

Table 7-1 *IPv6 Multicast Address Representations*

Representation	IPv6 Multicast Address
Preferred	ff00:0000:0000:0000:0000:0000:0000:0000/8
No leading 0s	ff00:0:0:0:0:0:0:0/8
Compressed	ff00::/8

Figure 7-2 shows the structure of an IPv6 multicast address:

- **ff00::/8 –** The first 8 bits are 1-bits (ff), reserved for IPv6 multicast.
- **Flags –** The next 4 bits are allocated for flags. The first three flags are: 0 (reserved), R (rendezvous point), and P (network prefix), which are beyond the scope of this book. The fourth flag is the transient flag (T flag), which denotes two types of multicast addresses:
  - **Permanent (0):** These addresses, known as *predefined multicast addresses*, are assigned by the Internet Assigned Numbers Authority (IANA) and include both well-known and solicited-node multicast.
  - **Nonpermanent (1):** These are "transient," or "dynamically assigned," multicast addresses. They are assigned by multicast applications.
- **Scope –** The Scope field defines the range to which routers can forward the multicast packet.
- **Group ID –** The next 112 bits represent the Group ID.

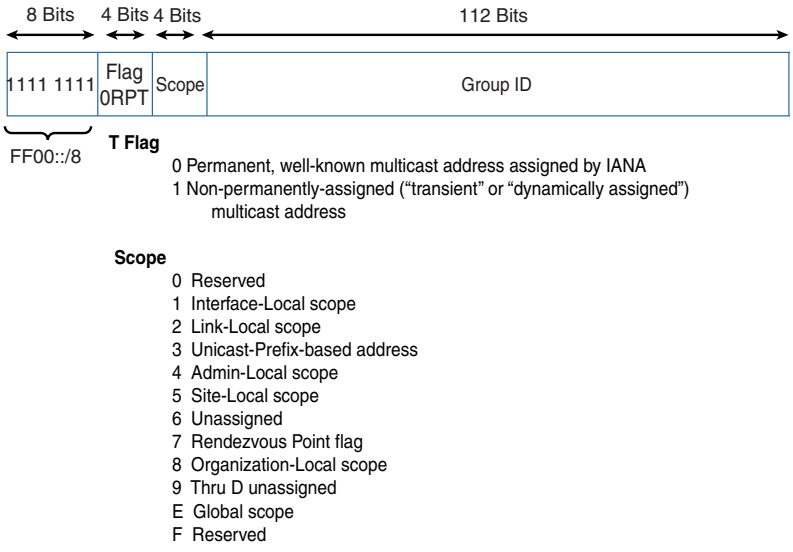


Figure 7-2 *IPv6 Multicast Address*

This chapter focuses on the two types of multicast addresses defined by the fourth flag: well-known and solicited-node multicast addresses. Well-known addresses are similar to IPv4 well-known multicast addresses. These are reserved addresses for the use of routing protocols and other topology discovery or maintenance protocols. Solicited-node multicast addresses are new to IPv6. This address type is used as a more efficient approach to IPv4's broadcast address.

Another type of multicast address is transient, or dynamically assigned, multicast addresses. These addresses are assigned by multicast applications, and they are used for various software applications and services, such as video delivery, remote imaging, and backups.

Scope

Let's continue our discussion of multicast by talking a little more about the Scope field. Scope is a 4-bit field used to define the range of the multicast packet, shown in Figure 7-2. The Scope field can have the following possible values:

- **0:** Reserved
- **1:** Interface-Local scope
- **2:** Link-Local scope
- **3:** Unicast-Prefix-based address
- **4:** Admin-Local scope
- **5:** Site-Local scope
- **6 :** Unassigned

- **7 :** Rendezvous Point flag
- **8 :** Organization-Local scope
- **9 :** Unassigned
- **A:** Unassigned
- **B :** Unassigned
- **C:** Unassigned
- **D:** Unassigned
- **E :** Global scope
- **F :** Reserved

RFC 4007, *IPv6 Scoped Address Architecture*, specifies the characteristics, expected behavior, and usage of IPv6 addresses of different scopes. Figure 7-3 provides a more graphical representation of these scopes. The Scope field allows devices to define the range of the multicast packet and allows routers to immediately determine how broadly to propagate the packet. This improves efficiency by preventing traffic from being sent outside the intended area.

Note There is no automatic mechanism for routers to filter multicast packets. The boundaries for site/organization local addresses must be manually configured. The exception is for link-local multicast, which is filtered automatically.

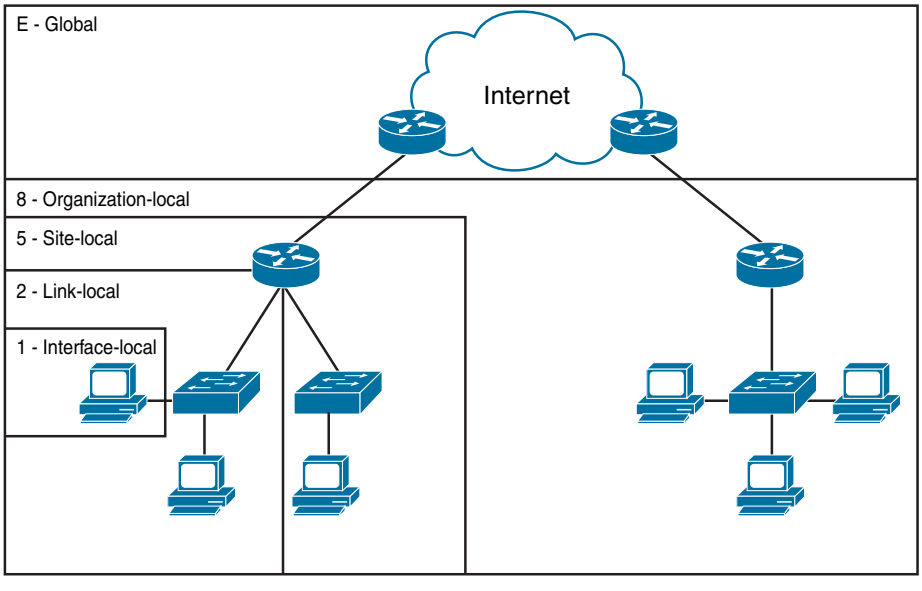


Figure 7-3 *Multicast Scope*

The Scope field is the fourth hexadecimal digit in the first hextet, for example:

- **ff02::2 –** All-IPv6 routers
- **ff05::2 –** All-IPv6 routers

Both of these multicast addresses are for the all-IPv6 routers multicast group. The difference is in the Scope field. ff0**2**::2 has a Scope field of 2, which is link-local scope. ff0**5**::2 has a Scope field of 5, which is site-local scope.

Multicast with Link-Local Scope Versus Link-Local Unicast Addresses

One of the most common multicast Scope values is link-local scope—the 2 in ff02. A multicast address with link-local scope (ff02) should not be confused with a link-local unicast address (fe80::/10).

A multicast address with link-local scope has these characteristics:

- It is a multicast address, not a unicast address.
- It can only be a destination address.
- It can be sent to a group of devices only on the same link and is not routable off the link.
- It is typically a solicited-node multicast or a well-known multicast used for neighbor discovery and routing protocol messages.

A packet with a link-local unicast address has these characteristics:

- It is a unicast address, not a multicast address.
- It can be a source or destination address.
- A link-local unicast address (source or destination address) is confined to the link and is not routable off the link.
- A link-local unicast address can be used as a destination address when needing to communicate with a single device on the link, such as a printer.
- A link-local unicast address can be a source address when communicating with devices on the same link.

A link-local unicast address may be the source address when the destination address is a multicast address with link-local scope. For example, Figure 7-4 shows an ICMPv6 Neighbor Solicitation followed by a Neighbor Advertisement message. Both of these messages use the device's link-local unicast address (LLA) as the source address and a well-known multicast address with link-local scope for the destination address. The source address is a link-local unicast address, coming from a single device.

The destination address is a multicast address with link-local scope—sent to a multicast group only on that link. The Router Solicitation message is for the all-IPv6 routers multicast group, and the Router Advertisement message is for the all-IPv6 devices multicast group.

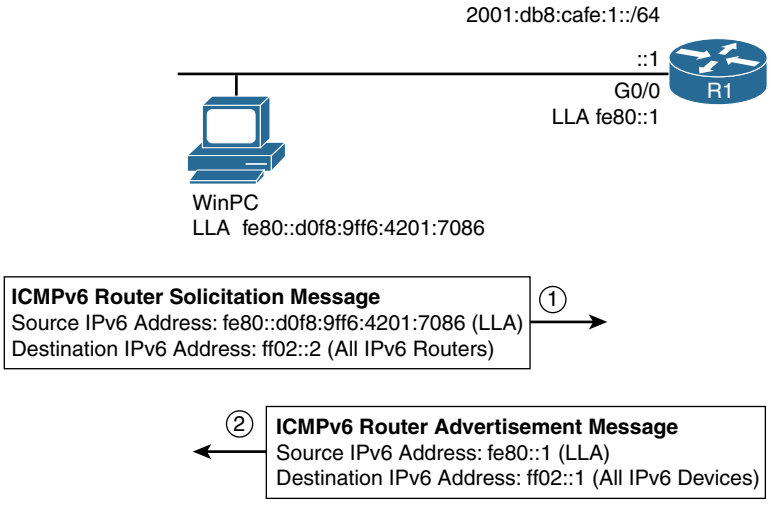


Figure 7-4 *Link-Local Unicast Addresses and Multicast Addresses with Link-Local Scope*

Well-Known Multicast Addresses

Well-known multicast addresses are predefined or reserved multicast addresses for assigned multicast groups. Well-known multicast addresses have the prefix ff00::/12. The T flag, the fourth flag in the Flag field, is set to 0. These addresses are equivalent to IPv4 well-known multicast addresses in the range 224.0.0.0 to 239.255.255.255. This address type is typically used for neighbor discovery and routing protocol messages.

RFC 2375, *IPv6 Multicast Address Assignments*, defines the initial assignment of IPv6 multicast addresses that have permanently assigned Global IDs. IANA maintains the list of well-known IPv6 multicast addresses; see IANA's IPv6 Multicast Address Space Registry, www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml.

Table 7-2 shows the format and some examples of well-known or assigned multicast addresses.

Table 7-2 *Well-Known Multicast Addresses*

/8 Prefix	Flag	Scope	Predefined Group ID	Compressed Format	Description
Interface-Local Scope					
ff	0	1	0:0:0:0:0:0:1	ff01::1	All-nodes
ff	0	1	0:0:0:0:0:0:2	ff01::2	All-routers
Link-Local Scope					
ff	0	2	0:0:0:0:0:0:1	ff02::1	All-nodes
ff	0	2	0:0:0:0:0:0:2	ff02::2	All-routers
ff	0	2	0:0:0:0:0:0:5	ff02::5	OSPF routers
ff	0	2	0:0:0:0:0:0:6	ff02::6	OSPF designated routers
ff	0	2	0:0:0:0:0:0:9	ff02::9	RIP routers
ff	0	2	0:0:0:0:0:0:a	ff02::a	EIGRP routers
ff	0	2	0:0:0:0:0:1:2	ff02::1:2	All-DHCP agents
Site-Local Scope					
ff	0	5	0:0:0:0:0:0:2	ff05::2	All-routers
ff	0	5	0:0:0:0:0:1:3	ff05::1:3	All-DHCP servers

Note IPv6 does not have a broadcast address, but there is an all-nodes or all-IPv6 devices multicast address, ff02::1, which has a similar effect.

In Table 7-2, notice that the same predefined Group ID can have various scopes or ranges. Depending on the scope, a packet sent to the all-routers Group ID 0:0:0:0:0:0:2 (::2) can be confined to a single link (ff0**2**::2) or can be for an entire site (ff0**5**::2).

Note When implementing site-local scope or any scope requiring multicast packets to be routed, IPv6 multicast routing must be enabled using the **ipv6 multicast-routing** global configuration command.

You might notice a consistency in the Group ID between IPv4 and IPv6 multicast addresses. For example, all-OSPF routers is 224.**0.0.5** in IPv4 multicast and ff02**::5** in IPv6 multicast. Both protocols use 5 as the Group ID. Well-known multicast addresses are

used with specific protocols such as Neighbor Discovery Protocol and OSPFv3, which are examined in more detail in the later chapters.

Figure 7-5 shows the topology this section uses to demonstrate multicast addresses. Chapter 6, "Link-Local Unicast Address," shows how to manually configure link-local addresses on each of the routers. To better illustrate multicast addresses for this chapter, instead of manually configured addresses the routers in the topology are using their default link-local addresses generated using EUI-64.

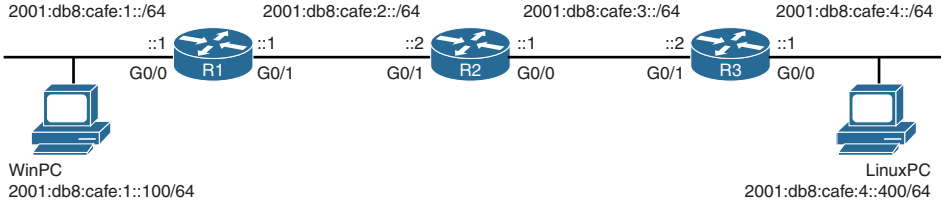


Figure 7-5 *Topology for Multicast Addresses Example*

The **show ipv6 interface gigabitethernet 0/0** command in Example 7-1 shows the multicast groups for router R1's g0/0 interface. The output shows that this interface is a member of three well-known multicast groups and has two solicited-node multicast addresses. Router R1 listens for and processes any packet that has a destination address with one of these multicast addresses:

Well-known multicast groups:

- **ff02::1 –** All-nodes multicast group for this link. The interface becomes a member of this multicast group when it is enabled for IPv6 by having a link-local unicast address.
- **ff02::2 –** All-routers multicast group for this link. R1 has been configured with the **ipv6 unicast-routing** command to enable routing of IPv6. The **show running-config** command in Example 7-1 shows the **ipv6 unicast-routing** command and the static route that was configured previously.
- **ff02::fb –** Multicast DNS for this link.

Solicited-node multicast addresses (explained later in this chapter):

- **ff02::1:ff00:1 –** This is the solicited-node multicast address for R1's global unicast address on this interface.
- **ff02::1:ff93:da00 –** This is the solicited-node multicast address for R1's link-local address on this interface.

Example 7-1 *Displaying Multicast Groups on Router R1's G0/0 Interface*

```
R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::5AAC:78FF:FE93:DA00
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
  Joined group address(es):
    FF02::1             ! All-IPv6 devices
    FF02::2             ! All-IPv6 routers
    FF02::FB            ! Multicast DNS
    FF02::1:FF00:1      ! Solicited-node multicast for GUA
    FF02::1:FF93:DA00   ! Solicited-node multicast for LLA
<output omitted for brevity>

R1# show running-config
<partial output>
ipv6 unicast-routing
ipv6 route ::/0 2001:DB8:CAFE:2::2
```

Example 7-2 shows the multicast groups for WinPC and LinuxPC. The **netsh interface ipv6 show joins** command shows the multicast groups for WinPC, and the **netstat -g** command shows the same information for LinuxPC. Notice that both hosts are members of ff01::1 and ff02::2, the all-IPv6 devices multicast group for local scope and link-local scope, respectively. (Linux resolves the ff02::2 address to ip6-allnodes.) WinPC and LinuxPC listen for and process any packets with this destination IPv6 address.

Example 7-2 *Displaying Multicast Groups on WinPC and LinuxPC*

```
Windows PC

WinPC> netsh interface ipv6 show joins

Interface 11: Local Area Connection

Scope  References  Last  Address
-----  ----------  ----  ---------------------------------
! All-IPv6 devices, local scope
0               0  Yes   ff01::1
! All-IPv6 devices, link-local scope
0               0  Yes   ff02::1
! Multicast Name Resolution
0               1  Yes   ff02::1:3
```

```
! Solicited-node GUA
0               1  Yes   ff02::1:ff00:100
! Solicited-node LLA
0               2  Yes   ff02::1:ff01:7086

<output omitted for brevity>
------------------------------------------------------------------
Ubuntu Linux PC

LinuxPC$ netstat -g
IPv6/IPv4 Group Memberships
Interface  RefCnt Group
---------- ------ ---------------------
! Solicited-node multicast GUA
eth0       1      ff02::1:ff00:400
! Solicited-node multicast LLA
eth0       1      ff02::1:ffaf:141b
! Multicast Name Resolution
eth0       1      ff02::fb
! All-IPv6 devices, link–local scope
eth0       1      ip6-allnodes
! All-IPv6 devices, local scope
eth0       1      ff01::1
<some output omitted for brevity>
```

Note The **netstat -g** command is also used for Mac OS to display IP multicast groups.

Note ff02::fb is the mDNS (multicast DNS) address for IPv6. mDNS is a zero-configuration service that resolves host names to IP addresses. mDNS is typically used in small networks that do not include a local DNS server. ff02::1:3 is the multicast address for all-LLMNR (Link-Local Multicast Name Resolution) hosts on a local network segment. LLMNR uses the DNS packet format to perform IP name resolution on the local link. mDNS and LLMNR are beyond the scope of this book. mDNS is described in RFC 6762, *Multicast DNS*. LLMNR is described in RFC 4795, *Link-Local Multicast Name Resolution*.

Solicited-Node Multicast Addresses

Another multicast address is the solicited-node multicast address, shown in Figure 7-1. The solicited-node multicast can be confusing to some, but this section explains how it is created and discusses the benefits of using this new type of multicast address.

A solicited-node multicast address is automatically created and assigned to an interface for every global unicast address, unique local address, and link-local address on that interface. These multicast addresses are automatically generated using a special mapping of the device's unicast address with the solicited-node multicast prefix ff02:0:0:0:0:1:ff00::/104, as shown in Table 7-3. (This prefix looks ugly, but at least it makes it easy to recognize a solicited-node multicast address.) The solicited-node multicast prefix ff02:0:0:0:0:1:ff00::/104 is prepended to the low-order 24 bits of the unicast address. You will see how this is done in a moment.

Table 7-3 *IPv6 Solicited-Node Multicast Address Representations*

Representation	IPv6 Loopback Address
Preferred	ff02:0000:0000:0000:0000:0001:ff00::/104
Compressed	ff02:0:0:0:0:1:ff00::/104

One of the benefits of using a Layer 3 multicast address is that it is mapped to a Layer 2 Ethernet MAC address. This allows the frame to be filtered by the switch. This means these packets will only be forwarded out ports where there are devices that are members of that multicast group. This is done using Multicast Listener Discovery (MLD) and is discussed later in this chapter.

Note IPv6 solicited-node multicasts are not filtered by MLD. These packets are forwarded out all ports because of the potentially huge forwarding tables needed to store these addresses. Solicited-node multicast and MLD are discussed later in this chapter.

If MLD is not implemented on the switch, the frame containing the multicast packet will be flooded out all ports (which is the default on most switches). Using the destination MAC address, the frame can still be filtered at the Ethernet NIC. The Ethernet NIC can determine whether to discard the frame or pass it up to an upper-layer protocol such as IPv6 for further processing. If the device is not a member of that multicast group, the Ethernet NIC drops the frame.

Note The mapping of IPv6 multicast addresses to Ethernet MAC addresses is discussed later in this chapter.

IPv6's solicited-node multicast address provides a more efficient solution for processes such as address resolution (ARP). The mapping of a solicited-node multicast address to an Ethernet MAC address allows the NIC to determine whether it is the intended recipient, without sending to an upper-layer protocol for processing.

Solicited-node multicast addresses are used for two essential IPv6 mechanisms, both part of Neighbor Discovery Protocol (NDP):

- **Address resolution:** Performing much the same function as an ARP Request in IPv4, an IPv6 device sends a Neighbor Solicitation message to a solicited-node multicast address to learn the link layer (typically Ethernet) address of a device on the same link. The device knows the IPv6 address of the destination on that link but needs to know its data-link (Ethernet) address. Address resolution is illustrated in Figure 7-6. The addresses used in this example are discussed shortly.
- **Duplicate Address Detection (DAD):** DAD allows a device to verify that its unicast address is unique on the link. A Neighbor Solicitation message is sent to the device's own solicited-node multicast address to determine whether anyone else has this same address.

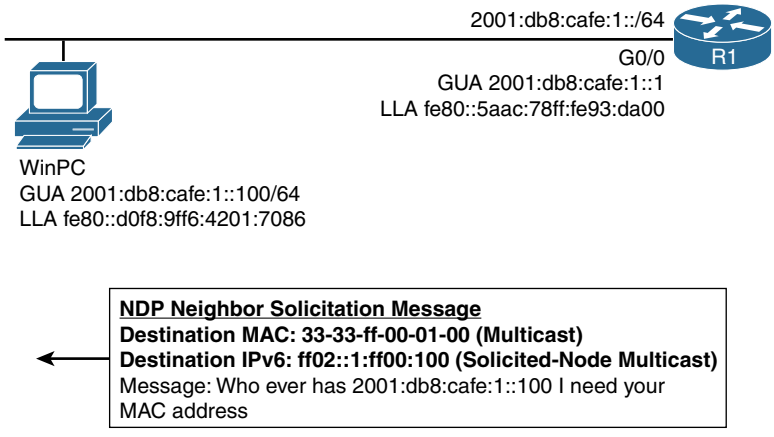


Figure 7-6 *Use of Solicited-Node Multicasts with Address Resolution*

Note Neighbor Discovery Protocol is discussed in more detail in Chapter 13, "ICMPv6 Neighbor Discovery."

Mapping Unicast Address to Solicited-Node Multicast Address

As mentioned previously, a solicited-node multicast address is automatically created for every unicast address (global uncast, unique local, and link-local) on the interface. The solicited-node multicast prefix ff02:0:0:0:0:1:ff00::/104 is prepended to the low-order 24 bits of the unicast address. Let's see how this is done.

In Figure 7-6, Router R1's GigabitEthernet 0/0 interface has two addresses: a global unicast address and a link-local unicast address. Using the **show ipv6 interface gigabitethernet 0/0** command in Example 7-3, you can see both of these unicast addresses, along with their associated solicited-node multicast addresses.

Example 7-3 *Displaying Solicited-Node Multicasts on Router R1's G0/0 Interface*

```
R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::5AAC:78FF:FE93:DA00
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
  Joined group address(es):
    FF02::1             ! All-IPv6 devices
    FF02::2             ! All-IPv6 routers
    FF02::FB            ! Multicast DNS
    FF02::1:FF00:1      ! Solicited-node multicast for GUA
    FF02::1:FF93:DA00   ! Solicited-node multicast for LLA
<output omitted for brevity>
```

In the output in Example 7-3, notice that Router R1 has two solicited-node multicast addresses: one for its global unicast address and one for its link-local unicast address. The unicast address–to–solicited-node multicast address associations are as follows:

- Global unicast address 2001:db8:cafe:1::1 to solicited-node multicast address ff02::1:ff00:1
- Link-local unicast address fe80::5aac:78ff:fe93:da00 to solicited-node multicast address ff02::1:ff93:da00

Figure 7-7 shows the mapping for router R1's global unicast address, and Figure 7-8 shows the mapping for R1's link-local address.

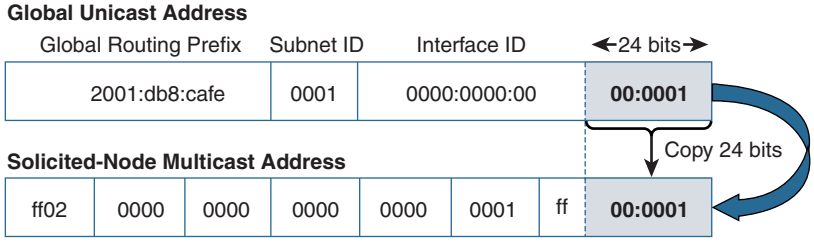


Figure 7-7 *Mapping R1's Global Unicast Address to a Solicited-Node Multicast Address*

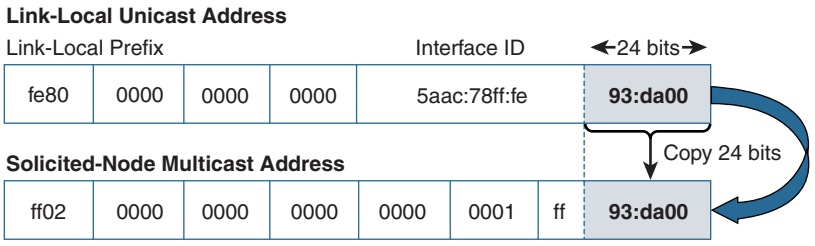


Figure 7-8 *Mapping R1's Link-Local Unicast Address to a Solicited-Node Multicast Address*

Notice that both of these solicited-node multicast addresses were created from their respective unicast addresses, with the low-order 24 bits in common. Each solicited-node address uses the ff02:0:0:0:0:1:ff00::/104 prefix plus the 24 bits of the unicast address.

Mapping to the Ethernet MAC Address

IPv6 multicast addresses, including solicited-node and well-known multicast addresses, are mapped to Ethernet MAC addresses. This is where you see the real advantage of using a multicast address over a broadcast address.

An Ethernet broadcast results in every NIC on the subnet receiving the Ethernet frame and passing it to an upper-layer protocol for further processing. As discussed previously, in the case with an ARP Request, every device's NIC on the subnet must receive the frame and pass it to the ARP process to determine whether it is the target. Unlike an Ethernet broadcast, an Ethernet multicast can be filtered by the NIC. You will see an example of this shortly.

33-33-xx-xx-xx-xx is the reserved Ethernet MAC address when carrying an IPv6 multicast packet, as described in RFC 7042, *IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters*. The lower 32 bits of the MAC address (xx-xx-xx-xx) are copied from the lower 32 bits of the IPv6 multicast address.

Note Why 33-33? This is the address of Xerox PARC (Palo Alto Research Center): 3333 Coyote Hill Road, Palo Alto, California. Ethernet was originally developed by Digital Equipment Corporation, Intel Corporation, and Xerox PARC.

Mapping Solicited-Node Multicast to Ethernet MAC Addresses

Figures 7-9 and 7-10 show the continuation of the mappings from a unicast address to a solicited-node multicast to an Ethernet multicast MAC address.

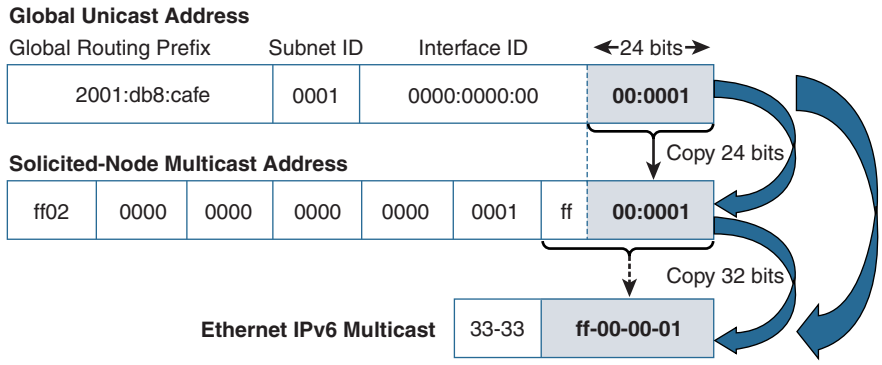


Figure 7-9 *Mapping R1's Global Unicast Address to a Solicited-Node Multicast Address to an Ethernet MAC Address*

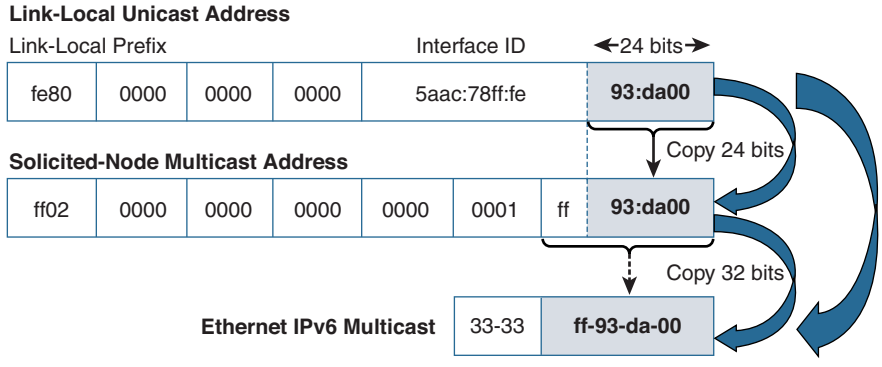


Figure 7-10 *Mapping R1's Link-Local Unicast Address to a Solicited-Node Multicast Address to an Ethernet MAC Address*

Using the topology in Figure 7-5, Table 7-4 shows the global unicast and link-local unicast addresses for router R1, WinPC, and LinuxPC. For each unicast address, the associated solicited-node multicast address and Ethernet multicast MAC address are shown. The solicited-node multicast addresses were created using the low-order 24 bits of the respective unicast address. Then, the Ethernet MAC address was created using the low-order 32 bits of the solicited-node multicast address. Bold formatting is used for the addresses to help display the 24 bits common in all three types of addresses: unicast, solicited-node multicast, and multicast MAC addresses.

Because a solicited-node multicast address is created using only the low-order 24 bits of its respective unicast address, and not the entire 64-bit Interface ID (in a /64 network), there is the potential for multiple devices to share the same solicited-node multicast address. This doesn't cause any problems. The upper-layer protocol, such as an ICMPv6 Neighbor Solicitation message, contains the target IPv6 address. The intended target, the device whose IPv6 address matches the target address, continues to process the packet, and the other devices end processing.

Table 7-4 *Unicast to Solicited-Node Multicast to Ethernet MAC Address Mappings*

Unicast Address		Solicited-Node Multicast Address	Ethernet Multicast MAC Address
Router R1			
Global	2001:db8:cafe:1::**1**	ff02::1:ff00:**1**	33-33-ff-**00-00-01**
Link-local	fe80::5aac:78ff:fe**93:da00**	ff02::1:ff**93:da00**	33-33-ff-**93-da-00**
WinPC			
Global	2001:db8:cafe:1::**100**	ff02::1:ff**00:100**	33-33-ff-**00-01-00**
Link-local	fe80::d0f8:9ff6:42**01:7086**	ff02::1:ff**01:7086**	33-33-ff-**01-70-86**
LinuxPC			
Global	2001:db8:cafe:4::**400**	ff02::1:ff00:**400**	33-33-ff-**00-04-00**
Link-local	fe80::250:56ff:fe**af:141b**	ff02::1:ff**af:141b**	33-33-ff-**af-14-1b**

What this means is that these devices not only process packets with a destination address of their unicast address but also packets that have their respective solicited-node multicast addresses. This also means that the Ethernet NIC will accept any Ethernet frames with the multicast MAC address mapped to a solicited-node multicast address.

Figure 7-11 shows WinPC's Layer 2 and Layer 3 addresses. The Layer 2 addresses are the Ethernet MAC addresses that WinPC's NIC accepts, along with the MAC address of the NIC. The IPv6 packets are then handed to the IPv6 process and continue to be processed.

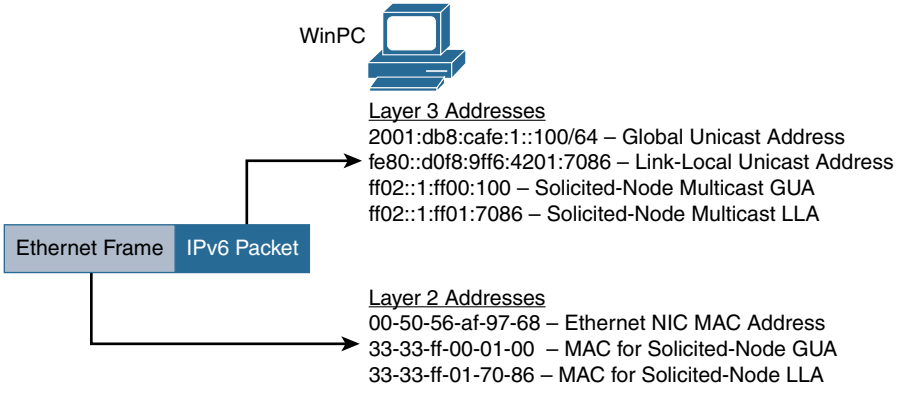


Figure 7-11 *WinPC Layer 3 and Layer 2 Addresses*

Example 7-4 shows an example of this mapping in a Neighbor Solicitation message, equivalent to an ARP Request in IPv4. This is the same Neighbor Solicitation message illustrated in Figure 7-6. Router R1 is the sender of the NS message. R1 knows the IPv6 GUA address of WinPC but not its Ethernet MAC address. Notice the mappings of the addresses:

- **ICMPv6 Neighbor Solicitation message:**
  - **Target Address 2001:db8:cafe:1::100 –** R1 needs the MAC address for this IPv6 unicast address. The low-order 24 bits of this address are mapped to a solicited-node multicast address and used as the destination address in the IPv6 header.
- **IPv6 header:**
  - **Destination Address ff02::1:ff00:100 –** This is the solicited-node multicast address mapped from the Neighbor Solicitation target address. The low-order 32 bits of this address are mapped to an Ethernet multicast MAC address.
- **Ethernet header:**
  - **Destination MAC Address 33:33:ff:00:01:00 –** This is the multicast MAC address mapped from the solicited-node multicast address. This address can be filtered by the Ethernet NIC.

Example 7-4 *Wireshark Capture of ICMPv6 Neighbor Solicitation Message from R1*

```
Ethernet II, Src: 58:ac:78:93:da:00, Dst: 33:33:ff:00:01:00
Internet Protocol Version 6
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 32
    Next header: ICMPv6 (0x3a)
    Hop limit: 255
    Source: 2001:db8:cafe:1::1
    Destination: ff02::1:ff00:100
Internet Control Message Protocol v6
    Type: 135 (Neighbor solicitation)
    Code: 0
    Target: 2001:db8:cafe:1::100
    ICMPv6 Option (Source link-layer address)
        Type: Source link-layer address (1)
        Length: 8
        Link-layer address: 58:ac:78:93:da:00
```

Figure 7-12 illustrates the mapping in Example 7-4.

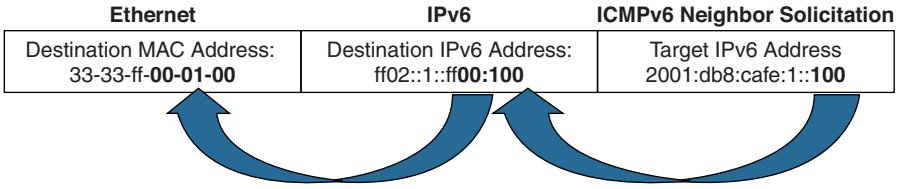


Figure 7-12 *Address Mappings for R1's Neighbor Solicitation Message*

The Ethernet frame carrying the IPv6 packet and the Neighbor Solicitation message has a mapped multicast address for the destination MAC address. In Example 7-4, every device's NIC on the subnet receives this frame and determines whether it is the intended target. Only those NICs, including WinPC, that have this mapped multicast MAC address will accept the Ethernet frame. The NICs for all the other devices drop the frame. Therefore, the decision can be made at the NIC without sending it to another process to make this determination.

Mapping Well-Known Multicast to Ethernet MAC Addresses

Well-known multicast addresses are also mapped to Ethernet MAC addresses. Table 7-5 shows several examples. Once again, a key benefit of using this mapping over using a broadcast address is that it allows the Ethernet NIC to examine the destination multicast MAC address and determine whether to pass the encapsulated data to an upper-layer protocol for processing.

Table 7-5 *Well-Known Multicast to Ethernet MAC Address Mappings*

Description	Well-Known Multicast	Mapped Ethernet MAC Address
All-Devices	ff02::**1**	33-33-ff-**00-00-01**
All-Routers	ff02::**2**	33-33-ff-**00-00-02**
All-OSPF Routers	ff02::**5**	33-33-ff-**00-00-05**
All-EIGRP Routers	ff02::**a**	33-33-ff-**00-00-0a**

Verifying the Address Mappings on Cisco IOS, Windows, and Linux

Using the information in Table 7-4, you can verify the solicited-node multicast addresses for router R1, WinPC, and LinuxPC.

The **show ipv6 interface** command shown in Example 7-5 verifies the solicited-node multicast addresses on an interface. Again, notice that the prefix ff02:0:0:0:0:1:ff00::/104 is prepended to the low-order 24 bits of the unicast address to form the solicited-node multicast address.

Example 7-5 *Verifying the Solicited-Node Multicasts on Router R1's G0/0 Interface*

```
R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::5AAC:78FF:FE93:DA00
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::FB
   FF02::1:FF00:1      ! Solicited-node multicast for GUA
   FF02::1:FF93:DA00   ! Solicited-node multicast for LLA
<output omitted for brevity>
```

Example 7-6 demonstrates the verification of the solicited-node multicast addresses on WinPC. The **ipconfig** command is used to display the global unicast and link-local unicast addresses. The **netsh interface ipv6 show joins** command displays the solicited-node multicast addresses for both unicast addresses.

Example 7-6 *Verifying the Solicited-Node Multicasts on WinPC*

```
WinPC> ipconfig

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1::100
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
   Default Gateway . . . . . . . . . : 2001:db8:cafe:1::1
<output omitted for brevity>

WinPC> netsh interface ipv6 show joins

Interface 11: Local Area Connection

Scope  References  Last  Address
-----  ----------  ----  ---------------------------------
0               0  Yes   ff01::1
0               0  Yes   ff02::1
0               1  Yes   ff02::1:3
! Solicited-node GUA
0               1  Yes   ff02::1:ff00:100
! Solicited-node LLA
0               2  Yes   ff02::1:ff01:7086

<output omitted for brevity>
```

Example 7-7 shows the commands for the solicited-node multicast address on LinuxPC. The **ifconfig** command displays the global unicast and link-local uncast addresses. The **netstat -g** command then verifies the solicited-node multicast address for both of these unicast addresses.

Example 7-7 *Verifying the Solicited-Node Multicasts on LinuxPC*

```
LinuxPC$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:af:14:1b
          inet6 addr:0.0.0.6  Bcast:255.255.255.255  Mask:0.0.0.0
          inet6 addr: 2001:db8:cafe:4::400/64 Scope:Global
          inet6 addr: fe80::250:56ff:feaf:141b/64 Scope:Link
<output omitted>

LinuxPC$ netstat -g
IPv6/IPv4 Group Memberships
Interface  RefCnt Group
---------- ------ ---------------------
! Solicted-node multicast GUA
eth0       1      ff02::1:ff00:400
! Solicted-node multicast LLA
eth0       1      ff02::1:ffaf:141b
! Multicast Name Resolution
eth0       1      ff02::fb
! All-IPv6 devices, link—local scope
eth0       1      ip6-allnodes
! All-IPv6 devices, local scope
eth0       1      ff01::1
<some output omitted for brevity>
```

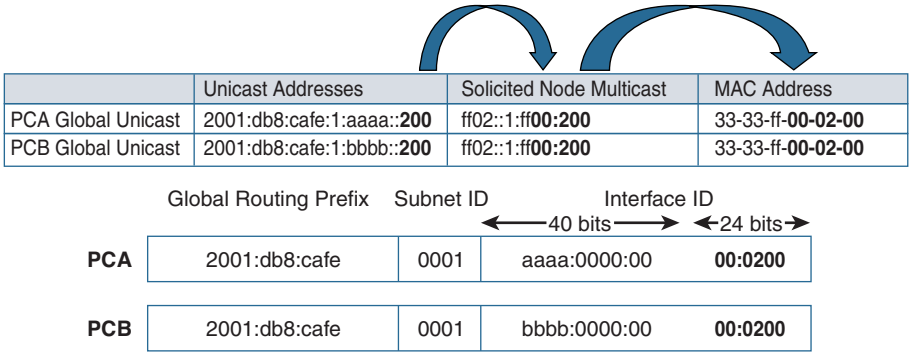
Multiple Devices Using the Same Solicited-Node Multicast Address

Can two devices on the same subnet have the same solicited-node multicast address? Yes. Is that a problem? No. Duplicate Address Detection (DAD) is not applicable because DAD is only performed on all unicast addresses prior to assigning them to an interface.

As mentioned earlier, it is possible that more than one device can have the same solicited-node multicast address on a subnet. This occurs only when the devices have the same low-order 24 bits in their global unicast or link-local unicast addresses.

Figure 7-13 shows an example of two hosts with different global unicast addresses but the same solicited-node multicast address. PCA and PCB have different GUA Interface IDs:

- **PCA's Interface ID:** *aaaa*:0000:0000:0200
- **PCB's Interface ID:** *bbbb*:0000:0000:0200



	Unicast Addresses	Solicited Node Multicast	MAC Address
PCA Global Unicast	2001:db8:cafe:1:aaaa::**200**	ff02::1:ff**00:200**	33-33-ff-**00-02-00**
PCB Global Unicast	2001:db8:cafe:1:bbbb::**200**	ff02::1:ff**00:200**	33-33-ff-**00-02-00**

Figure 7-13 *PCA and PCB with Different GUA Addresses but the Same Solicited-Node Multicast Address*

PCA and PCB have different GUA addresses because the first hextet of their Interface IDs differ. Any difference in the first 40 bits of the Interface ID does not affect the solicited-node multicast address because only the last 24 bits are copied to the solicited-node multicast address.

So, what happens when two (or more) devices share the same solicited-node multicast address? Figure 7-14 shows how an ICMPv6 Neighbor Solicitation (NS) message would be processed by two hosts that share the same solicited-node multicast address. Remember that a Neighbor Solicitation message is similar to an ARP Request in IPv4.

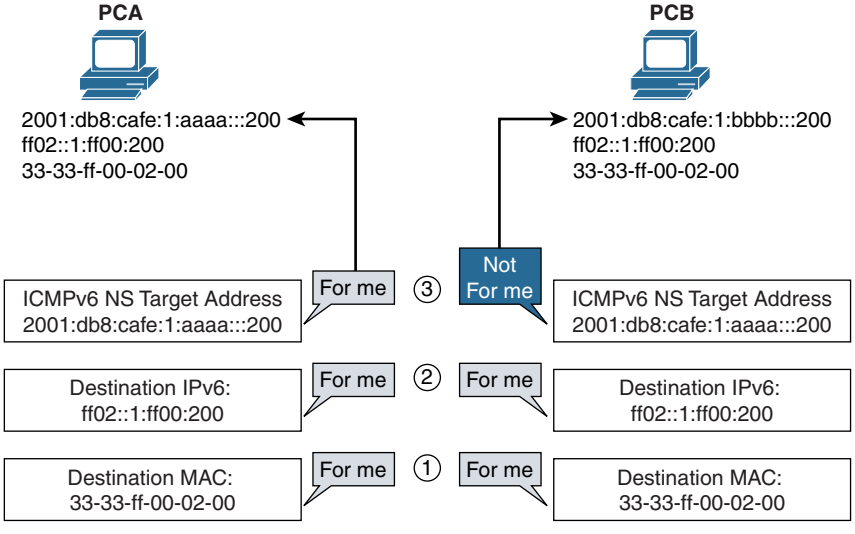


Figure 7-14 *PCA and PCB Processing an ICMPv6 Neighbor Solicitation Message*

In this example, PCA has a GUA address of 2001:db8:cafe:1:*aaaa*::200, and PCB has a GUA address of 2001:db8:cafe:1:*bbbb*::200 (the first hextet in the Interface IDs is different). A device has sent the NS message, searching for the MAC address 2001:db8:cafe:1:aaaa::200. This is the target address in the NS message.The following steps describes the scenario in Figure 7-14:

Step 1. Both NICs of PCA and PCB accept the Ethernet frame because they are both mapped to receive this multicast address, 33-33-ff-00-02-00.

Step 2. Both hosts pass the encapsulated packet to the IPv6 process. PCA and PCB will both accept and process the packet because they are both mapped to receive the solicited-node multicast address ff02::1:ff00:200.

Step 3. The encapsulated data is then sent to the ICMPv6 NS process. This is where the NS message finds its target device. The target address in the ICMPv6 message contains the IPv6 address the sender of the NS message is looking for. This matches the GUA address of PCA, so it accepts and processes the NS message. The target address doesn't match any of the addresses of PCB, so it discards it. PCA then sends a reply, an ICMPv6 Neighbor Advertisement message, with the MAC address of its Ethernet NIC (not the MAC address associated with the solicited-node multicast address).

Devices sharing the same solicited-node multicast address is uncommon on most networks because it is unlikely that devices will have the same low-order 24 bits in their Interface IDs. But if it does happen, it doesn't cause a problem.

Note Duplicate Address Detection (DAD) uses NS and NA messages. ICMPv6 Neighbor Solicitation and Neighbor Advertisement messages are discussed in Chapter 13.

One Solicited-Node Multicast Address for Multiple Unicast Addresses

It is possible and even common for a device to have the same solicited-node multicast address for both its global unicast and link-local unicast addresses. Example 7-8 shows how this might happen on the router R1.

Example 7-8 *R1 Multicast Groups*

```
R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::5AAC:78FF:FE93:DA00
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
  Joined group address(es):
    FF02::1                 ! All-IPv6 devices
    FF02::2                 ! All-IPv6 routers
    FF02::FB                ! Multicast DNS
! Solicited-node multicast for GUA
    FF02::1:FF00:1
! Solicited-node multicast for LLA
    FF02::1:FF93:DA00
<output omitted for brevity>

R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 address fe80::1 link-local
R1(config-if)# end

R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::1
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
  Joined group address(es):
! All-IPv6 devices
    FF02::1
! All-IPv6 routers
    FF02::2
! Multicast DNS
    FF02::FB
! Solicited-node multicast for GUA and LLA
      FF02::1:FF00:1
```

Initially, router R1 has different solicited-node multicast addresses for its global unicast and link-local unicast addresses. This is shown with the first **show ipv6 interface gigabitethernet 0/0** command.

Next, the link-local address on this interface is statically configured to fe80::1. The low-order 24 bits of this address are the same low-order 24 bits as in the global unicast address 2001:db8:cafe:1::1.

The second **show ipv6 interface gigabitethernet 0/0** command shows the single solicited-node multicast address ff02::1:ff00:1 for both the global unicast and link-local unicast addresses.

A device with two addresses sharing the same solicited-node multicast address can also occur when a device uses SLAAC to create its global unicast address. As discussed previously, SLAAC uses either the EUI-64 process or a random 64-bit value to create the Interface ID. In either case, it is common for an operating system such as Windows to use the same process for both the global unicast address and the link-local address. Even with the random 64-bit value, the Interface ID is the same for both unicast addresses. Since both the global unicast and link-local unicast addresses have exactly the same Interface ID, both these addresses have the same solicited-node unicast address. SLAAC is discussed in Chapter 9, "Stateless Address Autoconfiguration (SLAAC)."

Multicast Listener Discovery

IPv6 routers use Multicast Listener Discovery (MLD) to discover multicast clients (listeners) on the particular subnet. When an IPv6 router receives a multicast packet on an interface, it may need to forward that packet (or, more likely, a stream of packets) out one or more interfaces. When one of those interfaces is a LAN with end systems, the router needs to determine whether any of those end systems are members of the packet's multicast group. The IPv6 router uses MLDv2 for this purpose. This process isn't new to IPv6.

In IPv4, the management of multicast groups is accomplished by using Internet Group Management Protocol (IGMP). Hosts use IGMP to dynamically register themselves in a multicast group on a particular network. This is done by hosts sending IGMP messages to their local multicast router, telling the router which multicast addresses they want to receive traffic. Routers configured for IGMP listen to IGMP messages from hosts. Routers periodically send out queries to discover which multicast groups are still active—in other words, the groups with hosts that still want to receive traffic for that multicast address.

The router can then determine which multicast addresses are inactive and no longer have hosts requiring that traffic. With the first version of IGMP, there was no way for a host to explicitly leave a multicast group—informing the router that it wanted to leave that group. IGMPv2 includes a leave mechanism for the host to inform the router that it is withdrawing from that multicast group.

IPv6 uses ICMPv6 Multicast Listener Discovery for the same services, basing its functionality on IGMPv2. So, if you are familiar with IGMP, MLD is very similar. An important difference is that MLD uses ICMPv6, which is used to transport the MLD messages.

MLD is defined in RFC 2710, Multicast Listener Discovery for IPv6. MLD version 2 was defined in RFC 3810, *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*. Based on IGMPv3, MLDv2 extends the first version of MLD to support Source Specific Multicast (SSM) and is backward compatible with MLDv1. SSM provides the ability for a host to request multicast packets, not only for a destination multicast address but also from a specific source address. MLDv2 is the default version for Cisco IOS.

Note All MLD messages are sent with a link-local source IPv6 address, an IPv6 Hop Limit of 1, and a Hop-by-Hop Options header with the Router Alert options flag set. The Router Alert flag is necessary to require routers to examine MLD messages sent to multicast addresses in which the routers themselves have no interest. In other words, it provides an efficient mechanism whereby routers can recognize when to intercept packets not addressed to them without having to extensively examine every packet.

Note As with all other IPv4 and IPv6 packets, the IPv6 source address must be a unicast address.

There are three types of MLD messages:

- **Multicast Listener Query** (Type = decimal 130): The router periodically transmits host membership query messages to determine which multicast groups still have members on the router's directly attached networks. There are two subtypes of Multicast Listener Query messages:
  - **General Query:** This is used to learn which multicast addresses have listeners on an attached link. The General Query is sent to the link-scope all-nodes multicast address ff02::1, to all-IPv6 devices on the link.
  - **Multicast-Address-Specific Query:** This is used to learn whether a particular multicast address (multicast group) has any listeners on an attached link. An Address-Specific Query is sent to the multicast address being queried.
- **Multicast Listener Report** (Type = decimal 131): This message is sent by the listener to register for a multicast group. The listener can send this message in response to a query or can send it unsolicited, without waiting for a query from the router. If in response to a query, only one member of the multicast group needs to send a Multicast Listener Report. In MLDv1, these Listener Reports are sent to the multicast address being reported. This was changed in MLDv2, and Listener Reports are sent to a special multicast address ff02::16, all-MLDv2-capapable routers.
- **Multicast Listener Done** (Type = decimal 132): When a listener no longer wants to receive traffic for a particular multicast group, it sends a Multicast Listener Done message to inform the router that it is leaving that multicast group. Listener Done messages are sent to the link-scope all-routers multicast address (ff02::2).

Figure 7-15 illustrates an example of MLDv2 General Query and Listener Report messages, which are described in the steps that follow:

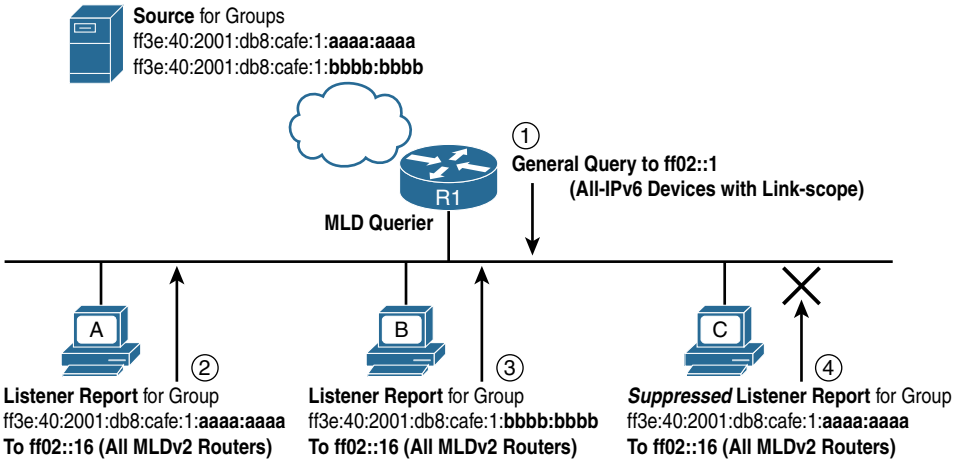


Figure 7-15 *MLDv2 General Query and Listener Reports*

Step 1. Router R1, which is the MLD-designated querier on the network, periodically sends a General Query message to the link-scope all-nodes multicast address ff02::1 to discover which multicast groups hosts want to receive the groups' traffic. The router with the lowest IPv6 address on a segment becomes the MLD-designated querier on that subnet. The election process and other MLDv2 information is described in RFC 3810, *MLDv2 for IPv6*.

Step 2. PC A, a member of the multicast group ff3e:40:2001:db8:cafe:1:aaaa:aaaa, receives the General Query and waits a random delay interval before sending a Multicast Listener Report. Not seeing a report from another host for this group, PC A transmits its Multicast Listener Report to ff02::16, all-MLDv2-capable routers. This informs router R1 that it still wants to receive traffic for this multicast address.

Step 3. PC B is a member of a different multicast group, ff3e:40:2001:db8:cafe:1:bbbb:bbbb. PC B sends a separate Multicast Listener Report to ff02::16, informing router R1 that it still wants to receive traffic destined for this multicast group.

Step 4. PC C is also a member of multicast group ff3e:40:2001:db8:cafe:1:aaaa:aaaa, but after waiting a random delay period, it sees that PC A has already sent a Multicast Listener Report to router R1 for the group. PC C suppresses sending the report because router R1 only needs to hear from one member of any multicast group.

Note RFC 3306, *Unicast-Prefix-Based IPv6 Multicast*, section 4, "Multicast Address Format," describes the format of the multicast address ff3e:40:2001:db8:cafe:1:aaaa:aaaa.

When a host no longer wants to receive traffic for a multicast group, it can inform the router by sending a Multicast Listener Done message. Figure 7-16 illustrates the process of a host leaving a multicast group, which is described in the following steps:

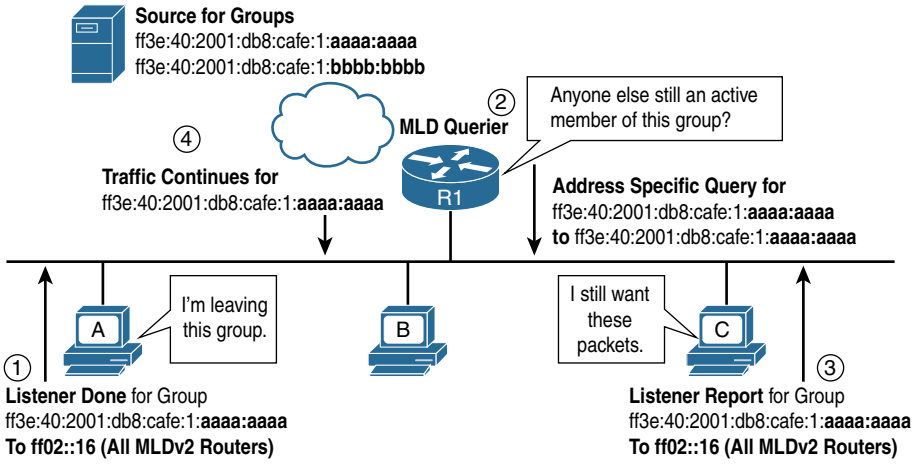


Figure 7-16 *MLDv2 Listener Done and Listener Reports*

Step 1. PC A no longer wants to receive traffic for the multicast group ff3e:40:2001:db8:cafe:1:aaaa:aaaa and sends a Multicast Listener Done message to link-scope all-routers multicast address ff02::2, to inform the router that it wants to leave the group.

Step 2. Router R1, the designated MLD querier for the network, receives the Listener Done message. The router maintains only a list of multicast groups on the network, not a list of the devices that are members of those groups. So when the router receives a Multicast Listener Done message, it must send a Multicast-Address-Specific Query to see whether any other devices still need to receive traffic for that multicast group. Router R1 sends a Multicast-Address-Specific Query specifically to the multicast group ff3e:40:2001:db8:cafe:1:aaaa:aaaa.

Step 3. PC C is still a member of the ff3e:40:2001:db8:cafe:1:aaaa:aaaa multicast group. PC C replies with a Multicast Listener Report for that group to inform the router that it still wants to receive traffic for that group. The Multicast Listener Report is sent to ff02::16, all-MLDv2-capapable routers.

Step 4. Router R1 receives the Multicast Listener Report from PC C and continues to send traffic to that multicast address. However, if router R1, after waiting a configurable time period, has not received a report from any host in that group, it stops forwarding traffic for that group.

Hosts communicate with routers to join and leave multicast groups, but routers also need a way to communicate with each other to route multicast traffic. A special protocol, PIM (Protocol Independent Multicast), is used for IPv4, and PIM6 is used for IPv6. Configuration of MLD and PIM6 are beyond the scope of this book.

MLD Snooping

MLD snooping allows the switch to examine MLD packets and make forwarding decisions based on their content. The default mode is for the switch to flood multicast packets out all ports except the incoming port.

With MLD snooping, shown in Figure 7-17, the switch can snoop Listener Reports from the hosts and create an entry in its Layer 2 forwarding table for the port on which it was received. If another host sends a Listener Report for the same group, the switch snoops their reports and adds them to the existing Layer 2 forwarding table entry. With MLD snooping enabled, multicast messages for this group are only sent out ports with hosts that are members of that group. In Figure 7-17, packets for ff3e:40:2001:db8:cafe:1:aaaa:aaaa would only be sent out ports for PC A and PC C.

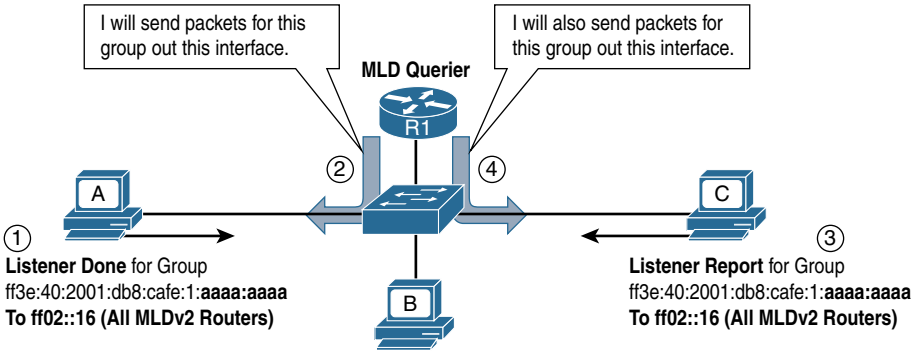


Figure 7-17 *MLD Snooping*

MLD snooping does not apply to solicited-node multicast addresses. Solicited-node multicast addresses are always forwarded out all ports because of the potentially huge forwarding tables needed to store these addresses.

Note For additional information on IPv6 multicast and multicast routing beyond the scope of this book, see resources by Tim Martin, Cisco Systems, including the YouTube video *IPv6 Summit 2015: IPv6 Multicast Technologies*, available at www.youtube.com/watch?v=H6bBiIPfYXM, and the excellent Cisco Press LiveLessons video series, *IPv6 Design and Deployment*. The Cisco Press book *Deploying IPv6 Networks* by Ciprian Popoviciu is another excellent resource for IPv6 multicast routing.

Summary

This chapter focuses on the IPv6 multicast addresses used for a device to send a single packet to multiple destinations simultaneously (one-to-many). An IPv6 multicast address has the prefix ff00::/8.

The format of an IPv6 multicast address is as follows:

- **ff00::/8:** The first 8 bits are 1-bits (ff).
- **Flags:** The next 4 bits are allocated for flags. The fourth flag is the Transient flag (T flag). The T flag denotes the two types of multicast address:
  - **Permanent (0):** These addresses, known as *predefined multicast addresses*, include both well-known and solicited multicast.
  - **Nonpermanent (1):** These are transient, or dynamically assigned, multicast addresses assigned by multicast applications.
- **Scope:** The Scope field defines the range to which routers can forward the multicast packet.
- **Group ID:** The next 112 bits represent the Group ID.

Well-known multicast addresses have the prefix ff00::/12. Well-known multicast addresses are predefined or reserved multicast addresses for assigned multicast groups. Some common well-known multicast groups used by ICMPv6 Neighbor Discovery Protocol include the following:

- **ff02::1 –** All-nodes multicast group for this link
- **ff02::2 –** All-routers multicast group for this link

Well-known multicast addresses are also by IPv6 routing protocols. For example, ff02::5 is for the all-OSPF routers, and ff02::a is for all-EIGRP routers. These addresses will be discussed in Chapter 15 EIGRP for IPv6 and Chapter 16 OSPFv3.

Every unicast address assigned to an interface also has an associated solicited-node multicast address. The solicited-node multicast prefix ff02:0:0:0:0:1:ff00::/104 is appended to the low-order 24 bits of the unicast address. IPv6's solicited-node multicast address provides a more efficient solution than a broadcast address. The multicast address is mapped to an Ethernet MAC address, which allows NICs to filter whether to accept the frame.

Multicast Listener Discovery version 2 (MLDv2) is based in Internet Group Management Protocol version 2 (IGMPv2) and is used by IPv6 routers to discover multicast clients (listeners) on the particular subnet. MLD snooping is implemented on the switch to filter multicast packets, other than solicited-node multicasts, so they are only forwarded out ports where there is a member of that group.

Review Questions

1. What is the prefix of an IPv6 multicast address?
2. What is the scope of an ff02::/16 multicast address?
3. What is the scope of an ff05::/16 multicast address?
4. What is the well-known multicast address for all IPv6 devices with link-local scope?
5. What is the well-known multicast address for all IPv6 routers?
6. What is the advantage of a multicast address over a broadcast address on an Ethernet LAN?
7. Solicited-node multicast addresses are automatically created for what types of addresses?
8. What type of ICMPv6 Neighbor Discovery Protocol message uses a solicited-node multicast address?
9. Can two devices on the same subnet have the same solicited-node multicast address? Why or why not?
10. Can a device have the solicited-node multicast address for both its global unicast and link-local unicast addresses? Why or why not?
11. Map each of the following unicast addresses to solicited-node multicast and Ethernet MAC addresses.
    a. 2001:db8:bee:47:0201:41ff:fea1:1067
    b. fe80::0201:41ff:fea1:1067
    c. 2001:db8:deed:30:feaf:d899:10bc:7
    d. 2001:db8:feed:1:a:1ab::7000
    e. 2001:db8:feed:1:a:2ab::7000
12. What protocol do IPv6 routers use to discover if there are any multicast clients (listeners) on the particular subnet?
13. What protocol can be implemented on an Ethernet switch to enable to filter frames that have a multicast address for the destination MAC address?

References

RFCs

RFC 2373, *IP Version 6 Addressing Architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc2373.txt, July 1998.

RFC 2375, *IPv6 Multicast Address Assignments*, R. Hinden, Ipsilon Networks, www.ietf.org/rfc/rfc2375.txt, July 1998.

RFC 3306, *Unicast-Prefix-Based IPv6 Multicast Addresses*, B. Haberman, www.ietf.org/rfc/rfc3306.txt, August 2002.

RFC 3956, *Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address*, P. Savola, CSC/FUNET, www.ietf.org/rfc/rfc3956.txt, November 2004.

RFC 3513, *Internet Protocol Version 6 (IPv6) Addressing Architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc3513.txt, April 2003.

RFC 4007, *IPv6 Scoped Address Architecture*, S. Deering, Cisco Systems, www.ietf.org/rfc/rfc4007.txt, March 2005.

RFC 4291, *IP Version 6 Addressing Architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc4291.txt, February 2006.

RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*, Y. Narten, IMB, www.ietf.org/rfc/rfc4861.txt, September 2007.

RFC 7042, IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters, D. Eastlake 3rd, www.ietf.org/rfc/rfc7042.txt, October 2013.

Websites, Videos, and Books

Netsh Commands for Interface Internet Protocol Version 6 (IPv6), technet.microsoft.com/en-us/library/cc753156(v=ws.10).aspx.

IPv6 Summit 2015: IPv6 Multicast Technologies, by Tim Martin, www.youtube.com/watch?v=H6bBiIPfYXM.

Cisco Press LiveLessons: IPv6 Design and Deployment, by Tim Martin, www.ciscopress.com/store/ipv6-design-and-deployment-livelessons-9780134655512, ISBN-10: 0-13-465551-6.

Deploying IPv6 Networks, by Ciprian Popoviciu, ISBN-10: 1-58705-210-5.

This page intentionally left blank

Part III

Dynamic IPv6 Addressing

This page intentionally left blank

Chapter 8

Basics of Dynamic Addressing in IPv6

This chapter introduces dynamic IPv6 address allocation, which provides a method for devices to create or obtain their global unicast addresses and other addressing information.

This chapter takes an overall look at the three methods for dynamic addressing and prepares you for the next three chapters, which discuss the configurations and details of each of these methods.

To help set the stage for more detailed discussion, this chapter discusses the following:

- It compares how dynamic address allocation is done with IPv4 using stateful DHCPv4 (DHCP for IPv4).
- It discusses the ICMPv6 Router Advertisement message used to suggest to devices which method they should use to obtain their IPv6 addressing information.
- It examines the three Router Advertisement flags—the A, O, and M flags. These flags determine which dynamic addressing method is being suggested by the RA message.
- It provides a brief overview of the three methods: Stateless Address Autoconfiguration (SLAAC), SLAAC with stateless DHCPv6, and stateful DHCPv6.
- It discusses basic DHCPv6 operations, including DHCPv6 message types.

The information in this chapter allows you to later examine SLAAC, stateless DHCPv6, and stateful DHCPv6 more thoroughly and with a better overall understanding.

Dynamic IPv4 Address Allocation: DHCPv4

This section provides a review of how IPv6 handles dynamic address allocation. An IPv4 address and related addressing information (subnet mask, default gateway, DNS server, and so on) can be assigned one of two ways: statically (manual configuration) or dynamically. Dynamic address allocation in IPv4 means one thing: DHCPv4.

The process begins with the client device (such as the host computer or printer) configured to obtain its addressing information dynamically. This is the default on most host computer operating systems. Figure 8-1 shows the DHCPv4 messages exchanged between the DHCPv4 client and server. Here is a brief overview of this process and these messages:

Step 1. **DHCPv4 Discover:** The IPv4 client sends a broadcast (Ethernet and IPv4) requesting the services of a DHCPv4 server. The Request may include its last-known IPv4 address or 0.0.0.0.

Step 2. **DHCPv4 Offer:** One or more DHCPv4 servers respond with a DHCPv4 Offer message. This is the reserved IPv4 address, subnet mask, and lease duration offered by the DHCPv4 server to the client. It also includes the client's MAC address, which the server uses to bind the IPv4 address to this client. The Offer may also include other options, such as a default gateway address and a list of DNS server addresses.

Step 3. **DHCPv4 Request:** The client responds with a DCHPv4 Request, to request the offered address. The Request is a broadcast to all DHCPv4 servers, and if there are multiple offers, the servers are informed whose offer has been accepted.

Step 4. **DHCPv4 Acknowledgement:** The final step is the DHCPv4 server sending an Acknowledgement to the client with the IPv4 address and other information.

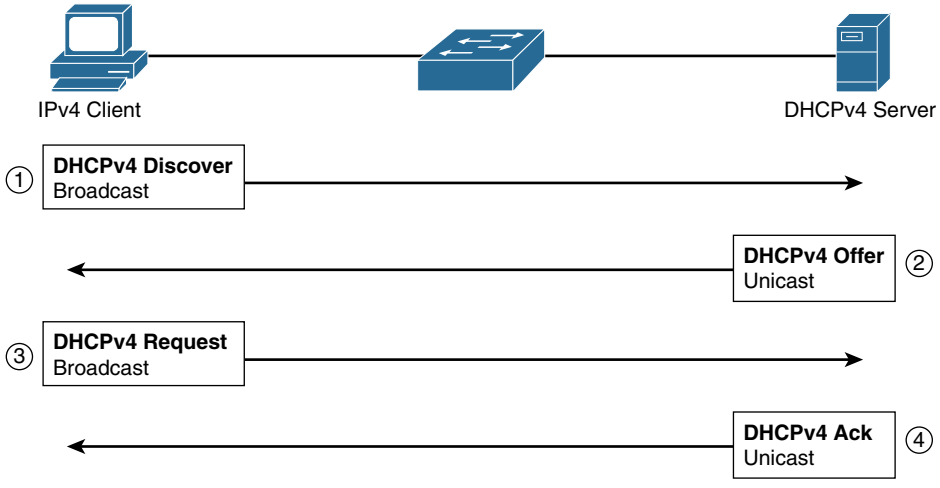


Figure 8-1 *DHCPv4 Messages*

Some of the details and options for DHCPv4 aren't included here. However, as you will see, in comparison to DHCPv6, the process for DHCPv4 is fairly simple and has relatively few options.

Note Although it is more common to refer to DHCP for IPv4 as just DHCP, we use DHCPv4 throughout this chapter to avoid any confusion with DHCPv6.

Dynamic IPv6 Address Allocation

In previous chapters we introduced the three methods for dynamic IPv6 addressing:

- **Method 1:** Stateless Address Autoconfiguration (SLAAC)
- **Method 2:** SLAAC and a stateless DHCPv6 server
- **Method 3:** Stateful DHCPv6 server

Figure 8-2 shows all the methods for configuring an IPv6 global unicast address, including manual and dynamic methods. Chapter 5, "Global Unicast Address," examines the manual configuration options. This chapter and the three that follow examine the various dynamic configuration methods shown in Figure 8-2.

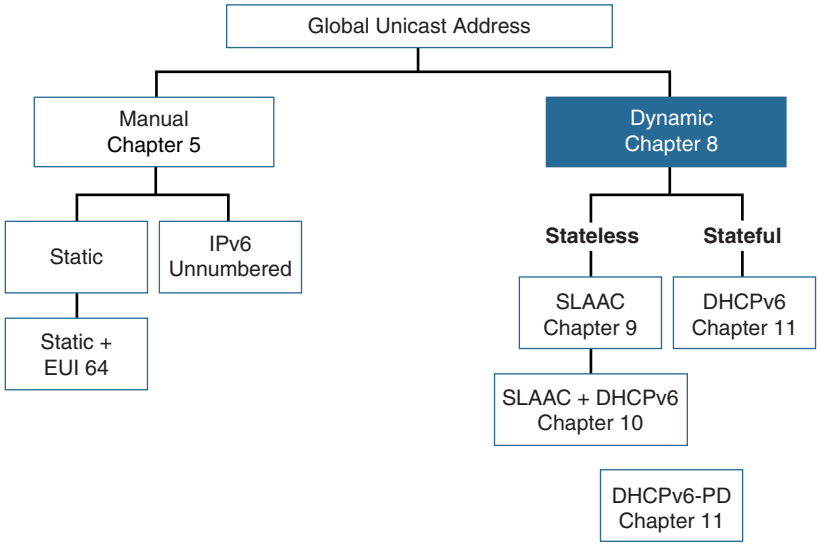


Figure 8-2 *Global Unicast Address Configuration Methods*

The method DHCPv6-PD in Figure 8-2 is DHCPv6 Prefix Delegation. DHCPv6-PD is used for service provider allocation of IPv6 prefixes to homes or small businesses.

With service provider deployment of IPv6, many ISPs have been deploying IPv6 access to their residential customers. In conjunction with widely deployed "always on" media such as cable and DSL Internet access, an efficient mechanism for delegating address prefixes to the customers' sites was needed. RFC 3769, *Requirements for IPv6 Prefix*

Delegation, addresses the delegation mechanism intended to automate the process of informing the customer's networking equipment of the prefixes to be used at the customer's site. DHCPv6 Prefix Delegation is discussed in Chapter 11, "Stateful DHCPv6."

Configuring a device to receive its addressing dynamically in IPv6 is not any different than in IPv4. Figure 8-3 shows an example for both Windows OS and Mac OS. Notice that the Mac OS prompt to configure IPv6 states "Automatically" instead of "Using DHCP," as it does for IPv4. This is because the RA message may suggest something other than DHCPv6.

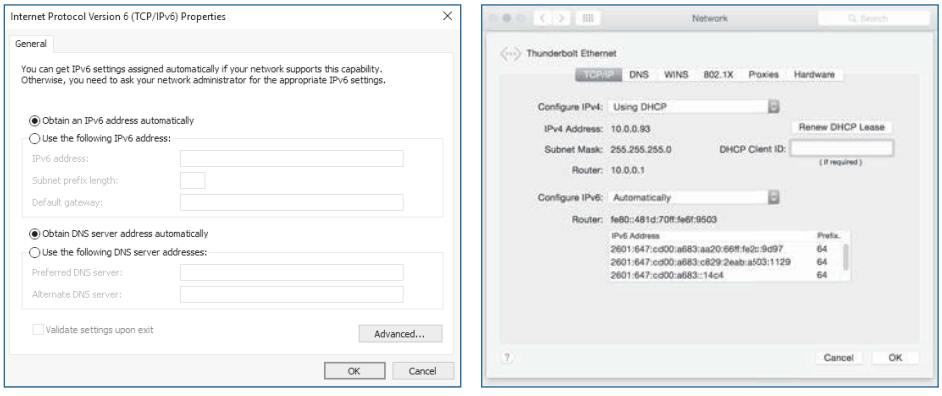


Figure 8-3 *Windows OS (Left) and Mac OS (Right) Dynamic Address Allocation*

ICMPv6 Router Solicitation and Router Advertisement Messages

Dynamic IPv6 address allocation begins with the ICMPv6 Router Solicitation and Router Advertisement messages. These messages are used for messaging between routers and devices on the same link.

Note The Router Solicitation and Router Advertisement messages are two of the five messages in ICMPv6 Neighbor Discovery Protocol (NDP or ND). The Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages are used for messaging between any two devices, and the Redirect message is used to optimize next-hop routing.

ICMPv6 NDP messages such as Router Solicitation and Router Advertisement messages are ICMPv6 messages encapsulated in IPv6, as shown in Figure 8-4.

Figure 8-4 *ICMPv6 RA/RS Message over IPv6*

Note A detailed discussion of all ICMPv6 NDP messages, including the RA and RS messages, is provided in Chapter 13, "ICMPv6 Neighbor Discovery." However, the RA and RS messages are also discussed in the next four chapters as they relate to the SLAAC and DHCPv6 address allocation methods.

IPv6 Router Advertisements are automatically sent on Ethernet and FDDI interfaces if IPv6 unicast routing is enabled on the interface. (FDDI is almost obsolete.) A Router Advertisement is sent on a subnet by an *IPv6 router*. A Cisco router is *not* an IPv6 router by default. To be an IPv6 router, the following global configuration command must be used:

```
Router(config)# ipv6 unicast-routing
```

A router doesn't have to be an IPv6 router to have an interface configured with an IPv6 address (link-local unicast, global unicast, or unique local unicast). Once an interface has an IPv6 address, it becomes a member of the all-IPv6 devices multicast group (ff02::1).

Configuring a Cisco router as an IPv6 router (using the **ipv6 unicast-routing** command) makes the following possible:

- The router can forward IPv6 packets transiting the router.
- It can be configured with a dynamic IPv6 routing protocol.
- It can send ICMPv6 Router Advertisement messages out Ethernet interfaces.
- Any interface with an IPv6 address becomes a member of the all-IPv6 routers multicast group (ff02::2).

Note A non-IPv6 router can be configured with IPv6 static routes, but it can only forward packets that originated from that router.

RA messages are originated by routers to advertise their presence and link-specific parameters such as the link prefix, prefix length, default gateway, and link maximum transmission unit (MTU). Router Advertisements are sent (using the destination IPv6 address) to the *all-IPv6 devices multicast* address (ff02::1), which is essentially the same as a broadcast. RA messages can also be sent as unicast, directly to the sender of the Router Solicitation message. This can be helpful in environments with a lot of mobile devices. This option is discussed in Chapter 9, "Stateless Address Autoconfiguration (SLAAC)."

The source IPv6 address is the link-local address of the interface. The source IPv6 address of the RA message is the address that receiving devices will use as their default gateway address.

Cisco routers send Router Advertisements:

- Every 200 seconds
- In response to a Router Solicitation message

Router Solicitation messages are sent by devices configured to obtain the IPv6 addresses dynamically. RS messages are sent to the *all-IPv6 routers multicast* address (ff02::2). The source IPv6 address is either the device's link-local address or an unspecified IPv6 address (::).

Figure 8-5 shows the interaction between WinPC's Router Solicitation message and router R1's Router Advertisement message, which involves the following steps:

Step 1. WinPC is configured to obtain its IPv6 addressing automatically. Since booting up, WinPC has not seen a Router Advertisement message, so it sends out a Router Solicitation message to inform the local IPv6 router that it needs an RA message. The Router Solicitation message is encapsulated in an IPv6 packet.

Router Solicitation Message

- **Source IPv6 address:** WinPC's link-local address, fe80::d0f8:9ff6:4201:7086
- **Destination IPv6 address:** All-IPv6 routers multicast address, ff02::2

Step 2. Router R1 receives the Router Solicitation message and responds with a Router Advertisement. The Router Advertisement suggests to the devices on the subnet how to obtain or create their global unicast address and other addressing information. Included in the RA message are the prefix and prefix length of the link. The default gateway address is R1's source IPv6 address, its link-local address.

Router Advertisement Message

- **Source IPv6 address:** Router R1's link-local address, fe80::1
- **Destination IPv6 address:** All-IPv6 devices multicast address, ff02::1
- **Suggested IPv6 address method (one of the three):**
  - **Method 1:** Stateless Address Autoconfiguration (SLAAC)
  - **Method 2:** SLAAC and a stateless DHCPv6 server
  - **Method 3:** Stateful DHCPv6 server

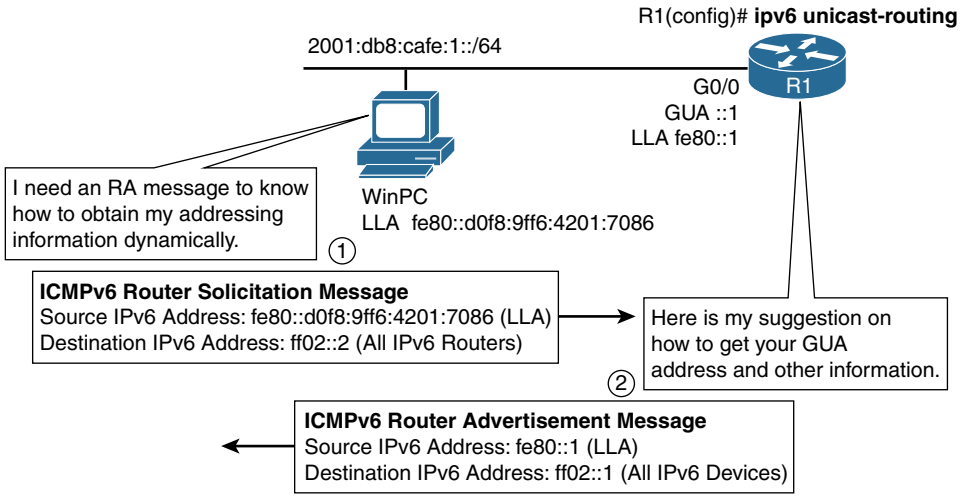


Figure 8-5 *Interaction Between ICMPv6 Router Solicitation and Router Advertisement Messages*

The method in the Router Advertisement is only a suggestion to the clients. The client operating systems may be configured to ignore the RA message. Most organizations prefer to statically configure IPv6 addressing information on specific systems, such as servers and printers. For extra precautions, you may want these systems to ignore the RA message. In Windows you can do this with the **netsh interface ipv6 set interface "Local Area Connection" routerdiscovery=disabled** command.

Router Advertisement Methods and the A, O, and M Flags

The Router Advertisement suggests to devices how to create or obtain a global unicast address and other addressing information for communicating on the link. The RA message uses three flags to tell devices how this is to be done:

- **Address Autoconfiguration flag (A flag):** When set to 1 (on), this flag tells the receiving host to use SLAAC to create its global unicast address. SLAAC allows the host to create its own GUA address by combining the prefix in the RA message with a self-generated Interface ID. The method the host uses to create its own Interface ID is up to the operating system. There are two options for creating the Interface ID:
  - EUI-64 process
  - Random 64-bit value
- **Other Configuration flag (O flag):** When set to 1 (on), this flag tells the host to get addressing information other than its global unicast address from a stateless DHCPv6 server. This information may include DNS server addresses and a domain name.

- **Managed Address Configuration flag (M flag):** When set to 1 (on), this flag tells a host to use a stateful DHCPv6 server for its global unicast address and all other addressing information. This is similar to DHCP for IPv4. The only information the host uses from the RA message is from the packet's source IPv6 address, which it uses as the default gateway address.

Note Unlike with DHCPv4, a DHCPv6 server does not provide the default gateway address. The point is that there is no better source for the address of the default gateway than the router itself.

It's important to understand the difference between a stateful DHCPv6 server and a stateless DHCPv6 server. A *stateful DHCPv6 server* assigns global unicast addresses to clients and maintains state information, a record of which client has been assigned which address. A *stateless DHCPv6 server* does not provide the global unicast address. It provides generic information—the same information to all clients, such as DNS server addresses and a domain name.

Table 8-1 shows the settings of the three flags and how they correspond to the RA message's three methods for address allocation. The methods and their flags are discussed in the following sections. A 1 indicates that the flag is set to "on," and a 0 indicates "off." Note that these are not the only flags contained in the RA message, but they are the ones pertinent to this topic.

Table 8-1 *Router Advertisement: Address Allocation Methods and RA Flags*

RA Address Allocation Method	A Flag (SLAAC)	O Flag (Stateless DHCPv6)	M Flag (Stateful DHCPv6)
Method 1: SLAAC (default)	1 (on)	0 (off)	0 (off)
Method 2: SLAAC and stateless DHCPv6	1 (on)	1 (on)	0 (off)
Method 3: Stateful DHCPv6	0 (off)	N/A	1 (on)

Router Advertisement flags are configured at the interface, which allows for different types of RA messages on each link.

You might be wondering what happens if all three flags are set to 1. When both the O and M flags are set to 1, the O flag becomes irrelevant. The device then uses stateful DHCPv6 for all its configuration information (except for the default gateway). If the

A flag is also set to 1, many operating systems (such as Windows) configure an additional address using SLAAC. The end result is that the device may have an address it created using SLAAC and another address obtained from a stateful DHCPv6 server.

The Address Autoconfiguration flag (A flag) defaults to 1. To set the A flag to 0, use the following syntax:

```
Router(config)# interface interface-type/interface-number
Router(config-if)# ipv6 nd prefix prefix/prefix-length no-autconfig
```

The Other Configuration flag (O flag) has a default setting of 0. To set to the O flag to 1, use the following syntax:

```
Router(config)# interface interface-type/interface-number
Router(config-if)# ipv6 nd other-config-flag
```

The Managed Address Configuration flag (M flag) has a default setting of 0. To set the M flag to 1, use the following syntax:

```
Router(config)# interface interface-type/interface-number
Router(config-if)# ipv6 nd managed-config-flag
```

Note By default the A flag is set to 1, and the O and M flags are set to 0. You can use the configuration commands to modify these defaults. Use the **no** option to set a flag back to its default setting. For example, **no ipv6 nd other-config-flag** sets the O flag back to its default setting of 0 (off). Chapters 9, 10, and 11 provide configuration examples using these three flags.

Method 1: Stateless Address Autoconfiguration (SLAAC)

The default behavior of a Cisco router's Router Advertisement message is SLAAC only. As shown in Figure 8-6, this method tells the receivers that the RA message contains all the addressing information a device needs, including the prefix that the device will use to create its own global unicast address using SLAAC. The three RA flags are set to their default settings:

- **A flag = 1:** Use SLAAC to create a global unicast address
- **O flag = 0:** No other information needed from a stateless DHCPv6 server
- **M flag = 0:** Do not need to communicate with a stateful DHCPv6 server

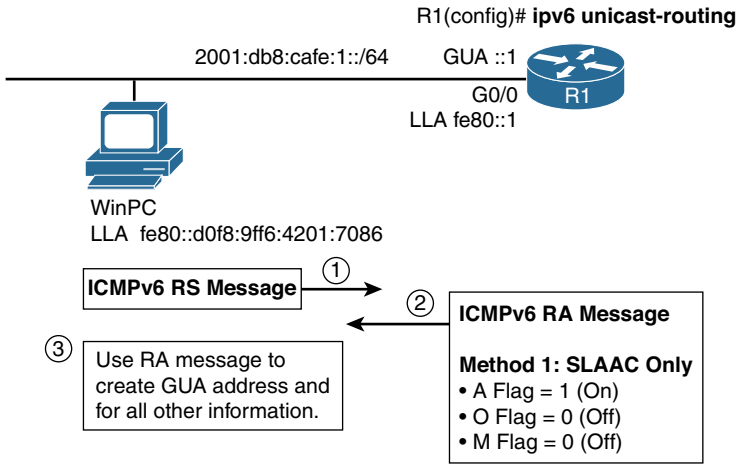


Figure 8-6 *Method 1: SLAAC Only*

These are the default settings for the A, O, and M flags, so no modification to these flags is necessary.

Devices set to obtain their IPv6 address dynamically:

- They use the prefix in the RA message to generate a global unicast address.
- They use other information in the RA message, such as prefix length and link MTU. Other information may also include domain name and DNS server addresses. The RA message does not include the domain name and DNS address by default. (This is discussed further in Chapter 9).
- They use the source IPv6 address of the packet, a link-local unicast address, as the default gateway address.
- There is no other information needed from a stateless or stateful DHCPv6 server.

To summarize, this RA message to devices says, "Use SLAAC to create your GUA address, and the RA message has everything else you need. There is no need to communicate with any type of DHCPv6 server."

In Chapter 9 we take a much closer look at SLAAC and the effects it has on host operating systems, including the following:

- Verification of SLAAC addresses on Windows, Linux, and Mac OS hosts
- Analysis of the RS and RA messages using debug and Wireshark
- A discussion of temporary addresses and issues concerning privacy
- The different states of an autoconfigured address

- Configuration of a Windows host to generate the Interface ID of a global unicast address using either random 64 bits or the EUI-64 process
- Configuration of a Windows host to enable or disable the creation of temporary global unicast addresses for privacy
- Default address selection, the process a device uses to determine which address (a link-local address or one or more global unicast addresses) to use

Method 2: SLAAC with Stateless DHCPv6

The SLAAC with stateless DHCPv6 method involves setting the Other Configuration flag (O flag) to 1. With this method the device creates its own global unicast address using SLAAC, just as happens in the first method. It also uses information such as the link MTU contained in the RA. However, having the O flag set to "on" tells the receiving device that other information is available from a stateless DHCPv6 server, as shown in Figure 8-7. The three RA flags are as follows:

- **A flag = 1:** Use SLAAC to create a global unicast address
- **O flag = 1:** Communicate with a stateless DHCPv6 server for other addressing information
- **M flag = 0:** Do not need to communicate with a stateful DHCPv6 server

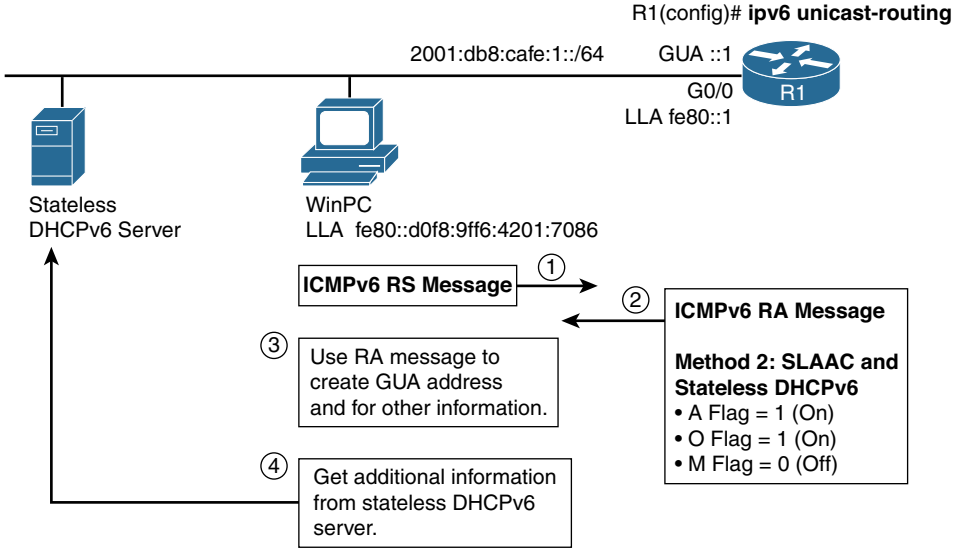


Figure 8-7 *Method 2: SLAAC and Stateless DHCPv6*

Use the following commands to change the O flag on R1's G0/0 from its default setting of "off" to "on":

```
R1(config)# interface g 0/0
R1(config-if)# ipv6 nd other-config-flag
```

Devices set to obtain their IPv6 address dynamically do the following:

- They use the prefix in the RA message to generate a global unicast address.
- They use other information in the RA message, such as prefix length and link MTU.
- They use the source IPv6 address of the packet, a link-local unicast address, as the default gateway address.
- They contact a stateless DHCPv6 server for additional information, such as domain name and DNS server addresses. The RA message does not specify what information can be obtained from the stateless DHCPv6 server.

To summarize, this RA message to devices says, "Use SLAAC to create your GUA address. There is some other information in the RA message. However, you will need to communicate with a stateless DHCPv6 server for other configuration information."

In Chapter 10, "Stateless DHCPv6," we discuss stateless DHCPv6 in more detail, including the following:

- Analysis of an RA message indicating SLAAC with stateless DHCPv6 using debug and Wireshark
- Configuration and verification of a Cisco router as a stateless DHCPv6 server
- Verification of SLAAC address and stateless DHCPv6 information on Windows host

Method 3: Stateful DHCPv6

The stateful DHCPv6 method involves modifying two flags: the Managed Address Configuration flag (M flag) and the Address Autoconfiguration flag (A flag). Illustrated in Figure 8-8, this method tells the receiving device to obtain its global unicast address and other information, except the default gateway address, from a stateful DHCPv6 server. The three RA flags are as follows:

- **A flag = 0:** Do not use SLAAC to create a global unicast address
- **O flag = 0:** No need to communicate with a stateless DHCPv6 server
- **M flag = 1:** Obtain the global unicast address and other information from a stateful DHCPv6 server

Note If the M flag is set to 1, then the O flag is redundant and can be ignored because the DHCPv6 server returns all available configuration information.

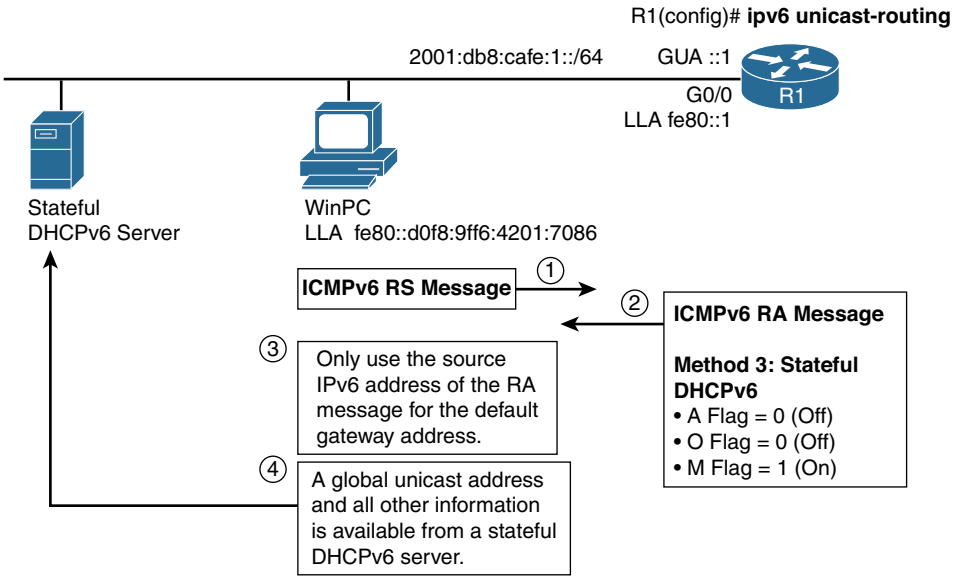


Figure 8-8 *Method 3: Stateful DHCPv6*

Use the following commands to change the A flag from its default setting of 1 to 0, and to change the M flag from its default setting of 0 to 1:

```
R1(config)# interface g 0/0
R1(config-if)# ipv6 nd prefix 2001:db8:cafe:1::/64 no-autconfig
R1(config-if)# ipv6 nd managed-config-flag
```

Devices set to obtain their IPv6 address dynamically do the following:

- They use the source IPv6 address of the packet, a link-local unicast address, of the RA message as the default gateway address.
- They contact a stateful DHCPv6 server for a global unicast address and all other information, such as domain name and DNS server addresses.

To summarize, this RA message to devices says, "Only use the RA message for the default gateway address. Do not use SLAAC to generate a global unicast address. Get a GUA address and all other information from a stateful DHCPv6 server."

Notice that the A flag has been set to 0. This prevents an operating system such as Windows from using both stateful DHCPv6 and SLAAC for its global unicast addresses. If both the A flag and the M flag are set to 1, Windows uses SLAAC to generate a GUA

address and obtains a second GUA address from the stateful DHCPv6 server. This doesn't create any problems for the device, but it may conflict with the network policy. This is discussed in more detail in Chapter 11.

In Chapter 11 we discuss stateful DHCPv6 in more detail, including the following:

- Analysis of an RA message indicating stateful DHCPv6 using debug and Wireshark
- Configuration and verification of a Cisco router as a stateful DHCPv6 server
- Verification of stateful DHCPv6 information on a Windows host
- How to modify the RA message so a device receives a single global unicast address from a stateful DHCPv6 server and does not use SLAAC for additional addresses
- Configuration of DHCPv6 relay so clients can have remote access to a DHCPv6 server

DHCPv6 Services

The router in its Router Advertisement message suggests the method that an IPv6 client uses to dynamically receive its configuration information. This may include communications with a stateless or stateful DHCPv6 server. This section discusses the process of DHCPv6 and introduces its new terminology, message types, and operations. In Chapters 10 and 11 we will take a closer look at how to configure and verify stateless and stateful DHCPv6.

Much has been made about the stateless nature of IPv6 address assignment. A device can receive its IPv6 address and other configuration information without the services of a DHCPv6 server. Through the use of ICMPv6 Router Advertisements and the ability for a device to create its own Interface ID, SLAAC provides dynamic addressing without the services of DHCP. To some, this stateless form of address autoconfiguration is all that an IPv6 network should ever need for dynamic address assignment. However, for many others, stateful DHCPv6 services are an important benefit to network operations—even a requirement for managing address allocations.

Some do not see the need for stateful DHCPv6 services; many others disagree. Since the early days of IPv6, there has been much debate between the two sides.

The fact is that both stateless and stateful address autoconfiguration are a reality in IPv6. DHCPv6 is defined in RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. There has been a lot of work done on this specification over the years. In 2001, Steve Deering from Cisco Systems stated at IETF 51, Proceedings of the 51st Internet Engineering Task Force, that the DHCPv6 specification has the highest revision number of any Internet draft.

As mentioned earlier, there are two forms of DHCPv6:

- **Stateful DHCPv6 services:** RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*
- **Stateless DHCPv6 services:** RFC 3736, *Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6*

Stateful DHCPv6 provides the same autoconfiguration services that DHCP performs for IPv4 (DHCPv4). Hosts receive all their addressing and other configuration information directly from a DHCPv6 server, except for the default gateway address. The differences with stateless DHCPv6 are that hosts receive their addressing information from Router Advertisements and obtain other configuration parameters from a DHCPv6 server.

Although similar in what it provides, DHCPv6 is almost an entire rewrite of DHCPv4, and the two protocols are independent of each other. If you are currently running a dual-stack network that uses DHCP, you need two separate DHCP services running, one for each protocol.

The process and messaging used in DHCPv6 are not very different than with DHCPv4. However, in the DHCPv4 world, you configure the client to use DHCPv4 simply by choosing an option such as "Obtain an IP address automatically" in Windows or "Using DHCP" on a Mac. In the world of IPv6, you configure the client the same way, but it does not know whether its addressing information will come from just the Router Advertisement (SLAAC), a combination of both SLAAC and stateless DHCPv6, or a stateful DHCPv6 server.

Note Before deploying hosts that implement dynamic IPv6 addressing techniques into the network, it is a good idea to first verify the behavior of the host operating system. It has been observed that not all operating systems honor the A, O, and M flags in the Router Advertisement, and you may get unexpected results.

The discussion here focuses specifically on the fundamental concepts and the router's role as a stateless DHCPv6 server. By no means is this meant to be the definitive guide on DHCPv6. For more information on DHCPv6 and implementing DCHPv6 on Cisco routers, read the following:

- RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*
- *Cisco IOS IPv6 Implementation Guide*

DHCPv6 Terminology and Message Types

DHCPv6 has its roots in DHCPv4, and the two have many similarities. For the most part, DHCPv6 provides the same services as DHCPv4—but with some variations. This section describes the terminology and the messages types used in DHCPv6.

These three DHCPv6 terms are identical to their counterparts in DHCPv4:

- **DHCPv6 client:** A DHCPv6 client initiates the process by sending a request to obtain its configuration parameters from one or more DHCPv6 servers.
- **DHCPv6 server:** The DHCPv6 server is configured to respond to the DHCPv6 requests from DHCPv6 clients. These clients might or might not be on the same link as the server.

- **DHCPv6 relay agent:** Quite often the DHCPv6 server and clients are on different networks or links. A relay agent is an intermediary device—typically a router—that receives the client's request and forwards it to one or more DHCPv6 servers on another network. The DHCPv6 relay agent operation is transparent to the client.

The next three terms—DUID, IA, and IAID—are new to DHCPv6; however, the process of using them with DHCPv4 has been under way for a few years. Understanding these terms is important when you're configuring some of the various options provided by DHCPv6:

- **DUID (DHCP Unique Identifier):** Every DHCPv6 participant, client, and server has a DUID that uniquely identifies the device. Each DHCPv6 server and client has exactly one DUID. The DUID allows the clients and servers to identify each other. A DUID consists of a 2-byte type code followed by a variable number of bytes that make up the actual identifier. Not including the type code, a DUID can be no more than 128 bytes. RFC 3315 defines three types of DUIDs:
  - Link-layer address plus time
  - Vendor-assigned unique ID based on the private enterprise number maintained by the Internet Assigned Numbers Authority (IANA)
  - Link-layer address

Note Selection of the DUID is device-independent. DUIDs are treated as "opaque values" and have no significance other than to uniquely identify the DHCPv6 client or server. The difference between the three types of DUIDs and the selection process is beyond the scope of this book. Please refer to RFC 3315 for more information.

- **IA (Identity Association):** An IA is a collection of addresses assigned to a client. A client has at least one IA assigned for each interface using the services of DHCPv6. Each IA has an IAID (Identity Association Identifier) that is used to uniquely identify the ID and is assigned by the client.
- **IAID (Identity Association Identifier):** Each IA has an IAID that is chosen by the client and is unique among all IAIDs belonging to that client. The IAID identifies a specific interface on the device. There was a time when we assumed that a host would only have one network interface. But times have changed and devices may have multiple network interfaces including an Ethernet NIC and a wireless NIC. Each interface on the DHCPv6 client or server is identified using an IAID.

DHCPv6 clients and servers use the following multicast addresses:

- **All_DHCP_Relay_Agents_and_Servers (ff02::1:2):** All DHCPv6 servers and relay agents are members of this link-local scope multicast group. Clients use this multicast address to communicate with DHCPv6 servers and relay agents on their link.

- **All_DHCP_Servers (ff05::1:3):** All DHCPv6 servers are members of this site-local multicast group. Relay agents use this multicast address to send messages to all DHCPv6 servers within a site or when they do not know the unicast address of the server.

DHCPv6 uses the following User Datagram Protocol (UDP) ports:

- **UDP port 546:** DHCPv6 servers and relays send messages to clients using UDP destination port 546. Clients listen for DHCPv6 messages on UDP port 546.
- **UDP port 547:** DHCPv6 clients send messages to servers and relay agents using UDP destination port 547. Servers and relay agents listen for DHCPv6 messages on UDP port 547.

DHCPv6 defines various message types for client/server communications. The complete list of DHCPv6 message types is described in RFC 3315 and listed in Table 8-2 for your reference. The following section of this chapter explains how many of these messages are used. For a more comprehensive understanding, refer to the resources mentioned earlier in the chapter.

These four DHCPv6 message types are the primary messages used to communicate between a DHCPv6 client and server:

- **SOLICIT (1):** DHCPv6 clients use a SOLICIT message to locate servers.
- **ADVERTISE (2):** A server sends an ADVERTISE message in response to a client's SOLICIT message to indicate that it is available for DHCPv6 service.
- **REQUEST (3):** A client sends a REQUEST message to request configuration parameters, including IPv6 addresses, from a specific DHCPv6 server.
- **REPLY (7):** A server sends a REPLY message containing assigned addresses and configuration parameters in response to a client's REQUEST message. A server also sends a REPLY message in response to a SOLICIT message when the Rapid Commit Option (described later in this chapter) is used. A server can also send a REPLY message in response to an INFORMATION-REQUEST, CONFIRM, RELEASE, or DECLINE message.

Table 8-2 *DHCPv6 Message Types Described in RFC 3315*

DHCPv6 Message Type	Description
SOLICIT (1)	A client sends a SOLICIT message to locate servers.
ADVERTISE (2)	A server sends an ADVERTISE message to indicate that it is available for DHCPv6 service, in response to a SOLICIT message received from a client.
REQUEST (3)	A client sends a REQUEST message to request configuration parameters, including IPv6 addresses, from a specific server.

DHCPv6 Message Type	Description
CONFIRM (4)	A client sends a CONFIRM message to any available server to determine whether the addresses it was assigned are still appropriate to the link to which the client is connected.
RENEW (5)	A client sends a RENEW message to the server that originally provided the client's addresses and configuration parameters to extend the lifetimes on the addresses assigned to the client and to update other configuration parameters.
REBIND (6)	A client sends a REBIND message to any available server to extend the lifetimes on the addresses assigned to the client and to update other configuration parameters; the client sends this message after it receives no response to a RENEW message.
REPLY (7)	A server sends a REPLY message containing assigned addresses and configuration parameters in response to a SOLICIT, REQUEST, RENEW, or REBIND message received from a client. A server sends a REPLY message containing configuration parameters in response to an INFORMATION-REQUEST message. A server sends a REPLY message in response to a CONFIRM message, confirming or denying that the addresses assigned to the client are appropriate to the link to which the client is connected. A server sends a REPLY message to acknowledge receipt of a RELEASE or DECLINE message.
RELEASE (8)	A client sends a RELEASE message to the server that assigned addresses to the client to indicate that the client will no longer use one or more of the assigned addresses.
DECLINE (9)	A client sends a DECLINE message to a server to indicate that the client has determined that one or more addresses assigned by the server are already in use on the link to which the client is connected.
RECONFIGURE (10)	A server sends a RECONFIGURE message to a client to inform the client that the server has new or updated configuration parameters and that the client is to initiate a RENEW/REPLY or INFORMATION-REQUEST/REPLY transaction with the server to receive the updated information.
INFORMATION-REQUEST (11)	A client sends an INFORMATION-REQUEST message to a server to request configuration parameters without the assignment of any IPv6 addresses to the client.
RELAY-FORWARD (12)	A relay agent sends a RELAY-FORWARD message to transfer messages to servers, either directly or through another relay agent. The received message, either a client message or a RELAY-FORWARD message from another relay agent, is encapsulated in an option in the RELAY-FORWARD message.

DHCPv6 Message Type	Description
RELAY-REPLY (13)	A server sends a REPLAY-REPLY message to a relay agent containing a message that the relay agent delivers to a client. The REPLAY-REPLY message can be passed on by other relay agents for delivery to the destination relay agent. The server encapsulates the client message as an option in the REPLAY-REPLY message, which the relay agent extracts and sends to the client.

DHCPv6 Communications

Figure 8-9 illustrates the steps of the DHCPv6 communication process, which involves the client, the server, and the router. Remember that an important difference between DHCPv6 and DHCPv4 is that the router determines how the addresses will be allocated dynamically. A client is configured to automatically obtain its addressing and other configuration information.

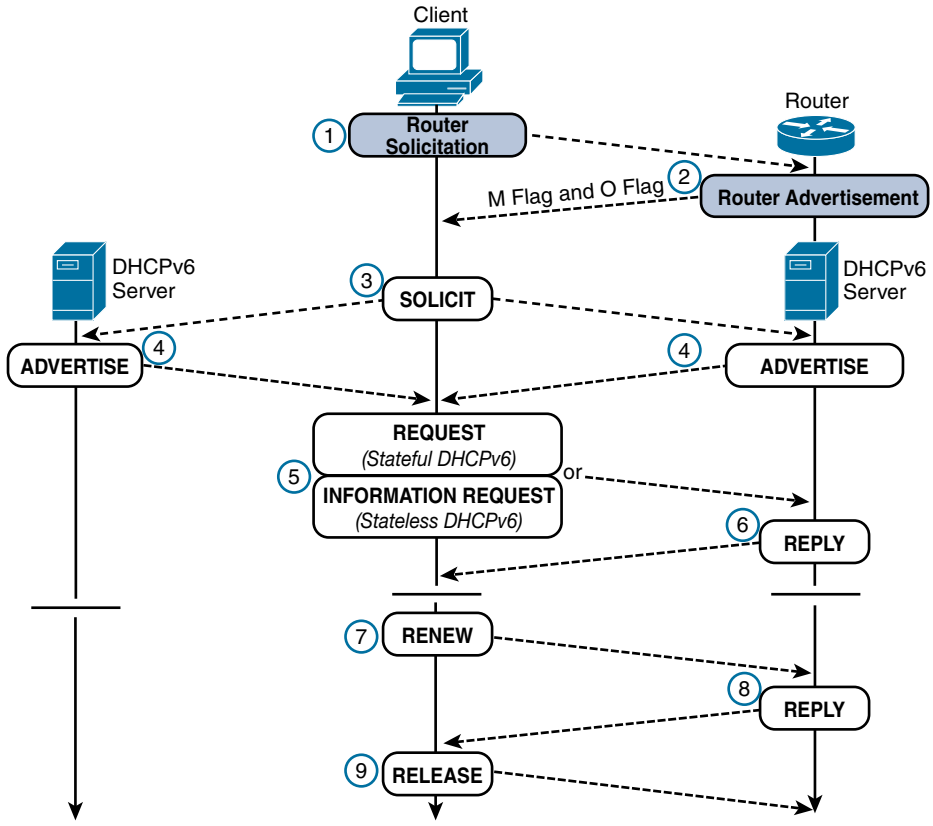


Figure 8-9 *Stateless and Stateful DHCPv6 Operations*

The host does not necessarily determine the method it uses to dynamically obtain its addressing information. If the host is depending on the Router Advertisement, the router controls this decision using the information contained in its RA message, specifically the A flag (Address Autoconfiguration flag), the M flag (Managed Address Configuration flag), and the O flag (Other Configuration flag). Depending on the operating system, a host can ignore the suggestion in the router's Router Advertisement and use any of these methods. As noted previously, it is a good idea to verify the behavior of the host operating system before deployment into the network

As shown in Figure 8-9, the DHCPv6 communications process is as follows:

Router Solicitation and Router Advertisement Messages

Step 1. If the client has not already received a Router Advertisement, it sends a Router Solicitation message to the all-routers multicast group, ff02::2. A host sends a Router Solicitation message when it has been configured to dynamically receive its prefix, prefix length, default gateway, and other information.

Step 2. A router periodically sends a Router Advertisement message in response to a Router Solicitation from a host. Two flags contained in the RA decide which method a host will use to receive its addressing and other configuration information: the M flag (Managed Address Configuration flag) and the O flag (Other Configuration flag). (The A flag is specific to SLAAC.)

If the M and/or O flag is set to 1, the DHCPv6 process continues, meaning that stateless or stateful DHCPv6 is being used. Step 5 illustrates how the client sends the appropriate request to the server, depending on whether stateful or stateless DHCPv6 is being used.

The M and/or O flag is set to 1 in the Router Advertisement message.

DHCPv6 Messages

Step 3. The host, now a DHCPv6 client, sends a SOLICIT message to the destination All_DHCP_Relay_Agents_and_Servers address, ff02::1:2, to locate a DHCPv6 server. In some cases, the client can be configured to use a unicast address to reach a specific server.

Step 4. One or more DHCPv6 servers respond with an ADVERTISE message to let the client know that it is available for DHCPv6 service. If the client receives more than one ADVERTISE message, it uses a decision process to choose the appropriate server, as defined in RFC 3315. These options are configured on both the client and the server.

Note The criteria for selecting a DHCPv6 server when a client receives ADVERTISE messages from multiple servers is beyond the scope of this book. For more information, see RFC 3315, section 17.1.3, "Receipt of Advertise Messages."

Step 5. The client sends a REQUEST or INFORMATION-REQUEST message to the server, depending on whether stateful or stateless DHCPv6 is being used. The type of message sent by the client is dependent on the M flag contained in the router's Router Advertisement. There are two possibilities:

- If stateful DHCPv6 is being used (the Router Advertisement M flag is set to 1), the client sends a REQUEST to obtain an IPv6 global unicast address and other configuration parameters.
- If the client is using stateless DHCPv6 (the Router Advertisement M flag is set to 0, and the O flag is set to 1), the client sends an INFORMATION-REQUEST message requesting only the configuration parameters from the server and not a global unicast address.

Step 6. If the server received a REQUEST from the client, it sends a REPLY containing assigned addresses and other configuration parameters. In response to an INFORMATION-REQUEST, the server sends a REPLY with only the configuration parameters.

Renew and Release

Step 7. When the DHCPv6 lease is about to expire, a client sends a RENEW message to the server that originally provided the client's addresses and configuration information to extend the lifetime of the lease.

Step 8. The server sends a REPLY message to acknowledge to the client that it is renewing the lease for its addresses and configuration parameters.

Step 9. When the client no longer needs one or more of its assigned addresses, it sends a RELEASE message to inform the DHCPv6 server.

Note A server also sends a REPLY message to acknowledge receipt of a RELEASE or DECLINE message.

Note If a client determines that an allocated address is already in use, typically through Duplicate Address Detection (DAD), it sends a DECLINE message to the server. The address is suspended on the host. The server marks this address as used by an unknown host and assigns another address to a client.

Summary

This chapter provides an overview of the three methods for dynamic IPv6 address.

- **Method 1:** Stateless Address Autoconfiguration (SLAAC)
- **Method 2:** SLAAC and a stateless DHCPv6 server
- **Method 3:** Stateful DHCPv6 server

The method that a device uses to determine how to obtain its addressing information is contained in the router's Router Advertisement message, specifically using three flags:

- **Address Autoconfiguration flag (A flag):** When set to 1 (on), this flag tells the receiving host to use SLAAC to create its global unicast address.
- **Other Configuration flag (O flag):** When set to 1 (on), this flag informs the host to get other addressing information, other than its global unicast address, from a stateless DHCPv6 server.
- **Managed Address Configuration flag (M flag):** When set to 1 (on), this flag notifies a host to use a stateful DHCPv6 server for its global unicast address and all other addressing information.

DHCPv6 is similar to its IPv4 counterpart but also has some significant differences. There are two forms of DHCPv6:

- **Stateful DHCPv6 services:** Global unicast address and other configuration information is obtained from a stateful DHCPv6 server.
- **Stateless DHCPv6 services:** Addressing information other than a global unicast address is obtained from a stateless DHCPv6 server.

One significant difference between DHCPv6 and DHCPv4 is that a DHCPv6 server does not provide a default gateway address. That information can only be dynamically obtained directly from the source, the router's Router Advertisement message.

Four DHCPv6 message types are the primary messages used to communicate between a DHCPv6 client and server:

- **SOLICIT (1):** DHCPv6 **clients use the** SOLICIT message to locate servers.
- **ADVERTISE (2):** A server sends an ADVERTISE message in response to a client's SOLICIT message to indicate that it is available for DHCPv6 service.
- **REQUEST (3):** A client sends a REQUEST message to request configuration parameters, including IPv6 addresses, from a specific DHCPv6 server.
- **REPLY (7):** A server sends a REPLY message containing assigned addresses and configuration parameters in response to a client's REQUEST message.

Review Questions

1. What method or methods are available for dynamic address allocation in IPv4?

2. What method or methods are available for dynamic address allocation in IPv6?

3. How does a client decide which method to use for dynamic address allocation in IPv6?

4. What is the purpose of a Router Solicitation message? What are the IPv6 source and destination addresses of this message?

5. What is the purpose of a Router Advertisement message, and when is it sent? What are the IPv6 source and destination addresses of this message?

6. Match each function to its Router Advertisement flag:

   A flag

   O flag

   M flag

   A. When set to 1 (on), this flag suggests that a device get all its addressing information, including a global unicast address, from a stateful DHCPv6 server.

   B. When set to 1 (on), this flag suggests that a device use SLAAC to create a global unicast address.

   C. When set to 1 (on), this flag suggests to a device that other configuration information is available from a stateless DHCPv6 server.

7. What are the settings 0 (off) or 1 (on) of the A, O, and M flags for each of the three address allocation methods:

   A. Method 1: Stateless Address Autoconfiguration (SLAAC)

   B. Method 2: SLAAC and a stateless DHCPv6 server

   C. Method 3: Stateful DHCPv6 server

8. How do a DHCPv4 server and DHCPv6 server differ in providing a default gateway address?

9. What does a DHCPv6 server use to uniquely identify a DHCPv6 client and server?

10. What four message types does DHCPv6 use for communications between a client and a stateful DHCPv6 server? List them in the order in which they are sent and which device (client or server) sends each message.

References

RFCs

RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, R. Droms, Cisco Systems, www.ietf.org/rfc/rfc3315, July 2003.

RFC 3596, *DNS extensions to support IP Version 6*, S. Thompson, Cisco Systems, www.ietf.org/rfc/rfc3596, October 2003.

RFC 3736, *Stateless Dynamic Host Configuration Protocol (DHCP) service for IPv6*, R. Droms, Cisco Systems, www.ietf.org/rfc/rfc3736, April 2004.

RFC 3769, *Requirements for IPv6 Prefix Delegation*, S. Miyakawa, NTT Communications Corporation, www.ietf.org/rfc/rfc3769, June 2004.

RFC 4862, *IPv6 Stateless Address Autoconfiguration*, S. Thompson, Cisco Systems, www.ietf.org/rfc/rfc4862, September 2007.

Website

Cisco IOS IPv6 Implementation Guide, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-dhcp.html.

Chapter 9

Stateless Address Autoconfiguration (SLAAC)

In Chapter 8, "Basics of Dynamic Addressing in IPv6," we introduced three methods an ICMPv6 Router Advertisement uses for dynamically allocating IPv6 global unicast addresses:

- **Method 1:** Stateless Address Autoconfiguration (SLAAC)
- **Method 2:** SLAAC and a stateless DHCPv6 server
- **Method 3:** Stateful DHCPv6 server

In this chapter we will focus on the first method, Stateless Address Autoconfiguration (SLAAC), which includes the following topics:

- The ICMPv6 Router Advertisement message and its use of the Address Autoconfiguration flag (A flag)
- The two methods SLAAC uses for creating an Interface ID: EUI-64 and a random 64-bit value (privacy extension)
- The privacy extension with SLAAC and the use of both a public global unicast address and a temporary global unicast address
- Examination of the fields in the ICMPv6 Router Advertisement message using Wireshark
- The various states and lifetimes of a global unicast address created using SLAAC
- The different RA configuration options, including adding a list of recursive DNS server addresses in the RA
- The process an IPv6 device uses to select whether to use a link-local address or one of multiple GUA addresses

The RA Message and SLAAC

Figure 9-1 shows some of the basic information in a Router Advertisement message that suggests SLAAC for dynamic address allocation.

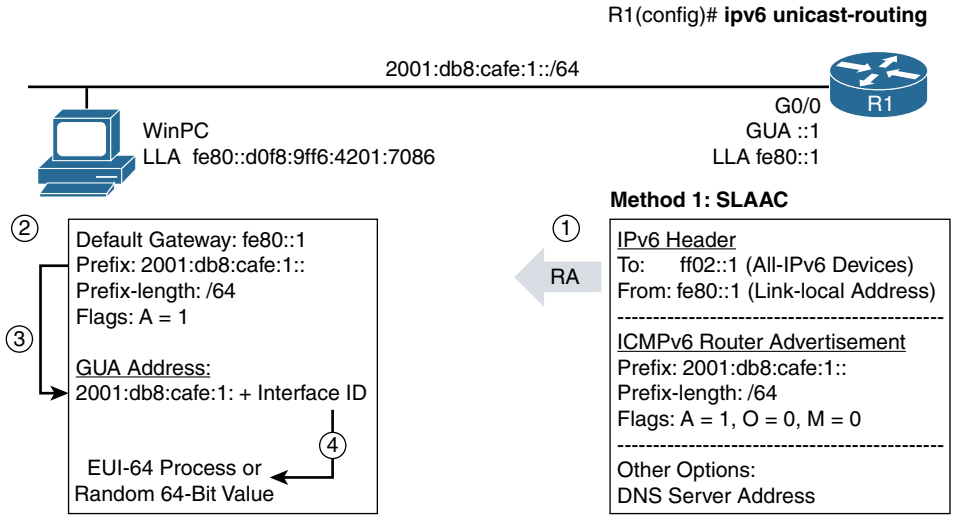


Figure 9-1 *Method 1: Router Advertisement and Basics of SLAAC*

As discussed several times already in this book, router R1 must be an *IPv6 router* to send out the Router Advertisement messages. This is done using the **ipv6 unicast-routing** global configuration command. A Cisco IPv6 router sends a Router Advertisement message periodically every 200 seconds by default and after receiving a Router Solicitation message. IPv6 Router Advertisements are automatically sent on Ethernet and FDDI interfaces if IPv6 unicast routing is enabled and an IPv6 address has been configured on the interface. (FDDI is almost obsolete.)

As discussed in Chapter 8, the RA message contains three flags to tell a device how to obtain or create its global unicast address:

- **Address Autoconfiguration flag (A flag):** When set to 1 (on), this flag tells the receiving host to use SLAAC to create its global unicast address.
- **Other Configuration flag (O flag):** When set to 1 (on), this flag tells the host to get other addressing information, other than its global unicast address, from a stateless DHCPv6 server.
- **Managed Address Configuration flag (M flag):** When set to 1 (on), this flag tells the host to use a stateful DHCPv6 server for its global unicast address and all other addressing information.

The four steps in Figure 9-1 summarize what you have already learned in previous chapters about Router Advertisement messages and SLAAC:

Step 1. The Router Advertisement message is encapsulated in an IPv6 header with a destination IPv6 address of ff02::1, the all-IPv6 devices multicast, and the source IPv6 address of R1's link-local address, fe80::1. (The RA may also be sent as a solicited unicast.) The link-local address was manually configured on R1, and it is the address the receiving devices can use for their default gateway address. The IPv6 Next Header field (not shown in Figure 9-1) has the value 58 decimal (3a hexadecimal) indicating that the encapsulated data is an ICMPv6 message.

Note As in ICMPv4, the Type field in ICMPv6 indicates that this is an ICMPv6 Router Advertisement. ICMPv6 is discussed in later chapters.

The RA message contains a lot of information, most of which is examined in detail throughout this chapter. For now, you can see that the RA message includes the prefix and prefix length of the link: 2001:db8:cafe:1:: and /64. These are actually options and are not required to be included in the RA, but Cisco IOS does include them by default.

Also by default, the A flag is set to 1, while the O and M flags are set to 0. The A flag set to 1 is a suggestion to the receiving devices to use the prefix in the RA message to generate a global unicast address (SLAAC). Because the O and M flags are set to 0, the suggestion to the receiving device is that no other information is available from a DHCPv6 server.

The RA message may also contain an option for DNS recursive server addresses used for the DNS name resolution in IPv6 hosts. (You will examine the details of this option later in this chapter.)

Note When the O and M flags are set to 0, there is no need for the receiving device to use the services of DHCPv6. This means that all the necessary addressing and link information must be contained in the RA message or manually configured on the device itself.

Step 2. WinPC receives the RA message and uses the source IPv6 address of the RA, fe80::1, as its default gateway address. The A flag is set to 1, which indicates that WinPC should use the prefix to create a global unicast address (SLAAC).

Step 3. WinPC uses the prefix 2001:db8:cafe:1: in the RA message as the prefix for its global unicast address.

Step 4. A host generates an Interface ID for the GUA address using either EUI-64 process or a random 64-bit value. Because WinPC is a Windows host, it uses a randomized Interface ID. WinPC then performs Duplicate Address Detection (DAD) on the GUA address to make sure no other devices on the link are using this address.

Note DAD is required to be performed for all unicast addresses (such as global unicast addresses and link-local unicast addresses) before the addresses are assigned to interfaces, regardless of whether they were obtained through SLAAC, DHCPv6, or manual configuration. There are some exceptions to this behavior, as discussed in in RFC 4429, *Optimistic Duplicate Address Detection (DAD) for IPv6*.

Using the **show ipv6 interface gigabitethernet 0/0** command in Example 9-1, you can verify the following information that R1 sends in its Router Advertisement messages:

Note Later in this chapter, you will see how to use Wireshark to examine all the information in R1's Router Advertisement message along with the Router Solicitation message from WinPC.

- **FF02::2:** Router R1 was configured previously as an IPv6 router using the **ipv6 unicast-routing** command, which results in R1 joining the all-IPv6 routers multicast group, ff02::2.
- **MTU is 1500 bytes:** This informs hosts of the maximum transmission unit (MTU) for the link. Hosts use this information to maximize the size of the IPv6 packet.
- **ND router advertisements are sent every 200 seconds:** This shows how often periodic RA messages are sent on this interface. The default is 200 seconds.
- **ND router advertisements live for 1800 seconds:** This is the Router Lifetime information sent in RA messages. This informs a host of the duration, in seconds, that the router should be used as the default gateway. A lifetime of 0 indicates that the router is not a default gateway. The Router Lifetime applies only to the router's function as a default gateway. It does not apply to other information contained in other message fields or options such as prefix and prefix length. The host refreshes its own timer every time it receives a Router Advertisement. The default is 1800 seconds.
- **ND advertised default router preference is Medium:** This is the value of the Router Preference sent in the RA messages. Hosts dynamically populate their Default Router List based on the source IPv6 addresses of the RA messages they receive. The Default Router Preference (default gateway) can be one of three states: high, medium (default), or low. This helps the host determine which router to use as the default gateway when it receives multiple RA messages. The default is medium.
- **Hosts use stateless autoconfig for addresses:** This indicates that the RA message sent on this interface is suggesting that hosts obtain their dynamic IPv6 addressing using

SLAAC, as a result of the A flag being set to 1. Because the O and M flags are set to 0, there is no mention of suggesting the use of a DHCPv6 server. In Chapters 10, "Stateless DHCPv6," and 11, "Stateful DHCPv6," you will see the effect of setting the O and M flags to 1.

Note Many Router Advertisement parameters such as the ones listed here are configurable, as discussed later in this chapter.

Note A device that is not a router maintains a *Default Router List*. When a device receives a Router Advertisement, it adds the link-local source address of the packet as one of the routers it can use as a default gateway. Each entry has an invalidation timer, the Router Lifetime, extracted from the Router Advertisement used to delete entries that are no longer being advertised.

Example 9-1 *Verification of Router R1 as an IPv6 Router*

```
R1(config)# ipv6 unicast-routing
R1(config)# exit
R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::1
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::FB
    FF02::1:FF00:1
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 30000)
  ND advertised reachable time is 0 (unspecified)
  ND advertised retransmit interval is 0 (unspecified)
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  ND advertised default router preference is Medium
  Hosts use stateless autoconfig for addresses.
R1#
```

If R1's G0/0 interface has been configured with an IPv6 address but has not been configured as an IPv6 router, it does not send RA messages. Example 9-2 shows what happens when the **ipv6 unicast-routing** command is *not* configured. Comparing the **show ipv6 interface gigabitethernet 0/0** command to the same command in Example 9-1, the output no longer shows the router belonging to the ff02::2 (the all-IPv6 routers multicast group) and any statements beginning with "ND advertised" or "ND router advertisements." The last statement, "Hosts use stateless autoconfig for addresses," has also been omitted.

Example 9-2 *Verification That Router R1 Is Not an IPv6 Router*

```
< The ipv6 unicast-routing command has not been configured>

R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::1
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::D
    FF02::16
    FF02::FB
    FF02::1:FF00:1
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 30000)
  ND NS retransmit interval is 1000 milliseconds
R1#
```

You can display the sending of Router Advertisement messages by using the **debug ipv6 nd** command, shown in Example 9-3. This command displays all Neighbor Discovery Protocol messages sent and received on the interface. This example includes the **ipv6 unicast-routing** command to first re-enable R1 as an IPv6 router.

Example 9-3 *Examining R1's RA Messages Using the* **debug ipv6 nd** *Command*

```
R1(config)# ipv6 unicast-routing
R1(config)# exit
R1# debug ipv6 nd
  ICMP Neighbor Discovery events debugging is on
R1#
*Nov 27 18:34:52.494: ICMPv6-ND: (GigabitEthernet0/0,FE80::1) send RA to FF02::1
*Nov 27 18:34:52.494: ICMPv6-ND: (GigabitEthernet0/0,FE80::1) Sending RA (1800) to
  FF02::1
*Nov 27 18:34:52.494: ICMPv6-ND:    MTU = 1500
*Nov 27 18:34:52.494: ICMPv6-ND:    prefix 2001:DB8:CAFE:1::/64 [LA] 2592000/604800
<output omitted for brevity>
R1# undebug all
```

The output of the **debug ipv6 nd** command may vary depending on the IOS version. Here's what you can see in the Example 9-3 output:

- **(GigabitEthernet0/0,FE80::1) Sending RA (1800) to FF02::1**
  - **Gigabit/Ethernet0/0:** This is the egress interface of the RA message.
  - **FE80::1:** This is the source IPv6 address of the RA message and the address that hosts can use to populate their Default Router List.
  - **Sending RA:** This indicates that the information on this line and the indented lines that follow are from a Router Advertisement message sent by this router.
  - **(1800):** This is the Router Lifetime, the duration (in seconds) that the router should be used as the default gateway. (1800 seconds equals 30 minutes.)
  - **to FF02::1:** ff02::1 is the all-IPv6-devices multicast address, the destination IPv6 address of the Router Advertisement.
  - **MTU = 1500:** This is the maximum transmission unit (MTU) for the link.
- **prefix 2001:DB8:CAFE:1::/64 [LA] 2592000/604800**
  - **prefix 2001:DB8:CAFE:1::/64:** These are the prefix and prefix length that devices can use to create a global unicast address using SLAAC.
  - **[LA]:** The L flag (On-Link flag) and A flag (Address Autoconfiguration flag) are both set to 1. When set to 1, the On-Link flag indicates that the prefix sent in the RA is on this link or subnet. The A flag indicates to devices that the prefix can be used to create an address with SLAAC. The L flag and On-Link flag are discussed in the next section.
  - **2592000/604800:** These are the Valid Lifetime and Preferred Lifetime, in seconds. The Valid Lifetime is the amount of time (in seconds) that a device should consider the prefix in the Router Advertisement as *valid*, which means it can be used as a source IPv6 address. (2,592,000 seconds equals 30 days.)

The Preferred Lifetime is the amount of time (in seconds) that a device should consider addresses generated from the RA's prefix and SLAAC as *preferred*. (604,800 seconds equals 7 days.) A preferred address means the device can initiate a new connection using this address as the source address. After the Preferred Lifetime expires, a device can continue to use the address for existing connections but can no longer create new connections using this address. The Valid and Preferred Lifetimes are discussed later in this chapter.

The **ipconfig** command in Example 9-4 shows the ensuing address information for WinPC. Notice that WinPC has two global unicast addresses. The first address, 2001:db8:cafe:1:d0f8:9ff6:4201:7086, is known as a *public address*. The second address, 2001:db8:cafe:1:78bd:10b0:aa92:62c, is a *temporary address*. The temporary address is added because Windows OS implements the privacy extension for SLAAC. Also, notice that both GUA addresses and the link-local address use a random 64-bit value to create the Interface ID. This is also a result of Windows using the privacy extension option. The privacy extension is used to help provide anonymity and privacy and is discussed in more detail later in this chapter.

Note RFC 4941, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, uses the terms *public address* and *temporary address* for the two types of global unicast addresses created using SLAAC. The term *public address* shouldn't be confused with the same term in IPv4. A public IPv4 address is one that is globally routable in the Internet, whereas a private IPv4 addresses is not. Both IPv6 public and temporary global unicast addresses are globally routable addresses in the Internet. (Unique local addresses are not globally routable.)

Example 9-4 *WinPC's Addressing Information Using SLAAC*

```
WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086
   Temporary IPv6 Address. . . . . . : 2001:db8:cafe:1:78bd:10b0:aa92:62c
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
   Autoconfiguration IPv4 Address  . : 169.254.112.134
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : fe80::1%11
```

On-Link Determination

When a host wants to send an IP packet to another host, how does it know whether to send that packet directly to the other host (if the destination is on the same subnet) or if it needs to send the packet to the default gateway?

In the IPv4 world, a host does a simple logical AND operation, using its IPv4 address and its subnet mask to determine its local network address. To determine whether the

destination is on its same network, it does a similar AND operation using the destination IPv4 address and its own subnet mask. If the network addresses match, then the two addresses are on the same subnet. If they differ, then the destination IPv4 address is on a different subnet, and the packet must be sent to the default gateway.

The host uses its own subnet mask for both AND operations. It doesn't know the subnet mask of the destination, but it doesn't matter. If the destination address is on its subnet, then it is using the same mask as the source. Otherwise, unexpected results can occur.

The IPv6 world is different. In IPv6, the host determines its local subnet, known as the *on-link prefix*, using two fields in the Router Advertisement, as shown in Example 9-3:

- Prefix
- On-Link flag (L flag)

When a host receives a Router Advertisement with a prefix and the L flag set to 1 (default), it adds this prefix to its Prefix List, a list of on-link prefixes. Any of the host's addresses that use this prefix (SLAAC generated, manually configured, or DHCPv6) will be considered on-link to this prefix, on this subnet.

For example, when the A flag is set to 1 (default), the host can create a GUA address using this prefix, and the prefix is considered on-link. The host can send any packets with this prefix in the destination IPv6 address directly to the device.

In Example 9-3, the RA includes the prefix 2001:db8:cafe:1::/64 with the L flag set to 1. The **netsh interface ipv6 show siteprefixes** command in Example 9-5 shows this prefix added to WinPC's Prefix List.

Example 9-5 *WinPC's Prefix List*

```
WinPC> netsh interface ipv6 show siteprefixes
Prefix                  Lifetime      Interface
----------------------  ------------  ----------------------
2001:db8:cafe:1::/64    7d23h59m56s   Local Area Connection
```

On-link means that a packet can be sent directly to a device without being forwarded through a router. According to RFC 4861, *Neighbor Discovery for IPv6*, a device considers an address to be on-link if one of the following conditions is present:

- A Router Advertisement message includes this prefix with the On-Link flag set to 1.
- A local router indicates that this address is on-link in a Redirect message. When a router forwards a packet out the same interface it was received on, the router sends a redirect message to the source of the packet. The source then considers this address as on-link and forwards subsequent packets directly to the device.
- An ICMPv6 Neighbor Advertisement message is received *for* the target address. (A Neighbor Advertisement is similar to an ARP Reply in IPv4.)
- Any ICMPv6 Neighbor Discovery message is received *from* this device.

The following are some of the characteristics of on-link determination:

- By default, a host treats only the prefix of its link-local address as on-link. A link-local address is permanently on-link.
- The prefix is considered on-link for the period specified by the Valid Lifetime. The Valid Lifetime is reset each time a new RA is received with this same prefix and the L flag set to 1.
- The prefix of an IPv6 address assigned to an interface using manual configuration, SLAAC, or DHCPv6 is *not* implicitly considered on-link. The host can only consider the prefix on-link if it can be explicitly determined.
- A destination is assumed to be off-link unless there is explicit information indicating that it is on-link.
- A host can have an IPv6 address that isn't related to any *on-link prefix*—in other words, doesn't belong to any subnet. It can also have an on-link prefix that is not associated with any of its addresses.

Packets sent to any addresses that are not on-link are sent to the default router (default gateway). If there is no default router, then the device should be indicated by an ICMPv6 Destination Unreachable message.

Note For more information see RFC 5942, *IPv6 Subnet Model: The Relationship Between Links and Subnet Prefixes*, and RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*.

Generating an Interface ID

SLAAC, as the name implies, is stateless. The host creates its own global unicast address, without the services of a stateful device such as a stateful DHCPv6 server. It does this by combining the prefix in the RA message with a self-generated Interface ID. The method the host uses to create the Interface ID depends on the operating system. There are two options for creating the Interface ID:

- EUI-64 process
- Random 64-bit value (privacy extension)

Note Cryptographically Generated Addresses (CGA) is a third option and is beyond the scope of this book. The Mac OS examples in this book use Mac OS 10.11. Mac OS 10.12 (Sierra) now uses Cryptographically Generated Addresses (CGA), RFC 3972, to generate the Interface ID. To disable CGA add the command **net.inet6.send.opmode=0** to the /etc/sysctl.conf file and reboot. For more information about CGA, I recommend the book IPv6 Security, by Eric Vynke.

The EUI-64 process uses the Ethernet MAC address to generate the Interface ID. Mac OSX and some Linux implementations use EUI-64 to create the Interface ID for the public address. The concern many have is related to the traceability of an address that uses an Ethernet MAC address. Another option is to use a randomized 64-bit value for the Interface ID, part of the privacy extension for SLAAC. The privacy extension also includes the use of temporary addresses. Both of these methods are examined in this section.

Note RFC 4862, *IPv6 Stateless Address Autoconfiguration*, doesn't specifically state that EUI-64 should be used for SLAAC. The use of the modified EUI-64 format for Interface IDs is discussed in RFC 4291, *IPv6 Version 6 Addressing Architecture*. Randomized Interface IDs are discussed in RFC 4941, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. An alternative method for generating Interface IDs based on MAC addresses without sacrificing security and privacy of users is discussed in RFC 7217, *A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)*.

Generating the Interface ID Using the EUI-64 Process

Figure 9-2 shows the topology used in this chapter. The link-local address for each router is displayed below the Interface ID of the router's global unicast address.

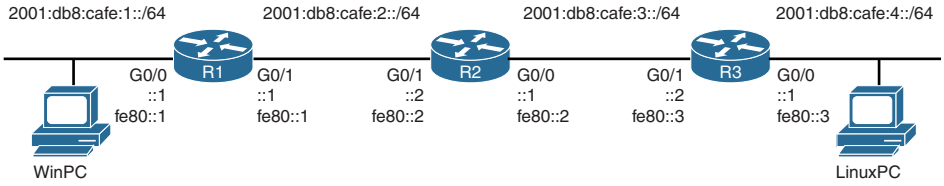


Figure 9-2 *Topology for Generating Interface IDs*

Example 9-6 shows the output from the **ifconfig** command on the LinuxPC, which is running Ubuntu Linux. The LinuxPC is enabled to obtain its IPv6 addressing dynamically. Router R3 has been configured with the **ipv6 unicast-routing** global configuration command and is sending Router Advertisements messages using the defaults. These defaults include an A flag with the value 1, and the O and M flags with the value 0.

Example 9-6 *LinuxPC's Addressing Information Using SLAAC and EUI-64*

```
LinuxPC$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:af:25:24
          inet6 addr: 2001:db8:cafe:4:250:56ff:feaf:2524/64 Scope:Global
          inet6 addr: fe80::250:56ff:feaf:2524/64 Scope:Link
          inet6 addr: 2001:db8:cafe:4:314a:dd3e:762f:e140/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:3415 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3438 (3.4 KB)  TX bytes:459921 (459.9 KB)
          Interrupt:16
<output omitted for brevity>
```

Notice that the first global unicast address (the public address) and the link-local address use EUI-64. The ff:fe in the middle of the Interface ID is a good indication that SLAAC with EUI-64 was most likely used. The second global unicast address, 2001:db8:cafe:4:314a:dd3e:762f:e140, uses a randomized Interface ID. This is the temporary address.

The 48-bit Ethernet MAC address for LinuxPC is shown in Example 9-6. As illustrated in Figure 9-3, an Ethernet 48-bit MAC address is a combination of a 24-bit Organizationally Unique Identifier (OUI) and a 24-bit device identifier, written in hexadecimal. Manufacturers of Ethernet network interface cards (NICs) have one or more OUIs or vendor codes. Each 24-bit OUI has a 24-bit device identifier that uniquely identifies the Ethernet NIC. In other words, appending a 24-bit device identifier to a 24-bit OUI uniquely identifies an Ethernet NIC. (There is also an IEEE standard for a 64-bit MAC address.)

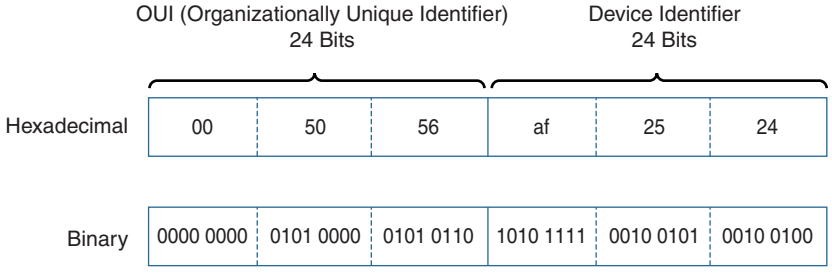


Figure 9-3 *LinuxPC's Ethernet MAC Address*

Note For a list of IEEE OUI codes, see *IEEE Guidelines for Use Organizationally Unique Identifier (OUI) and Company ID (CID)* at http://standards.ieee.org/develop/regauth/oui/oui.txt.

The modified EUI-64 is a concatenation of the 24-bit OUI with the U/L OUI bit *flipped*, a 16-bit value of fffe, and the 24-bit device identifier. This process is accomplished in three simple steps, as illustrated in Figure 9-4:

Note This is the same EUI-64 process you saw in Chapters 5, "Global Unicast Address," and 6, "Link-Local Address."

Step 1. After converting the MAC address to binary, the MAC address is split in the middle, with the 24-bit OUI on the left and the 24-bit device identifier on the right.

Step 2. fffe is inserted between the OUI and the device identifier. The binary equivalent of fffe is 1111 1111 1111 1110. fffe is an IEEE-reserved value which indicates that the EUI-64 address was generated from a 48-bit MAC address.

Step 3. The Universally/Locally (U/L) bit, also known as the Local/Global bit, is the seventh bit of the first byte and is used to determine whether the address is universally or locally administered. If this bit is 0, IEEE, through the designation of a unique company ID, has administered the address. If the U/L bit is a 1, the address is locally administered. The network administrator has overridden the manufactured address and specified a different address.

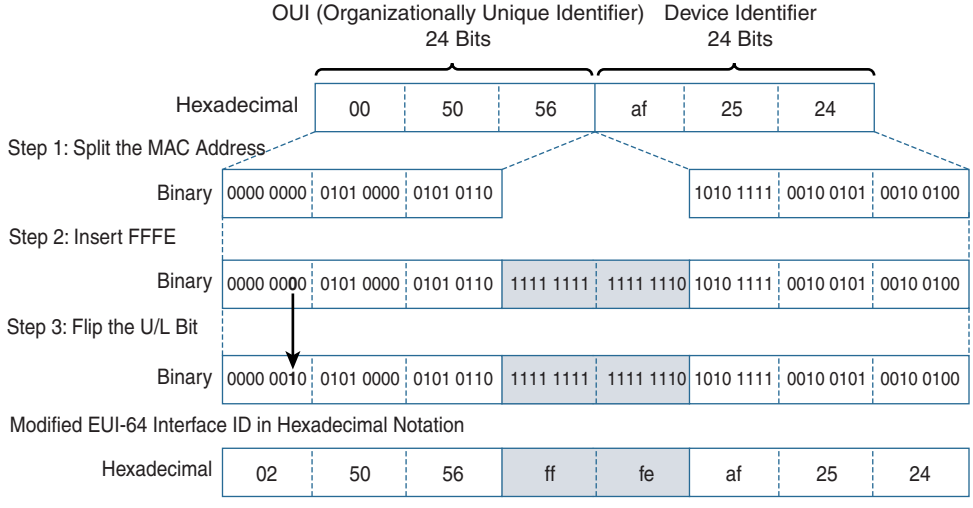


Figure 9-4 *Modified EUI-64 Format on LinuxPC*

There seems to be some conflict as to whether this bit should be flipped from 0 to 1 or 1 to 0. Some documentation states that the U/L bit should be modified only if it is a 0, in which case it should be flipped to 1. However, it seems that Cisco devices flip this bit regardless of its value. As shown in Figure 9-4, flipping the U/L bit modifies the second hexadecimal value in the Interface ID.

Note RFC 7042, *IANA Considerations and IETF Protocol Usage for IEEE 802 Parameters*, discusses the logic behind flipping the U/L bit. The decision was to invert the U/L bit to make it easier for network operators to type in local-scope identifiers.

Using the EUI-64 process, the 48-bit MAC address was used to create a 64-bit IPv6 Interface ID. The MAC address 00-50-56-af-25-24 was translated into the EUI-64 format 00-50-56-ff-fe-af-25-24. EUI-64 inserted fffe into the middle of the address, and the modification of the seventh bit altered the second hexadecimal value from 0 to 2.

Note The IPv6 Interface ID is 64 bits, which accommodates longer 64-bit MAC addresses and also falls nicely on a 64-bit boundary for performance efficiency. The modified EUI-64 format is used when the interface has a 48-bit MAC address, which is the current standard. Additional information can be found in the *IEEE Guidelines for Use Organizationally Unique Identifier (OUI) and Company ID (CID)*, *IEEE 1394-1995: IEEE Standard for a High Performance Serial Bus* (also known as FireWire), and *IEEE 802.15: Low-Rate Wireless Personal Area Networks (LR-WPANs)* (also known as ZigBee).

To summarize, LinuxPC received its prefix 2001:db8:cafe:4: from router R1's RA message. The LinuxPC's operating system uses EUI-64 to create a global unicast address. The EUI-64 process uses the 48-bit Ethernet MAC address for the Interface ID, inserts fffe in the middle, and flips the seventh bit. The RA-supplied prefix (2001:db8:cafe:4:) is prepended to the EUI-64-generated Interface ID (0250:56ff:feaf:2524) to form the global unicast address 2001:db8:cafe:4:0250:56ff:feaf:2524.

Configuring a Windows Host to Use EUI-64

The **ipconfig /all** command in Example 9-7 shows that WinPC is using a random 64-bit value to create an Interface ID for both its global unicast addresses and its link-local address. You saw this earlier, with the **ipconfig** command Example 9-5. Notice that the Ethernet MAC address, the physical address, was not used to generate the Interface ID.

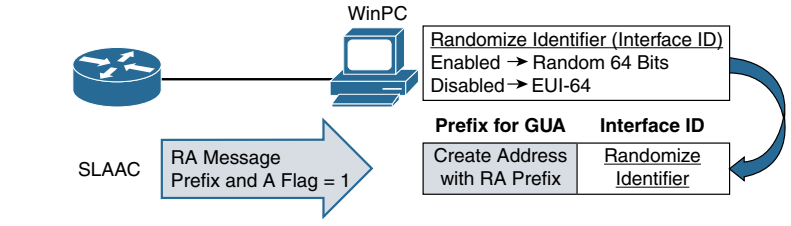
Example 9-7 *WinPC's Addressing Information Using SLAAC and a Random 64-Bit Interface ID*

```
WinPC> ipconfig /all
<output omitted for brevity>

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Network Connection

   Physical Address. . . . . . . . . : 00-50-56-AF-97-68

   DHCP Enabled. . . . . . . . . . . : Yes

   Autoconfiguration Enabled . . . . : Yes

   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086(Preferred)
   Temporary IPv6 Address. . . . . . : 2001:db8:cafe:1:78bd:10b0:aa92:62c (Preferred)
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11(Preferred)

   Default Gateway . . . . . . . . . : fe80::1%11
```

A Windows host can be configured to use EUI-64 for the public GUA address instead of the random 64-bit Interface ID. To change the default behavior of a Windows host to use EUI-64, you disable the randomize identifier:

```
C:\> netsh interface ipv6 set global randomizeidentifiers=disabled store=active
C:\> netsh interface ipv6 set global randomizeidentifiers=disabled
store=persistent
```

Figure 9-5 illustrates the use of these commands on WinPC.




```
WinPC> netsh interface ipv6 set global randomizeidentifiers=enabled store=active
Ok.

WinPC> netsh interface ipv6 set global randomizeidentifiers=enabled store=persistent
Ok.
```

Figure 9-5 *Modifying the Windows Randomize Identifier for EUI-64*

Disabling the randomize identifier parameter forces Windows to use EUI-64 for the Interface ID, as shown in Example 9-8. Notice that this command affects how the Interface ID was generated for both its public address and its link-local address. Both of these Interface IDs no longer use the random 64-bit value d0f8:9ff6:4201:7086 but now have the EUI-64-generated value 250:56ff:feaf:9768.

Example 9-8 *WinPC's Addressing Information Using SLAAC and Privacy Extension*

```
WinPC> netsh interface ipv6 set global randomizeidentifiers=disabled store=active
Ok.

WinPC> netsh interface ipv6 set global randomizeidentifiers=disabled
  store=persistent
Ok.

WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:250:56ff:feaf:9768
   Temporary IPv6 Address. . . . . . : 2001:db8:cafe:1:78bd:10b0:aa92:62c
```

```
   Link-local IPv6 Address . . . . . : fe80::250:56ff:feaf:9768%11
    Autoconfiguration IPv4 Address . : 169.254.112.134
    Subnet Mask . . . . . . . . . . . : 255.255.0.0
    Default Gateway . . . . . . . . . : fe80::1%11

<output omitted for brevity>
```

However, this **netsh** command has no effect on WinPC's temporary IPv6 address. The temporary addresses, created through the use of the privacy extension for SLAAC, only use randomly generated values for the Interface ID.

Privacy Extension for Stateless Address Autoconfiguration

Devices using Stateless Address Autoconfiguration to create a global unicast use a combination of a prefix advertised by the router's RA message and locally available information. (This also applies to the link-local address without the use of the prefix.) When the device uses the EUI-64 process, the Interface ID is generated using the MAC address. Using an Ethernet MAC address makes it easy to associate the IPv6 address to the actual device.

However, some have concerns about using an Interface ID that can be associated directly to a physical device (or Ethernet NIC)—an address that never changes. Any time a fixed identifier (address) is used in multiple sessions and with various applications, it becomes possible to correlate the same address to seemingly unrelated activity. RFC 4941, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, offers the following example:

> *For example, a network sniffer placed strategically on a link across which all traffic to/from a particular host crosses could keep track of which destinations a node communicated with and at what times. Such information can in some cases be used to infer things, such as what hours an employee was active, when someone is at home, and so on. Periodically changing the Interface ID of an address makes it more difficult for eavesdroppers and other information collectors, such as websites and mobile apps, to associate these different addresses and transactions to a particular device.*

RFC 4941, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, addresses these concerns:

- **Generation of randomized Interface IDs:** This is a mechanism for creating an Interface ID that is not traceable to a physical device. It can be used with both a public address (the initial address created by SLAAC) and a temporary address to help provide privacy. A temporary address must use a randomized Interface ID.

- **Generation of temporary addresses:** This provides additional addresses that have relatively short lifetimes and are used as the source address when originating connections. These addresses have the same prefix as a public address and use only a randomized value for the Interface ID. (The public address is typically used by other devices as the destination IPv6 address when they are initiating the connection.)

In other words, a device implementing the SLAAC privacy extensions means the following:

- The public address can use a randomized Interface ID instead of EUI-64. (Public addresses can also use EUI-64.)
- Temporary addresses can be generated and use only a randomized Interface ID. These addresses are in addition to the public address.

A global unicast address generated using EUI-64 is easily traceable because the Interface ID contains the Ethernet MAC address. Because MAC addresses are globally unique (or at least should be), it is possible to track the activity of the device even when it has different prefixes as it attaches to various networks. This concern was the motivation behind the privacy extension option for SLAAC.

Privacy Extension and Generating Randomized Interface IDs

A randomized Interface ID helps provide privacy by eliminating the tracking of packets to a device's Interface ID using a unique Ethernet MAC address. By default Windows generates randomized Interface IDs for its link-local address and GUA addresses created with SLAAC.

Example 9-8 shows Windows configured to use EUI-64 for generating the Interface ID on the public GUA address and link-local address instead of the default privacy extension. These commands re-enable the default behavior of a Windows host to use the randomized identifiers for SLAAC addresses:

```
C:\> netsh interface ipv6 set global randomizeidentifiers=enabled store=active
C:\> netsh interface ipv6 set global randomizeidentifiers=enabled store=persistent
```

Note Windows uses the privacy extension by default. The **netsh** commands here need to be configured only if the randomize identifier was previously disabled, forcing Windows to use EUI-64.

The default behavior of WinPC is to use a randomly generated Interface ID for its SLAAC-generated global unicast addresses and its link-local address, as shown in Example 9-9.

Example 9-9 *WinPC's Addressing Information Using SLAAC and Privacy Extension*

```
WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086
   Temporary IPv6 Address. . . . . . : 2001:db8:cafe:1:78bd:10b0:aa92:62c
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
   Autoconfiguration IPv4 Address  . : 169.254.112.134
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : fe80::1%11

<output omitted for brevity>
```

Privacy Extension and Temporary Addresses

Another aspect of the privacy extension for SLAAC, which adds a bit more complexity to understanding IPv6, is the use of temporary IPv6 addresses. Since Windows Vista, Microsoft has enabled the privacy extension on its Windows operating system by default.

Figure 9-6 illustrates both uses of the privacy extension for Windows. The previous section discusses the use of the randomized identifier. Windows uses two parameters for the privacy extension to SLAAC:

- **Randomize identifiers parameter:** Enables/disables the randomization of the Interface ID. When disabled, the Interface ID is generated using EUI-64. Enabled by default.
- **Privacy parameter:** Enables/disables the use of temporary addresses. Enabled by default.

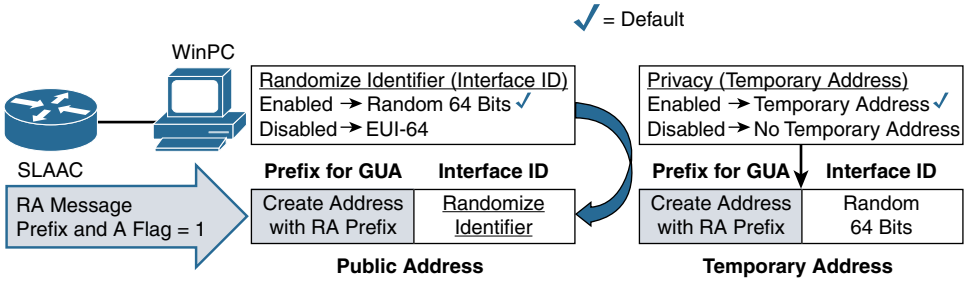


Figure 9-6 *Windows Privacy Extensions*

Temporary addresses have Interface IDs that are created only by using a randomly generated 64-bit value. They are used as the source IPv6 address when the device originates the connection. Temporary addresses have a short lifetime, usually hours or days. It is common to have multiple temporary addresses to make sure existing connections can continue while a new temporary address is created for new connections. The lifetime of temporary addresses is covered later in this chapter.

Note A device may have multiple global unicast addresses and also a link-local address. The GUA addresses can be on the same network or on different networks. Later in this chapter you will see how a device selects which address to use as a source address, using the default address selection process.

Disabling the Use of Temporary Addresses

The first **ipconfig** command in Example 9-10 shows that WinPC has both a public global unicast address and a temporary global unicast address. Two **netsh interface ipv6 set privacy=disabled** commands are used to disable the use of temporary addresses. The second **ipconfig** command shows that WinPC no longer has a temporary address and only a public GUA address. (This may require disabling and re-enabling the Ethernet interface.)

Example 9-10 *Verifying and Disabling the Privacy Extension*

```
WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086
   Temporary IPv6 Address. . . . . . : 2001:db8:cafe:1:78bd:10b0:aa92:62c
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
   Autoconfiguration IPv4 Address  . : 169.254.112.134
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : fe80::1%11

<output omitted for brevity>

! Disable the use of temporary addresses

WinPC> netsh interface ipv6 set privacy=disabled store=active
Ok.

WinPC> netsh interface ipv6 set privacy=disabled store=persistent
Ok.
```

```
! WinPC no longer uses temporary addresses

WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
   Autoconfiguration IPv4 Address  . : 169.254.112.134
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : fe80::1%11

<output omitted for brevity>
```

Use these commands to re-enable the default behavior of a Windows host to use temporary addresses when using SLAAC:

```
C:\> netsh interface ipv6 set privacy state=enabled store=active
C:\> netsh interface ipv6 set privacy state=enabled store=persistent
```

Note The use of privacy extensions with Linux and Mac OS (randomized identifier and temporary addresses) may vary. Example 9-6 shows that Ubuntu Linux (LinuxPC) uses EUI-64 to create the public GUA address, and a temporary address is generated with a randomized identifier. I have seen similar behavior on a MacBook Pro running Mac OS X. It is a good idea to always verify the behavior of the operating system and consult proper documentation as actual implementations may vary. Generally speaking, to enable the privacy extensions on Linux OS, you use the **sysctl net.ip6.conf.if.use_tempaddr=2** command, and for Mac OS, you use the **sysctl net.inet6.ip6.use_tempaddr=1** command.

Why does IPv6 use multiple addresses? In DHCP for IPv4, a client typically receives an address and periodically wants to renew that address with the server. With DHCPv4 you typically want a device to keep the same IPv4 address—or at least don't have any issue with it doing so. The IPv4 address is most likely a private IPv4 address and NAT will hide the address, but only to the outside world.

However, with IPv6 and SLAAC, for privacy reasons, you want to use an address only for few hours or days. As mentioned earlier, temporary addresses have a shorter lifetime than public addresses and are used when the device initiates a connection. Multiple temporary addresses are used to ensure that existing connections can continue while a new temporary address is created for any new connections.

Autoconfigured Address States and Lifetimes

As you have seen, a router's RA message contains information to assist a device in creating a public IPv6 address and, if the receiving device is enabled for the privacy extension, a temporary IPv6 address.

An address created using SLAAC, whether a public or temporary address, is associated with a state based on the age (lifetime) of the address. The types of addresses and lifetimes are described here and illustrated in Figure 9-7:

- **Tentative address:** The uniqueness of the address is in the process of being verified. A tentative address is not considered to be assigned to an interface. An interface discards received packets addressed to a tentative address but accepts Neighbor Discovery packets related to Duplicate Address Detection for the tentative address.
- **Valid address:** The address is a preferred or deprecated address. A valid address can be the source or destination address of a packet. The amount of time that an address remains in the valid and preferred states is included in the Router Advertisement message. The Valid Lifetime must be greater than or equal to the Preferred Lifetime.
  - **Preferred address:** The interface address has been verified as unique. The device can send and receive traffic using this address. New connections can be initiated using a preferred address as the source address. The period of time that an address can remain in the preferred state is included in the Router Advertisement message.
  - **Deprecated address:** The address assigned to an interface is still valid, but implementation is discouraged. A deprecated address should no longer be used as a source address in new communications, but packets sent from or to deprecated addresses are delivered as expected. A deprecated address can continue to be used as a source address in existing communications where changing to a preferred address might cause a problem with specific upper-layer activity, such as an existing TCP connection.
- **Invalid address:** A valid address becomes invalid when its Valid Lifetime expires. Invalid addresses should not appear as the destination or source address of a packet.

The amount of time addresses remain valid depends on two timers:

- **Preferred Lifetime:** This is the length of time a valid address is preferred until it becomes deprecated. When the Preferred Lifetime expires, the address becomes deprecated.
- **Valid Lifetime:** This is the length of time an address remains in the valid state. The Valid Lifetime must be greater than or equal to the Preferred Lifetime. When the Valid Lifetime expires, the address becomes invalid.

RFC 4862 discusses the reason for and usage of the preferred and deprecated addresses:

> Dividing valid addresses into preferred and deprecated categories provides a way of indicating to upper layers that a valid address may become invalid shortly and that future communication using the address will fail, should the address's valid lifetime expire before communication ends. To avoid this scenario, higher layers should use a preferred address (assuming one of sufficient scope exists) to increase the

likelihood that an address will remain valid for the duration of the communication. It is up to system administrators to set appropriate prefix lifetimes in order to minimize the impact of failed communication when renumbering takes place. The deprecation period should be long enough that most, if not all, communications are using the new address at the time an address becomes invalid.

Figure 9-7 illustrates the address states and lifetimes. This diagram can help visualize the transition in the example that follows.

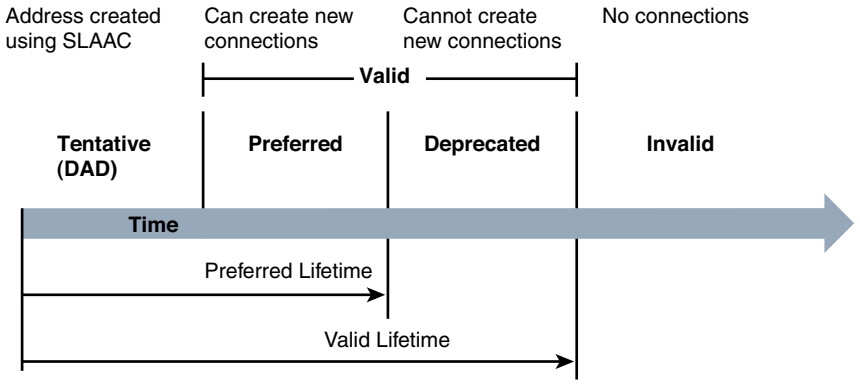


Figure 9-7 *Address States and Lifetimes*

Example: Autoconfigured Address States and Lifetimes

The Valid Lifetime and Preferred Lifetime are sent in the Router Advertisement message. Using the same topology used earlier in the chapter, shown again in Figure 9-8, let's again examine an RA message sent from router R1. Example 9-11 uses the **debug ipv6 nd** command to see the RA from router R1. You can see that **2592000/604800**, the Valid Lifetime and Preferred Lifetime sent in the RA, is highlighted:

- **Valid Lifetime:** 2,592,000 seconds, or 30 days
- **Preferred Lifetime:** 604,800 seconds, or 7 days

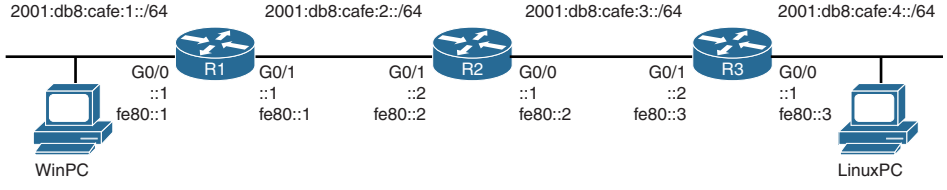


Figure 9-8 *Topology for Autoconfigured Address States and Lifetimes Example*

Example 9-11 *Examining R1's RA Messages Using the* **debug ipv6 nd** *Command*

```
R1# debug ipv6 nd
  ICMP Neighbor Discovery events debugging is on
R1#
*Nov 27 18:34:52.494: ICMPv6-ND: (GigabitEthernet0/0,FE80::1) send RA to FF02::1
*Nov 27 18:34:52.494: ICMPv6-ND: (GigabitEthernet0/0,FE80::1) Sending RA (1800) to
  FF02::1
*Nov 27 18:34:52.494: ICMPv6-ND:  MTU = 1500
*Nov 27 18:34:52.494: ICMPv6-ND:    prefix 2001:DB8:CAFE:1::/64 [LA] 2592000/604800
<output omitted for brevity>
R1# undebug all
```

Router R3 is using the same default lifetime values. WinPC and LinuxPC have both been configured to obtain their IPv6 address information dynamically. Using the process diagram in Figure 9-9, let's examine the process of WinPC and LinuxPC generating their addresses. Both hosts' operating systems implement the privacy extension by default. This means that each creates both a public and a temporary IPv6 address.

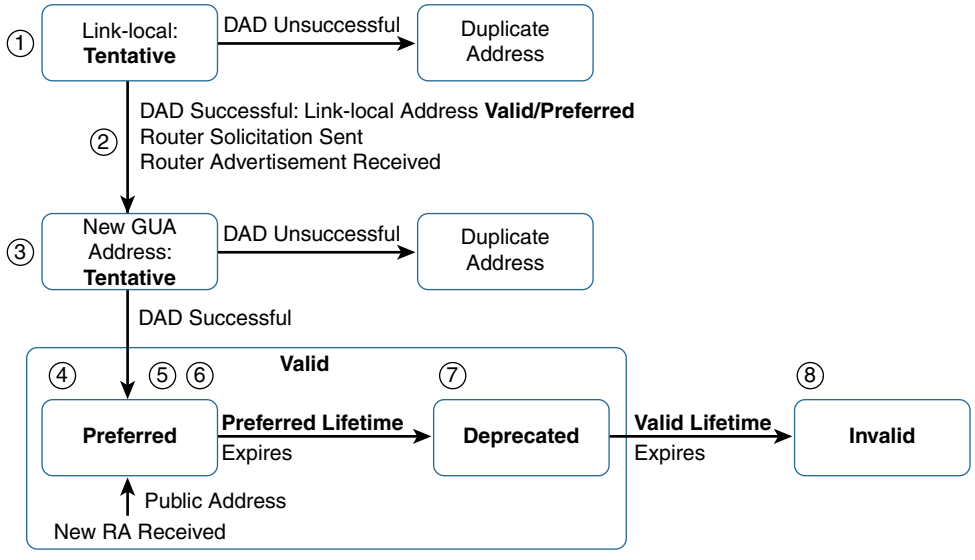


Figure 9-9 *SLAAC Address State Process Diagram*

Note This chapter includes examples using both WinPC and LinuxPC to show how different operating systems implement the privacy extension for SLAAC.

Figure 9-9 shows the following process:

Step 1. **Link-Local Address:** IPv6-enabled devices such as Windows and Linux computers automatically create an IPv6 link-local address on their interface(s) during startup.

- **WinPC:** WinPC creates a link-local address, fe80::d0f8:9ff6:4201:7086, and performs Duplicate Address Detection (DAD) to ensure that the address is unique. During the DAD process, the link-local address is in tentative state (that is, it is a tentative address). Once the address is determined to be unique, the link-local address transitions to the valid/preferred state. Because a link-local address is not used for communications beyond the link, there are not the same privacy concerns as with a global unicast address. Therefore, the Valid Lifetime and Preferred Lifetime are set to infinite. The link-local address remains in the preferred valid state indefinitely. Notice in Example 9-12 that the Interface ID is created using a randomized Interface ID.
- **LinuxPC:** The same process occurs for the LinuxPC and its link-local address, fe80::250:56ff:feaf:2524, shown in Example 9-13. Notice that the Interface ID has fffe in the middle, which most likely means it was created using EUI-64.

Step 2. **RS and RA Message:** Using the link-local address as the source IPv6 address, WinPC and LinuxPC each send a Router Solicitation message requesting a Router Advertisement. (RA messages are also sent periodically.)

- **Router R1:** Router R1, the local router for WinPC, sends the Router Advertisement message with the prefix 2001:db8:cafe:1::/64 and the Valid Lifetime and Preferred Lifetime set to 2592000/604800.
- **Router R3:** Router R3, the local router for LinuxPC, sends a similar Router Advertisement message with the prefix 2001:db8:cafe:4::/64 and the same Valid Lifetime and Preferred Lifetime settings, 2592000/604800.

Step 3. **SLAAC:** Devices use the information in the RA to generate one or more routable address.

- **WinPC:** Using the prefix in R1's RA, WinPC creates two addresses using SLAAC:

  Public IPv6 address: 2001:db8:cafe:1:d0f8:9ff6:4201:7086

  Temporary IPv6 address: 2001:db8:cafe:1:78bd:10b0:aa92:62c

  Both of these addresses are initially put in the tentative state while DAD is performed to ensure their uniqueness. Also, they both use randomized Interface IDs.

  Example 9-11 shows that the prefix 2001:db8:cafe:1::/64 has the L flag set to 1, which indicates to the WinPC that this prefix is on-link. 2001:db8:cafe:1::/64 is the on-link prefix for both addresses. Any packets with a destination IPv6 using this prefix can be sent directly to the device.

- **LinuxPC:** LinuxPC also uses SLAAC to create both public and temporary addresses, using the prefix in R3's RA:

  Public IPv6 address: 2001:db8:cafe:4:250:56ff:feaf:2524

  Temporary IPv6 address: 2001:db8:cafe:4:314a:dd3e:762f:e140

  Notice that Ubuntu Linux uses EUI-64 to create the Interface ID for its public address, but a randomized identifier for the Interface ID of its temporary address. The Interface ID of a temporary address can only be created using a randomized identifier, or it defeats the purpose of trying to provide privacy.

  Similar to WinPC, the RA received by LinuxPC has the prefix 2001:db8:cafe:4::/64 and has the L flag set to 1, indicating that this prefix is on-link.

Step 4. **Valid/Preferred:** Once DAD determines that these addresses are unique, the addresses transition from the tentative to the valid/preferred state. The public and temporary global unicast addresses along with the link-local unicast address for WinPC are shown in Example 9-12. Example 9-13 shows the global unicast addresses and link-local unicast address on LinuxPC.

Example 9-12 *WinPC's Addressing Information*

```
WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086
   Temporary IPv6 Address. . . . . . : 2001:db8:cafe:1:78bd:10b0:aa92:62c
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
<output omitted for brevity>
```

Example 9-13 *LinuxPC's Addressing Information*

```
LinuxPC$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:af:25:24
          inet6 addr: 2001:db8:cafe:4:250:56ff:feaf:2524/64 Scope:Global
          inet6 addr: fe80::250:56ff:feaf:2524/64 Scope:Link
          inet6 addr: 2001:db8:cafe:4:314a:dd3e:762f:e140/64 Scope:Global
<output omitted for brevity>
```

Step 5. **Public Address:** The public address uses the Valid Lifetime and Preferred Lifetime in the Router Advertisement—30 days (2,592,000 seconds) and 7 days (604,800 seconds), respectively.

- **WinPC:** Current Windows lifetimes can be displayed using the **netsh interface ipv6 show address 11** command, as shown in Example 9-14. (The 11 is the zone ID shown in Example 9-12.) These are the results for this example:

  2001:db8:cafe:1:d0f8:9ff6:4201:7086

  Valid Lifetime: 29 days 23 hours 59 minutes 32 seconds

  Preferred Lifetime: 6 days 23 hours 59 minutes 32 seconds

- **LinuxPC:** To see the current Valid Lifetime and Preferred Lifetime on the Ubuntu Linux host LinuxPC, use the **ip -6 addr show dev** *interface* command, shown in Example 9-15. These are the results for this example:

  2001:db8:cafe:4:250:56ff:feaf:2524

  Valid Lifetime: 2,591,947 seconds

  Preferred Lifetime: 604,747 seconds

  These are current lifetimes. In other words, the displayed lifetimes for WinPC and LinuxPC have been decremented since the last time an RA message was received. Each time the host receives an RA, the Valid Lifetime and Preferred Lifetime are reset to the value in the RA.

Note Depending on the values of the lifetimes in the RA, the actual lifetimes used by a host may differ from those in the RA. This is beyond the scope of this book. See RFC 4862 for more information.

Step 6. **Temporary Address:** A temporary address typically has a shorter lifetime than a public address. Unlike for a public address, the Preferred Lifetime is not reset when another RA is received. (This may change, depending on factors beyond the scope of this book.) The device continues to decrement the Preferred Lifetime of the temporary address until it becomes deprecated.

- **WinPC:** Windows defaults to a Valid Lifetime and Preferred Lifetime of 7 days (604800 seconds). The current lifetimes are shown in Example 9-14:

  2001:db8:cafe:1:78bd:10b0:aa92:62c

  Valid Lifetime: 6 days 23 hours 59 minutes 32 seconds

  Preferred Lifetime: 6 days 23 hours 59 minutes 32 seconds

- **LinuxPC:** Temporary addresses on the Ubuntu Linux host default to a Valid Lifetime of 7 days (604,800 seconds) and a Preferred Lifetime of 1 day (86,400 seconds). The current lifetimes are shown in Example 9-15:

  2001:db8:cafe:4:314a:dd3e:762f:e140

  Valid Lifetime: 604,747 seconds

  Preferred Lifetime: 85,747 seconds

The Valid Lifetime and Preferred Lifetime shown for both devices are decremented times since this temporary address was created. In other words, unlike with the public addresses, these temporary address lifetimes continue to decrement until they become deprecated.

Step 7. **Deprecated Address:** After a Preferred Lifetime expires, the address state becomes deprecated, and no new connections should be made using this address. When a temporary address becomes deprecated, a new temporary address must be generated. In normal operations, there should be no more than one temporary address that is in the valid/preferred state.

Step 8. **Invalid:** If no new RA is received, the Valid Lifetime eventually expires, and the address becomes invalid. The address is then removed from the interface.

Example 9-14 *Valid Lifetime and Preferred Lifetime for WinPC Addresses*

```
WinPC> netsh interface ipv6 show address 11
Address 2001:db8:cafe:1:d0f8:9ff6:4201:7086 Parameters
---------------------------------------------------------
Interface Luid     : Local Area Connection
Scope Id           : 0.0
Valid Lifetime     : 29d23h59m32s
Preferred Lifetime : 6d23h59m32s
DAD State          : Preferred
Address Type       : Public
Skip as Source     : false

Address 2001:db8:cafe:1:78bd:10b0:aa92:62c Parameters
---------------------------------------------------------
Interface Luid     : Local Area Connection
Scope Id           : 0.0
Valid Lifetime     : 6d23h59m32s
Preferred Lifetime : 6d23h59m32s
DAD State          : Preferred
Address Type       : Temporary
Skip as Source     : false

Address fe80::d0f8:9ff6:4201:7086%11 Parameters
---------------------------------------------------------
Interface Luid     : Local Area Connection
Scope Id           : 0.11
Valid Lifetime     : infinite
Preferred Lifetime : infinite
DAD State          : Preferred
Address Type       : Other
Skip as Source     : false
```

Example 9-15 *Valid and Preferred Lifetime for Linux Addresses*

```
LinuxPC# ip -6 addr show dev eth0

2: eth0: <BROADCAST,MULTICAST,UP, LOWER_UP> mtu 1500 state UNKNOWN qlen1000
  inet6 2001:db8:cafe:4:314a:dd3e:762f:e140/64 scope global temporary dynamic
     valid_lft 604747sec preferred_lft 85747sec

  inet6 2001:db8:cafe:4:250:56ff:feaf:2524/64 scope global dynamic
     valid_lft 2591947sec preferred_lft 604747sec

 inet6 fe80::250:56ff:feaf:2524/64 scope link
     valid_lft forever preferred_lft forever
```

For more information on the generation and expiration of temporary addresses, see RFCs 4862 and 4941.

Displaying IPv6 Lifetimes and State Information on Windows, Linux, and Mac OS

Examples 9-11 and 9-12 show the commands to view the Valid Lifetime and Preferred Lifetime on Windows and Linux. This section provides a summary of those commands.

To display IPv6 addresses assigned per interface, their address type, Duplicate Address Detection (DAD) state (preferred or deprecated), and Valid and Preferred Lifetimes, use the commands:

```
C:\> netsh interface ipv6 show address
C:\> netsh interface ipv6 show address zone-id
```

To display the list of IPv6 interfaces, their interface index, interface metric, maximum transmission unit (MTU), state, and name, use the command:

```
C:\> netsh interface ipv6 show interface
```

To display privacy parameters, including Valid Lifetime and Preferred Lifetime, use the command:

```
C:\> netsh interface ipv6 show privacy
```

Note For IPv6 information specific to Microsoft Windows, see Ed Horley's book *Practical IPv6 for Windows Administrators*. To stay current on everything IPv6, see Ed's website and blog, howfunky.com.

In Linux, to display interface address information, including Valid Lifetime and Preferred Lifetime, use this command:

```
Linux# ip -6 addr show dev interface
```

In Mac OS, to display the Valid Lifetime and Preferred Lifetime of addresses, use the command:

```
MacOS# ifconfig -L
```

Router Advertisement Fields and Options

This section examines the fields in the Router Advertisement message, along with some of the options to modify the RA, such as modifying the Valid Lifetime and Preferred Lifetime and the addition of DNS addresses in the RA.

Examining the Router Advertisement with Wireshark

Example 9-16 displays a Wireshark protocol analysis of R1's ICMPv6 Router Advertisement message. This section discusses the fields highlighted in the example. For a description of all the fields included in an RA, see RFC 4861, *Neighbor Discovery in IPv6*.

The ICMPv6 message is encapsulated in an IPv6 header (not shown in Example 9-16):

- **Source Address:** fe80::1 (link-local address of R1)
- **Destination Address:** ff02::1 (all-IPv6 devices multicast group or a solicited unicast)
- **Next Header:** 0x3a (an ICMPv6 header, 58 in decimal)

Example 9-16 *Wireshark Analysis of R1's Router Advertisement*

```
Internet Control Message Protocol v6
    Type: Router Advertisement (134)
    Code: 0
    Checksum: 0xcaf0 [correct]
    Cur hop limit: 64
    Flags: 0xc0
        0... .... = Managed address configuration: Not set
        .0.. .... = Other configuration: Not set
        ..0. .... = Home Agent: Not set
        ...0 0... = Prf (Default Router Preference): Medium (0)
        .... .0.. = Proxy: Not set
        .... ..0. = Reserved: 0
    Router lifetime (s): 1800
    Reachable time (ms): 0
    Retrans timer (ms): 0
    ICMPv6 Option (Source link-layer address : 58:ac:78:93:da:00)
        Type: Source link-layer address (1)
        Length: 1 (8 bytes)
        Link-layer address: 58:ac:78:93:da:00 (58:ac:78:93:da:00)
    ICMPv6 Option (MTU : 1500)
        Type MTU (5)
```

```
    Length: 1 (8 bytes)
     Reserved
     MTU: 1500
ICMPv6 Option (Prefix information : 2001:db8:cafe:1::/64)
     Type: Prefix information (3)
     Length: 4 (32 bytes)
     Prefix Length: 64
     Flag: 0xc0
         1... .... = On-Link flag(L): Set
         .1.. .... = Autonomous address-configuration flag(A): Set
         ..0. .... = Router address flag(R): Not set
         ...0 0000 = Reserved: 0
     Valid Lifetime: 2592000
     Preferred Lifetime: 604800
     Reserved
     Prefix: 2001:db8:cafe:1::  (2001:db8:cafe:1::)
```

The following are some of the significant fields in the ICMPv6 message:

- **Type (134):** The Type field is set to 134, indicating that this is a Router Advertisement message.
- **Cur Hop Limit (64):** The Cur Hop Limit (Current Hop Limit) is the value the router recommends for hosts on the network to use as the Hop Limit field in their IPv6 packets. A value of 0 means that the router is not recommending a hop limit and that the host's operating system should determine its own value. The default is 64.
- **Managed Address Configuration flag (M flag) (0):** When set to 1, this tells the host to use stateful configuration (DHCPv6). The default is 0.
- **Other Configuration flag (O flag) (0):** When set to 1, this tells the host that additional information is available from the DHCPv6 server, such as a domain name or DNS-related information. The default is 0.

Note If neither M nor O flags are set, this indicates that information is unavailable through DHCPv6.

- **Default Router Preference (Medium):** When receiving RA messages from multiple routers, the Default Router Preference (DRP) is used to determine which router to prefer as the default gateway. The preference values are High (01), Medium (00), Low (11), and Reserved (10). If the DRP values are equal, the host uses the source address from the first RA message it received as its default gateway. The default is Medium.
- **Router Lifetime (1800):** This tells a host the duration, in seconds, for which the router should be used as the default gateway. A lifetime of 0 indicates that the router is not a default gateway. The Router Lifetime applies only to the router's function as a default gateway. It does not apply to other information contained in other message

fields or options, such as prefix and prefix length. The host refreshes its own timer every time it receives a Router Advertisement. The default is 1800 (seconds).

Note The format of an RA message and its options is discussed in more detail in Chapter 13, "ICMPv6 Neighbor Discovery."

The following fields are options following the ICMPv6 header:

- **(Source) Link Layer Address (58:ac:78:93:da:00):** This is the Layer 2 link layer (data link layer) address of the sender. In this example, it is R1's Ethernet source MAC address.
- **MTU (1500):** This informs hosts of the maximum transmission unit (MTU) for the network. Hosts use this information to maximize the size of the IPv6 packet.
- **Prefix Length (64):** This is the number of leading bits in the prefix that are valid. The value ranges from 0 to 128. The Prefix Length field provides necessary information for on-link determination (when combined with the L flag in the prefix information option). It also assists with address autoconfiguration.
- **On-Link flag (L flag) (1):** When set to 1, this indicates that the prefix can be used for on-link determination; the prefix advertised in the RA is on this link (subnet). When it is not set, the advertisement makes no statement about on-link or off-link properties of the prefix. The default is 1.
- **Autonomous Address Configuration flag (A flag) (1):** When this flag is set to 1 (on), it tells the receiving host to use SLAAC to create its global unicast address. The default is 1.
- **Valid Lifetime (2592000):** This is the length of time an address remains in the valid state. The Valid Lifetime must be greater than or equal to the Preferred Lifetime. When the Valid Lifetime expires, the address becomes invalid. The default is 2,592,000 seconds (30 days).
- **Preferred Lifetime (604800):** This is the length of time a valid address is preferred. When the Preferred Lifetime expires, the address becomes deprecated. The default is 604,800 seconds (7 days).
- **Prefix (2001:db8:cafe:1::):** This notifies the host of the prefix that can be used for Stateless Address Autoconfiguration.

Note Wireshark is available as a free download from www.wireshark.org. For books and other resources for Wireshark from one of the founders of Wireshark University, Laura Chappell, go to www.wiresharkbook.com.

Modifying the Valid Lifetime and Preferred Lifetime in the RA Message

The default Valid Lifetime and Preferred Lifetime that are sent in the Router Advertisement message can be modified using the interface command:

```
Router(config-if)# ipv6 nd prefix ipv6-prefix/prefix-length [valid-lifetime]
[preferred-lifetime]
```

In Example 9-17, the Valid Lifetime and Preferred Lifetime for R1's G0/0 Router Advertisement message are modified to 15 days (1,296,000 seconds) and 2 days (172,800 seconds), respectively.

Example 9-17 *Examining R1's New Lifetimes Using the* **debug ipv6 nd** *Command*

```
R1(config)# interface gigabitethernet 0/0
R1(config-if)#ipv6 nd prefix 2001:db8:cafe:1::/64 1296000 172800
R1(config-if)#end
R1# debug ipv6 nd
  ICMP Neighbor Discovery events debugging is on
R1#
*Nov 27 20:12:50.490: ICMPv6-ND: (GigabitEthernet0/0,FE80::1) send RA to FF02::1
*Nov 27 20:12:50.490: ICMPv6-ND: (GigabitEthernet0/0,FE80::1) Sending RA (1800) to
  FF02::1
*Nov 27 20:12:50.490: ICMPv6-ND:   MTU = 1500
*Nov 27 20:12:50.490: ICMPv6-ND:   prefix 2001:DB8:CAFE:1::/64 [LA] 1296000/172800
<output omitted for brevity>
R1# undebug all
```

Including the DNS Address in the Router Advertisement

The Router Advertisement message provides information such as prefix, prefix length, MTU, and other information. Some of this information, such as the prefix, depends on the configuration of the interface. Other information, such as the MTU, is associated with the type of interface. There is also information with preset default values such as the Valid Lifetime and Preferred Lifetime.

DNS server addresses are not included in the RA by default. The RA message must be configured to include these addresses. RFC 6106, *IPv6 Router Advertisement Options for DNS Configuration*, defines the Recursive DNS Server (RDNSS) and DNS Search List (DNSSL) options in the Router Advertisement. Prior to RFC 6106, DNS addresses could only be obtained using stateless or stateful DHCPv6.

Note RFC 6106 uses the term RDNSS. A recursive DNS nameserver is the DNS server responsible for providing the proper IP address of the intended domain name to the requesting host. In comparison, an authoritative DNS nameserver is responsible for providing the answers to RDNSS queries.

To configure a Cisco router to include a list of DNS servers in its Router Advertisement, use the interface command:

```
Router(config-if)# ipv6 nd ra dns server ipv6-address dns-lifetime
```

Up to eight DNS server addresses can be included. *dns-lifetime* is the amount of time (in seconds) that the DNS server is advertised in the RA message. The range is from 200 to 4,294,967,295 seconds, with a default of 400 seconds. Example 9-18 shows the configuration and verification, including a DNS server on router R1.

Note Google public DNS IPv6 addresses are 2001:4860:4860::8888 and 2001:4860:4860::8844.

Example 9-18 *Configuring and Verifying the RDNSS Option on R1*

```
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 nd ra dns server 2001:db8:cafe:99::9999
R1(config-if)# end
R1# debug ipv6 nd
  ICMP Neighbor Discovery events debugging is on
R1#
*Dec  3 16:14:04.647: ICMPv6-ND: (GigabitEthernet0/0,FE80::1) send RA to FF02::1
*Dec  3 16:14:04.647: ICMPv6-ND: (GigabitEthernet0/0,FE80::1) Sending RA (1800)
  to FF02::1
*Dec  3 16:14:04.647: ICMPv6-ND:   MTU = 1500
*Dec  3 16:14:04.647: ICMPv6-ND:   DNS lifetime 400
*Dec  3 16:14:04.647: ICMPv6-ND:     server 2001:DB8:CAFE:99::9999
*Dec  3 16:14:04.647: ICMPv6-ND:   prefix 2001:DB8:CAFE:1::/64 [LA] 2592000/604800
R1#
```

Example 9-19 shows a Wireshark analysis of R1's Router Advertisement that includes the RDNSS option.

Example 9-19 *Wireshark Analysis of RDNSS Option in R1's Router Advertisement*

```
Internet Control Message Protocol v6
   Type: Router Advertisement (134)
<output omitted for brevity>
ICMPv6 Option (Recursive DNS Server 2001:db9:cafe:99::99)
     Type: Recursive DNS Server (25)
     Length: 3 (24 bytes)
     Reserved
     Lifetime: 400
     Recursive DNS Servers: 2001:db9:cafe:99::99 (2001:db9:cafe:99::99)
```

It is important that both the router sending the RA message and the devices using the RA for SLAAC support RFC 6106. To cover all bases, some implementations send the DNS server address in the RA and also use stateless DHCPv6 to advertise the same DNS address. This provides a transition for operating systems that do not yet support RFC 6106. A good resource for IPv6 support in various operating systems is the Wikipedia page *Comparison of IPv6 Support in Operating Systems*, at en.wikipedia.org/wiki/Comparison_of_IPv6_support_in_operating_systems.

Router Advertisement Configuration Options

There are several options for managing and configuring a Router Advertisement message. The following is a summary of some of these options:

Note A couple of these options have been discussed previously, and some of the others will be discussed in the Chapters 9 and 10.

- **Other Configuration flag:** Setting the Other Configuration flag indicates to attached hosts that they should use a stateless DHCPv6 server to obtain the other (nonaddress) information:

```
Router(config-if)# ipv6 nd other-config-flag
```

  This option is discussed further in Chapter 10.

- **Managed Address Configuration flag:** Setting the Managed Address Configuration flag indicates to attached hosts that they should use stateful DHCPv6 to obtain addresses and other information:

```
Router(config-if)# ipv6 nd managed-config-flag
```

  This option is discussed further in Chapter 11.

- **Modifying the Default Router Preference (DRP):** This command is used to modify the Default Router Preference on a specific interface:

```
Router(config-if)# ipv6 nd router-preference { high | medium | low }
```

  For example, a DRP is useful when two routers on a link are providing equivalent but not equal-cost routing. Policy may dictate that hosts should prefer one of the routers. The default is medium.

- **Modifying Prefix Parameters:** This command is used to configure IPv6 prefixes that are included in the Router Advertisements:

```
Router(config-if)#  ipv6 nd prefix { ipv6-prefix/prefix-length | default }
[ no-advertise | [ valid-lifetime preferred-lifetime [ off-link |
no-rtr-address | no-autoconfig | no-onlink ] ] ] at valid-date |
preferred-date [ off-link | no-rtr-address | no-autoconfig ]
```

- This command allows control over the individual parameters per prefix, including not advertising a specific prefix. This is discussed further in Chapter 11.

  By default, the prefixes of addresses configured on an interface are included in the RA. Additional prefixes can also be advertised using the **ipv6 nd prefix** command. When specific prefixes are configured using the **ipv6 nd prefix** command, only these prefixes are advertised.

  All prefixes configured on interfaces that originate IPv6 Router Advertisements are advertised with a Valid Lifetime of 2,592,000 seconds (30 days) and a Preferred Lifetime of 604,800 seconds (7 days).

  By default:

  - All prefixes are inserted in the routing table as Connected prefixes.
  - All prefixes are advertised as on-link (the L flag is set to 1).
  - All prefixes are advertised as autoconfiguration prefixes (the A flag is set to 1).

  The syntax shown above is described as follows:

  - ***ipv6-prefix***: Specifies the IPv6 network number to include in router advertisements (RAs).
  - ***/prefix-length***: Specifies the length of the IPv6 prefix.
  - **default:** Specifies that the default values should be used.
  - **no-advertise:** (Optional) Specifies that the prefix will not be advertised. This also means there is no On-Link flag set for this prefix to indicate that the prefix is on-link. This can result in unintended consequences for addresses using this prefix that may be assigned manually or using DHCPv6. SLAAC with this prefix wouldn't be an option since the prefix is not being advertised.
  - ***valid-lifetime***: (Optional) Specifies the amount of time (in seconds) that the specified IPv6 prefix is advertised as being valid. The range is from 0 to 4,294,967,295 seconds.
  - ***preferred-lifetime***: (Optional) Specifies the amount of time (in seconds) that the specified IPv6 prefix is advertised as being preferred. The range is from 0 to 4,294,967,295 seconds.
  - **off-link:** (Optional) Configures the specified prefix as off-link. The prefix is advertised with an A flag of 0. The prefix is not inserted into the routing table as a Connected prefix. If the prefix is already present in the routing table as a Connected prefix (for example, because the prefix was also configured using the **ipv6 address** command), then it is removed.
  - **no-rtr-address:** (Optional) Indicates that the router does not send the full router address in prefix advertisements and does not set the R bit.

- **no-autoconfig:** (Optional) Indicates to hosts on the local link that the specified prefix cannot be used for IPv6 autoconfiguration. The prefix is advertised with the A flag set to 0.
- **no-onlink:** (Optional) Configures the specified prefix as not on-link. The prefix is advertised with the L flag at 0.
- **at *valid-date*:** (Optional) Specifies the date and time at which the lifetime and preference expire. The prefix is valid until this specified date and time are reached.
- ***preferred-date*:** (Optional) Specifies the preferred expiration date.

- **Including the DNS Server:** As discussed previously, this command configures the RA to include a list of DNS server addresses:

```
Router(config-if)# ipv6 nd ra dns server ipv6-address seconds
```

This command can be used to configure up to eight DNS server addresses in an RA. The optional *seconds* is the amount of time (in seconds) that the DNS server is valid, as advertised in the RA. The range is from 200 to 4,294,967,295 seconds. A value of 0 means the DNS server will no longer be used.

- **Configuring the Router Lifetime:** This command is used to configure the Router Lifetime value in IPv6 Router Advertisements on an interface:

```
Router(config-if)# ipv6 nd ra lifetime seconds
```

As discussed previously, the Router Lifetime applies only to the router's function as a default gateway. The default is 1,800 seconds. Setting the value to 0 indicates that the router should not be considered a default router on this interface. The Router Lifetime value can be set to a nonzero value to indicate that it should be considered a default router on this interface. The nonzero value for the Router Lifetime value should not be less than the Router Advertisement interval.

- **Modifying the Router Advertisement Interval:** This command is used to configure the interval between IPv6 Router Advertisement transmissions on an interface:

```
Router(config-if)# ipv6 nd ra interval { maximum-secs [minimum-secs] | msec
maximum-ms [minimum-ms] }
```

If you configure Router Lifetime (the lifetime for which the router can be used as a default gateway), the interval between RA transmissions should be less than or equal to the Router Lifetime. To prevent synchronization with other IPv6 nodes, the actual interval used is randomly selected from a value between the minimum and maximum values.

Users can explicitly configure a minimum RA interval. The minimum RA interval may never be more than 75% of the maximum RA interval and never less than 3 seconds (if specified in seconds). If the minimum RA interval is not configured, it is calculated as 75% of the maximum RA interval.

If the user specifies the time in milliseconds, then the minimum RA interval is 30 milliseconds. This limit allows configuration of very short RA intervals for Mobile IPv6.

The maximum and minimum RA intervals govern only unsolicited RA messages (periodic Router Advertisements). RA messages are also sent upon receiving an RS message. If multiple RS messages are received every second, there is a minimum delay of 3 seconds between the RA messages. This limits the number of solicited RA messages transmitted from the interface.

- **Sending a Solicited Unicast Router Advertisement:** This command configures a unified solicited Router Advertisement response method on an interface:

```
Router(config-if)# ipv6 nd ra solicited unicast
```

Solicited Router Advertisements are sent in response to a Router Solicitation message. Large networks with a high concentration of mobile devices might experience unnecessary battery depletion due to a high volume of solicited Router Advertisement messages sent as multicasts to all devices. The **ipv6 nd ra solicited unicast** command is used to send the RA as a unicast when it was requested by a Router Solicitation message. This helps extend the battery life of mobile devices in the network.

- **Suppressing the Router Advertisement:** This command is used to suppress IPv6 Router Advertisement transmissions on a LAN interface:

```
Router(config-if)# ipv6 nd ra suppress [all]
```

IPv6 Router Advertisements are automatically sent on Ethernet and FDDI interfaces if IPv6 unicast routing is enabled on the interfaces. IPv6 Router Advertisements are not sent on other types of interfaces. The **all** option suppresses all RAs on an interface.

When using stateful DHCPv6, it is important not to suppress the RA message on the interface without knowing the consequences:

- Stateful DHCPv6 does not provide the default gateway. This can only be obtained dynamically from the Router Advertisement.
- The On-Link flag in the RA (when set to 1) is used to indicate that the prefix in the RA is on-link. This is used by addresses configured manually, using SLAAC, or stateful DHCPv6 to determine whether the address is on-link. Otherwise, the address is considered off-link.

Note For a complete listing of IPv6 RA and other IPv6 IOS commands, see *Cisco IOS IPv6 Command Reference*, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book.html.

Default Address Selection

IPv6 addressing architecture allows for multiple unicast addresses to be assigned to the same interface. These addresses may differ in the following ways:

- Scope (link-local, global, and so on)
- Public or temporary
- Preferred or deprecated
- "Home address" or "care-of address" (used for mobility)

As a result, at times IPv6 devices must decide among multiple source addresses. It is necessary to have algorithms for selecting the proper address so developers and administrators can predict the behavior of their systems. This process is defined in RFC 6724, *Default Address Selection for Internet Protocol Version 6 (IPv6)*.

Note RFC 6724 discusses the default address selection for both source and default addresses. This chapter discusses only the source address selection process. For the complete details, along with the destination address selection process, see RFC 6724.

When there is more than one possible source address, the source address selection produces a single source IPv6 address for a given destination IPv6 address. The algorithm uses pairwise comparison rules between the source addresses. There are eight rules, applied in the following order:

- **Rule 1: Prefer same address:** Prefer the same address if the source and destination addresses are the same. For example, if you have WinPC ping itself, it should use the same address for the source address that it uses for the destination address.
- **Rule 2: Prefer appropriate scope:** Prefer address pairs that are of the same scope or type (link-local, global, and so on). For example, if WinPC pings the link-local address of router R1, it should use its own link-local address as the source address.
- **Rule 3: Avoid deprecated addresses:** A preferred (non-deprecated) address is preferred over a deprecated address. Deprecated addresses need to be avoided and used only to continue existing communications. If WinPC has both a preferred and deprecated address on the same interface, it should use the preferred address when initiating any new communications.
- **Rule 4: Prefer home addresses (mobility):** Prefer a home address to a care-of address. A *home address* is an IPv6 address assigned to a mobile node and used as the permanent address. This is the "normal" permanent IPv6 address used by the device on its home network. Packets sent to this device are always sent to the home address.

A *care-of address* is a secondary, temporary IPv6 address associated with a mobile node when visiting a foreign link, away from its home link. When a mobile node is on its home link, it might have an address that is both its home address and a care-of address.

- **Rule 5: Prefer outgoing interface:** Prefer a source address that is on the same outgoing interface used to forward the packet. In other words, this rule favors a source address that is on the same interface that will be used to send the packet to the destination address.
- **Rule 6: Prefer matching label:** The default policy table is a longest-matching-prefix lookup table, much like a routing table. The table includes the prefix, precedence, and a label. The precedence is used to sort the table by destination address. The label is used by policies to prefer a specific source address prefix for use with a destination address prefix. These are addresses with the same label. This results in the preference of using native source addresses with native destination addresses, such as 6to4 source addresses with 6to4 destination addresses. Example 9-20 shows an example of WinPC's default policy table.

Example 9-20 *WinPC's Default Policy Table*

```
WinPC> netsh interface ipv6 show prefixpolicies
Querying active state...

Precedence  Label  Prefix
----------  -----  --------------------------------
        50      0  ::1/128
        40      1  ::/0
        30      2  2002::/16
        20      3  ::/96
        10      4  ::ffff:0:0/96
         5      5  2001::/32
```

- **Rule 7: Prefer temporary addresses:** This rule gives preference to privacy addresses, preferring a temporary address to a public address. For example, WinPC is initiating communications with LinuxPC. Because WinPC implements the privacy extension, it uses its temporary address in preferred state as the source address.
- **Rule 8: Use longest prefix matching:** Given a common prefix length, the rule prefers the address with the longest matching prefix. For example, PC-1 has two GUA addresses on its interface: 2001:db8:cafe:1001::1/64 (source address A) and 2001:db8:cafe:1fff::1/64 (source address B). PC-1 is going to ping PC-2 at 2001:db8:cafe:1000::1. Figure 9-10 shows that PC-2's 2001:db8:cafe: matches both of PC-1's addresses, but looking at the fourth hextet in binary, we can see that 2001:db8:cafe:1001::1 is a longer prefix match.

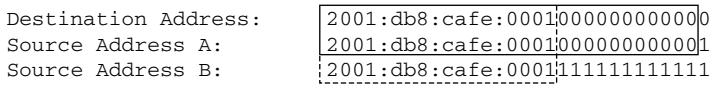


Figure 9-10 *SLAAC Address State Process Diagram*

Rule 8 may be superseded if the implementation has another method of choosing among the source addresses for reasons such as better communications performance.

Configuring the Router's Interface as a SLAAC Client

Typically router interfaces are configured manually with an IPv6 address. There might be times when you need a router to obtain its IPv6 address automatically, using Stateless Address Autoconfiguration. The **ipv6 address autoconfig** interface command performs IPv6 Stateless Address Autoconfiguration using Router Advertisement messages to discover prefixes on the link and then to add the EUI-64-based addresses to the interface. The syntax for the **ipv6 address autoconfig** interface command is described in Table 9-1.

Table 9-1 *Configuring the Router Interface for Stateless Address Autoconfiguration*

Command	Description
Router(config)# **interface** *type number*	Specifies an interface type and number and places the router in interface configuration mode.
Router(config-if)# **ipv6 address autoconfig**	Enables automatic configuration of IPv6 addresses using Stateless Address Autoconfiguration on an interface and enables IPv6 processing on the interface.

Summary

This chapter discusses the various aspects of Stateless Address Autoconfiguration (SLAAC). By default, a Cisco IPv6 router sends a Router Advertisement message every 200 seconds or after receiving a Router Solicitation message. For an IPv6 router, IPv6 unicast routing must be enabled.

The RA is sent from the link-local address of the router. Devices can use this address as their default gateway address. The destination address is ff02::1 (all-IPv6 devices multicast) or can be a solicited unicast address. Some of the fields in the RA include the following:

- **Cur Hop Limit:** This is the value the router recommends for hosts on the network to use as the Hop Limit field in their IPv6 packets.
- **Managed Address Configuration flag (M flag):** When set to 1, this tells the host to use stateful configuration (DHCPv6).

- **Other Configuration flag (O flag):** When set to 1, this tells the host that additional information is available from the DHCPv6 server, such as a domain name or DNS-related information.
- **Default Router Preference (DRP):** When receiving RA messages from multiple routers, the DRP is used to determine which router to prefer as the default gateway.
- **Router Lifetime:** Informs a host about the duration, in seconds, that the router should be used as the default gateway.
- **(Source) Link Layer Address:** This is the Layer 2 address of the sender.
- **MTU:** Informs hosts of the maximum transmission unit (MTU) for the network. Hosts use this information to maximize the size of the IPv6 packet.
- **On-Link flag (L flag):** When set, this flag indicates that this prefix can be used for on-link determination; the prefix advertised in the RA is on this link (subnet).
- **Autonomous Address Configuration flag (A flag):** When set to 1 (on), this flag tells the receiving host to use SLAAC to create its global unicast address.
- **Valid Lifetime:** Specifies the length of time an address remains in the valid state before becoming invalid.
- **Preferred Lifetime:** Specifies the length of time a valid address is preferred before becoming deprecated.
- **Prefix Information:** Notifies the host of the prefix and prefix length that can be used for Stateless Address Autoconfiguration.

IPv6 determines whether a prefix is on-link from the prefix with an On-Link flag set to 1 in a Router Advertisement. There are other methods as well, including a redirect message from a router.

The RA message can include various other options, such as a list of DNS server addresses or modifications to preset values.

By default, the A flag is set to 1, and the O and M flags are set to 0. This suggests to the device that it use the prefix in the RA to create its global unicast address and no other information is available using DHCPv6. Depending on the implementation of the operating system, the device generates its Interface ID using either EUI-64 or a random 64-bit value (privacy extension).

EUI-64 uses the MAC address for part of the address, which introduces privacy concerns. RFC 4941, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, addresses these concerns by using a randomized Interface ID and generating a second address, which is a temporary address. The first address is known as the public address. It can use either EUI-64 or a randomized 64-bit Interface ID. The temporary address must use a random 64-bit value for its Interface ID.

A temporary address is in one of five states:

- **Tentative address:** The uniqueness of the address is in the process of being verified using Duplicate Address Detection (DAD) and cannot be used other than for IPv6 Neighbor Discovery messages for DAD.
- **Valid address:** The address is a preferred or deprecated address. A valid address can be the source or destination address of a packet.
- **Preferred address:** The interface address has been verified as unique. The device can send and receive traffic using this address.
- **Deprecated address:** The address assigned to an interface is still valid, but implementation is discouraged. A deprecated address should no longer be used as a source address in new communications, but packets sent from or to deprecated addresses are delivered as expected.
- **Invalid address:** A valid address becomes invalid when its Valid Lifetime expires. An invalid address should not appear as the destination or source address of a packet.

The amount of time that addresses remain valid depends on two timers:

- **Preferred Lifetime:** The length of time a valid address is preferred, until it becomes deprecated
- **Valid Lifetime:** The length of time an address remains in the valid state before becoming invalid

IPv6 addressing architecture allows multiple unicast address to be assigned to the same interface. RFC 6724, *Default Address Selection for Internet Protocol Version 6 (IPv6)*, defines the process used for both source and default address selection.

Review Questions

1. By default, how often does an IPv6 unicast-enabled Cisco router send RA messages?
   a. Every 10 seconds
   b. Every 60 seconds
   c. Every 180 seconds
   d. Every 200 seconds
   e. Every 1800 seconds
2. Which flag is set to 1, suggesting to devices that they use SLAAC?
   a. Address Autoconfiguration flag (A flag)
   b. Other Configuration flag (O flag)
   c. Managed Address Configuration flag (M flag)
   d. On-Link flag (L flag)

3. What does the duration in the Router Lifetime specify?
   a. How long the information in an RA message should be considered valid
   b. How long this router should be used as a default gateway
   c. How long addresses created by SLAAC are considered valid
   d. How often RA messages will be sent on this link
4. How does an IPv6 device determine what subnet its GUA address is on?
   a. By using its prefix length
   b. By using the prefix length in the RA
   c. By using the prefix in the RA with the A flag set to 1
   d. By using the prefix in the RA with the L flag set to 1
5. Which SLAAC method generates Interface IDs using the MAC address?
6. The privacy extensions for SLAAC allow for which of the following? (Choose two.)
   a. Variable prefix lengths
   b. Randomized Interface IDs
   c. Address hiding using translation
   d. Temporary addresses
   e. Private addresses
7. Match the type of address to its description:

   Tentative address

   Valid address

   Preferred address

   Deprecated address

   Invalid address

   a. This address has been determined to be unique but should not be used as a source or destination address of a packet.
   b. This address has been verified as unique and can be used to initiate a new connection.
   c. This address includes both preferred and deprecated addresses.
   d. This address should only be used for existing connections but not for new connections.
   e. This address is in the process of being verified as unique and can't be used for connections other than for DAD.

8. What method is used to ensure that a unicast address is unique prior to its use?
9. What RA field can devices use to determine which default gateway to use when receiving RAs from different routers?
10. Using the default address selection process for source addresses, what source address would a device use when sending a ping to a link-local address on the same link?
11. Using the default address selection for source addresses, which source address would a device use to initiate a connection with a device on another network: a public address or a temporary address?
12. Using the default address selection for source addresses, which source address would a device use to initiate a connection with a device on another network: a preferred address or a deprecated address?

References

RFCs

RFC 4191, *Default router preferences and more specific routes*, R. Draves, Microsoft, www.ietf.org/rfc/rfc4191, November 2005.

RFC 4291, *IPv6 version 6 addressing architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc4291, February 2006.

RFC 4429, *Optimistic Duplicate Address Detection (DAD) for IPv6*, N. Moore, Monash University, www.ietf.org/rfc/rfc4429, April 2006.

RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*, T. Narten, IBM, www.ietf.org/rfc/rfc4861, September 2007.

RFC 4862, *IPv6 Stateless Address Autoconfiguration*, S. Thomson, Cisco Systems, www.ietf.org/rfc/rfc4862, September 2007.

RFC 4941, *Privacy extensions for Stateless Address Autoconfiguration in IPv6*, T. Narten, IBM, www.ietf.org/rfc/rfc4941, September 2007.

RFC 6106, *IPv6 Router Advertisement options for DNS configuration*, J. Jeong, Brocade/ETRI, tools.ietf.org/html/rfc6106, November 2010.

RFC 6724, *Default address selection for Internet Protocol version 6 (IPv6)*, D. Thaler, Microsoft, tools.ietf.org/rfc/rfc6724.txt, September 2012.

RFC 7042, *IANA considerations and IETF protocol usage for IEEE 802 parameters*, D. Eastlake, Eastlake Enterprises, tools.ietf.org/html/rfc7042, September 2008.

Websites

IPv6 Addressing (TechRef), technet.microsoft.com/en-us/library/dd392266(v=ws.10).aspx.

Netsh Commands for Interface Internet Protocol Version 6 (IPv6), technet.microsoft.com/en-us/library/cc753156(v=ws.10).aspx.

Comparison of IPv6 Support in Operating Systems, en.wikipedia.org/wiki/Comparison_of_IPv6_support_in_operating_systems.

IEEE Guidelines for Use Organizationally Unique Identifier (OUI) and Company ID (CID), http://standards.ieee.org/develop/regauth/tut/eui.pdf.

IEEE 1394-1995: IEEE Standard for a High Performance Serial Bus, standards.ieee.org/findstds/standard/1394-1995.html.

IEEE 802.15: Low-Rate Wireless Personal Area Networks (LR-WPANs), standards.ieee.org/about/get/802/802.15.html.

Cisco IOS IPv6 Command Reference, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book/ipv6-i3.html.

Ed Horley's Blog, www.howfunky.com.

Other

Practical IPv6 for Windows Administrators, by Ed Horley.

This page intentionally left blank

Chapter 10

Stateless DHCPv6

As mentioned in previous chapters, there are three methods for dynamically allocating IPv6 addressing and configuration information:

- **Method 1:** Stateless Address Autoconfiguration (SLAAC)
- **Method 2:** SLAAC and a stateless DHCPv6 server
- **Method 3:** Stateful DHCPv6 server

In Chapter 9, "Stateless Address Autoconfiguration (SLAAC)," we discussed the first method, using SLAAC. In this chapter we examine the second method, using SLAAC with the addition of stateless DHCPv6 for other configuration information. The focus here is on stateless DHCPv6. Everything we discussed in Chapter 9 regarding SLAAC applies to Method 2 as well, including the following:

- SLAAC is used to generate global unicast addresses using either EUI-64 or the privacy extension's 64-bit randomized identifier.
- The SLAAC privacy extension uses public and temporary addresses, and its various address states (tentative, valid, preferred, deprecated, invalid).
- On-link determination and default address selection.

The only difference with the method discussed in this chapter is that after a device generates one or more addresses using SLAAC, it contacts a stateless DHCPv6 server for additional information.

Remember that a *stateless* DHCPv6 server doesn't allocate or maintain any IPv6 global unicast addressing information. A stateless server only provides common network information that is available to all devices on the network, such as a list of DNS server addresses or a domain name.

SLAAC with Stateless DHCPv6

DHCPv6 is defined in RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, and stateless DHCPv6 is defined in RFC 3736, *Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6*. Figure 10-1 shows an overview of the process for SLAAC with stateless DHCPv6.

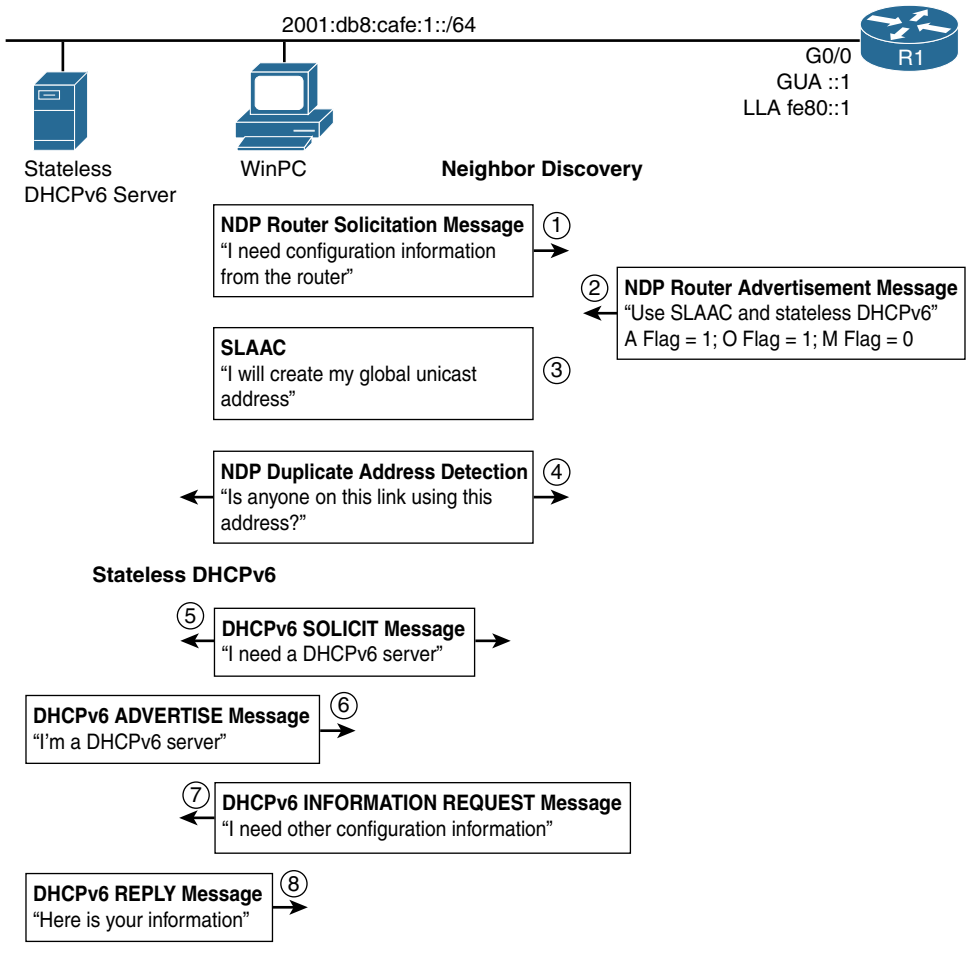


Figure 10-1 *Stateless DHCPv6 Operations*

As shown in Figure 10-1, the following steps are used for SLAAC with stateless DHCPv6:

Step 1. WinPC sends a Router Solicitation message unless a Router Advertisement message has already been received.

Step 2. The RA message is sent with the default A flag (Address Autoconfiguration flag) set to 1, suggesting to the receiving host that it use SLAAC. The O flag (Other Configuration flag) is configured to 1, suggesting that other configuration information is available from a stateless DHCPv6 server. The M flag (Managed Address Configuration flag) remains at its default of 0, indicating that the services of a stateful DHCPv6 server are not needed. (The contents of the RA message are displayed using Wireshark in the next section.)

Step 3. Upon receiving the RA, WinPC uses the source IPv6 address of the RA, fe80::1, as its default gateway address. Because the A flag is set to 1, WinPC performs all the same SLAAC operations discussed in the previous chapter, including the following:

- Using the prefix 2001:db8:cafe:1: in the RA message, it creates one or more global unicast addresses. The Interface ID is generated using either EUI-64 or a random 64-bit value. DAD is performed on all unicast addresses to ensure that they are unique.
- By default, Windows hosts use the privacy extension and create both a public and a temporary GUA address using a randomized identifier.

Step 4. WinPC performs DAD (Duplicate Address Detection) to ensure that any GUA addresses created using SLAAC are unique. DAD is essentially the same as a gratuitous ARP in IPv4. WinPC sends out a Neighbor Solicitation message, looking for a MAC address for its own IPv6 address. If it does not receive a reply (Neighbor Advertisement), then it knows the address is unique. DAD is discussed in Chapter 13, "ICMPv6 Neighbor Discovery."

Step 5. The RA message's O flag was set to 1, suggesting that additional information is available from a stateless DHCPv6 server. WinPC sends out a DHCPv6 SOLICIT message to ff02::1:2, the all-DHCPv6 servers multicast address.

Step 6. One or more DHCPv6 servers respond with a DHCPv6 ADVERTISE message, indicating that they are available for DHCPv6 service.

Step 7. WinPC responds to the selected server by sending an INFORMATION REQUEST message, asking for other configuration information.

Step 8. The DHCPv6 server responds with a REPLY message that contains the other configuration information.

Note The details of DHCPv6 messages are discussed in Chapter 8, "Basics of Dynamic Addressing in IPv6."

Implementing Stateless DHCPv6

Implementing SLAAC with stateless DHCPv6 requires the following:

- Setting the router's RA message's O flag to 1
- Configuring a stateless DHCPv6 server

Using the topology in Figure 10-2, the following sections demonstrate both tasks, including configuring router R1 as a stateless DHCPv6 server.

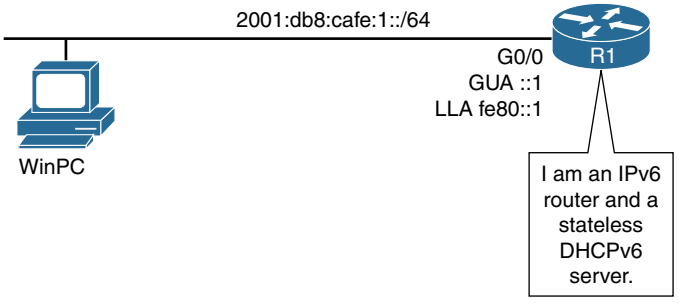


Figure 10-2 *Topology for Stateless DHCPv6 Configuration*

Configuring the RA Message's Other Configuration Flag

By default, the Other Configuration flag is set to 0. To set the O flag to 1, use this interface command:

```
Router(config-if)# ipv6 nd other-config-flag
```

The **no** option with this command sets this flag to its default of 0.

Example 10-1 configures R1 to send out its RA message with the O flag set to 1. The configuration of the **ipv6 unicast-routing** command has also been included, which is required to send the RA. The A flag, which suggests the use of SLAAC, is set to 1 by default and does not need to be configured. The **show ipv6 interface gigabitethernet 0/0** command is used to verify the change in the RA message.

Example 10-1 *Configuring and Verifying the RA's O Flag on R1*

```
R1(config)# ipv6 unicast-routing
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 nd other-config-flag
R1(config-if)# end
R1# show ipv6 interface gigabitethernet 0/0
```

```
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::1
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::FB
    FF02::1:FF00:1
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 30000)
  ND advertised reachable time is 0 (unspecified)
  ND advertised retransmit interval is 0 (unspecified)
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  ND advertised default router preference is Medium
  Hosts use stateless autoconfig for addresses.
  Hosts use DHCP to obtain other configuration.
R1#
```

The last two lines of the output of the **show ipv6 interface gigabitethernet 0/0** command indicate how hosts will obtain their addressing information:

- **Hosts use stateless autoconfig for addresses.** Indicates that the A flag is set to 1, suggesting to receiving devices that they use SLAAC for creating the global unicast address.
- **Hosts use DHCP to obtain other configuration.** Indicates that the O flag is set to 1, suggesting to receiving devices that they obtain other configuration information from a stateless DHCPv6 server.

Wireshark Analysis of Router Advertisement: SLAAC and Stateless DHCPv6

Example 10-2 displays a protocol analysis of R1's ICMPv6 Router Advertisement message using Wireshark. This is similar to the same RA message discussed in Chapter 9 with one difference: The O flag is set to 1 (highlighted in the example). Also highlighted is the A flag, which is set to its default value of 1.

The ICMPv6 Router Advertisement message is encapsulated in an IPv6 header (not shown in Example 10-2):

- **Source Address:** fe80::1 (Link-local address of R1)
- **Destination Address:** ff02::1 (all-IPv6 devices multicast group or can also be a solicited unicast)
- **Next Header:** 0x3a (Next Header is an ICMPv6 header, 58 in decimal)

Example 10-2 *Wireshark Analysis of R1's Router Advertisement*

```
Internet Control Message Protocol v6
    Type: Router Advertisement (134)
    Code: 0
    Checksum: 0x796a [correct]
    Cur hop limit: 64
    Flags: 0xc0
        0... .... = Managed address configuration: Not set
        .1.. .... = Other configuration: Set
        ..0. .... = Home Agent: Not set
        ...0 0... = Prf (Default Router Preference): Medium (0)
        .... .0.. = Proxy: Not set
        .... ..0. = Reserved: 0
    Router lifetime (s):
    Reachable time (ms): 0
    Retrans timer (ms): 0
    ICMPv6 Option (Source link-layer address : 58:ac:78:93:da:00)
        Type: Source link-layer address (1)
        Length: 1 (8 bytes)
        Link-layer address: 58:ac:78:93:da:00 (58:ac:78:93:da:00)
    ICMPv6 Option (MTU : 1500)
        Type MTU (5)
        Length: 1 (8 bytes)
        Reserved
        MTU: 1500
    ICMPv6 Option (Prefix information : 2001:db8:cafe:1::/64)
        Type: Prefix information (3)
        Length: 4 (32 bytes)
        Prefix Length: 64
        Flag: 0xc0
            1... .... = On-link flag(L): Set
            .1.. .... = Autonomous address-configuration flag(A): Set
            ..0. .... = Router address flag(R): Not set
            ...0 0000 = Reserved: 0
        Valid Lifetime: 2592000
        Preferred Lifetime: 604800
        Reserved
        Prefix: 2001:db8:cafe:1:: (2001:db8:cafe:1::)
```

Configuring a Router as a Stateless DHCPv6 Server

Configuring a router as a stateless DHCPv6 server is simple and straightforward. Two basic steps are used to configure stateless DHCPv6 services:

Step 1. Configure a DHCPv6 server pool name with configuration parameters.

Step 2. Enable the DHCPv6 server pool on an interface.

Table 10-1 shows the commands used in the first step to create the DHCPv6 pool and specifies some of the configuration parameters available.

Table 10-1 *Stateless DHCPv6 Configuration Pool Commands*

Command	Description
Router(config)# **ipv6 dhcp pool** *poolname*	Creates a DHCPv6 pool and enters DHCPv6 pool configuration mode.
Router(config-dhcp)# **dns-server** *ipv6-address*	Specifies the IPv6 DNS servers available to a DHCPv6 client.
Router(config-dhcp)# **domain-name** *domain*	Configures a domain name for a DHCPv6 client.

Note For complete DHCPv6 configuration options and commands, refer to the chapter "Implementing DHCP for IPv6" in *Cisco IPv6 Implementation Guide*, at www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-dhcp.html.

In the second step, you enable the DHCPv6 service on an interface using the **ipv6 dhcp server** interface command. Table 10-2 shows the commands used to associate a DHCPv6 pool with an interface. The rapid-commit option with the **ipv6 dhcp server** command is discussed in the next section.

Table 10-2 *Associating the DHCPv6 Pool to an Interface*

Command	Description
Router(config)# **interface** *type number*	Specifies an interface type and number and places the router in interface configuration mode.
Router(config-if)# **ipv6 dhcp server** *poolname* [**rapid-commit**]	Enables DHCPv6 service on an interface. *poolname* (optional) is a user-defined name for the local prefix pool. The pool name can be a symbolic string (such as Engineering) or an integer (such as 0). **rapid-commit** (optional) enables the use of the two-message exchange for address allocation and other configuration. If it is enabled, the client includes the rapid-commit option in a solicit message.

Example 10-3 shows the commands to configure router R1 as a stateless DHCPv6 server, with the *poolname* highlighted.

Example 10-3 *Stateless DHCPv6 Configuration on R1*

```
! Configure the stateless DHCPv6 server pool
R1(config)# ipv6 dhcp pool STATELESS-DHCPv6
R1(config-dhcpv6)# dns-server 2001:db8:cafe:1::8888
R1(config-dhcpv6)# domain-name example.com
R1(config-dhcpv6)# exit

! Set the O flag to 1 and enable DHCPv6 service on the interface
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 nd other-config-flag
R1(config-if)# ipv6 dhcp server STATELESS-DHCPv6
R1(config-if)#
```

The **ipv6 dhcp pool STATELESS-DHCPv6** command creates the DHCPv6 pool STATELESS-DHCPv6 and enters DHCPv6 pool configuration mode. The **dns-server 2001:db8:cafe:1::8888** command specifies the address of the DNS server, and the **example.com** command configures a domain name for the DHCPv6 clients.

The interface command **ipv6 dhcp server STATELESS-DHCPv6** enables the DHCPv6 service on the client-facing interface and associates it with the pool STATELESS-DHCPv6. Also shown is the **ipv6 nd other-config-flag** command that is used to set the O flag to 1.

Note The **debug ipv6 dhcp** [**detail**] command can be used to display DHCPv6 messages and operations.

Verifying Stateless DHCPv6 on a Windows Client

Example 10-4 shows the **ipconfig /all** command on WinPC. Notice that the client has received all its necessary addressing and configuration information. The prefix and default gateway address were received from R1's Router Advertisement. (The Interface IDs were randomly generated.) Highlighted in the example are the address of the IPv6 DNS server, 2001:db8:cafe:1::8888, and the DNS suffix list (domain name), example.com. Both of these were obtained from a stateless DHCPv6 server, also router R1.

Example 10-4 *WinPC* **ipconfig /all** *Command*

```
WinPC> ipconfig /all
<output omitted for brevity>
Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Network Connection
   Physical Address. . . . . . . . . : 00-50-56-AF-97-68
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086
                                       (Preferred)
   Temporary IPv6 Address. . . . . . : 2001:db8:cafe:1:78bd:10b0:aa92:62c
                                       (Preferred)
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11(Preferred)
   Default Gateway . . . . . . . . . : fe80::1%11
   DHCPv6 IAID . . . . . . . . . . . : 234901590
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-1A-5F-DA-B3-00-50-56-8C-C0-45
   DNS Servers . . . . . . . . . . . : 2001:db8:cafe:1::8888
   Connection-specific DNS Suffix Search List : example.com
```

Also highlighted in the example are the DUID (DHCP Unique Identifier) and the IAID (Identity Association Identifier) of WinPC. Every DHCPv6 participant, client, and server needs to be uniquely identified. In DHCPv6 this identification consists of two parts:

- **DUID (DHCP Unique Identifier):** The DUID is the system identifier that uniquely identifies the device. Each DHCPv6 server and client has exactly one DUID. The DUID allows the clients and servers to identify each other.
- **IAID (Interface Association Identifier):** The IAID identifies a specific interface on the devices, the system. There was a time when you could assume that a host would have only one network interface. But times have changed, and devices may have multiple network interfaces, including an Ethernet NIC, a WLAN NIC, FireWire, Bluetooth, or a USB attachment to the network. Each interface on the DHCPv6 client or server is identified using an IAID.

Verifying the Router as a Stateless DHCPv6 Server

Using router R1 in Example 10-5, the **show ipv6 dhcp** and **show ipv6 dhcp interface** commands can be used to examine the DHCPv6 services. The **show ipv6 dhcp** command displays the DUID of the router. The **show ipv6 dhcp interface** command is used to display DHCP information on an interface, including the associated DHCP pools and whether the rapid-commit option is being used.

Example 10-5 *Verifying DHCPv6 Services on R1*

```
R1# show ipv6 dhcp
This device's DHCPv6 unique identifier(DUID): 0003000158AC7893DA00
R1#

R1# show ipv6 dhcp interface gigabitethernet 0/0
GigabitEthernet0/0 is in server mode
  Using pool: STATELESS-DHCPv6
  Preference value: 0
  Hint from client: ignored
  Rapid-Commit: disabled
R1#
```

DHCPv6 Options

Both stateless and stateful DHCPv6 include the following two options:

- **rapid-commit:** This option reduces the number of DHCPv6 messages from four to two.
- **relay agent:** This option enables access to DHCPv6 services on another network.

rapid-commit Option

By default, DHCPv6 messages exchanged between a client and server require four messages (SOLICIT, ADVERTISE, REQUEST, and REPLY). The rapid-commit option reduces this from four to two messages.

The rapid-commit option begins in the initial DHCPv6 SOLICIT message, with the client requesting the rapid-commit option. This informs the DHCPv6 server that it wants to shorten the exchange from four messages to two, as illustrated in Figure 10-3. If a server is enabled for the rapid-commit option, it responds with a REPLY message containing the rapid-commit option and commits the assigned addresses included in the REPLY. Therefore, only two messages are exchanged (SOLICIT and REPLY) instead of the normal four messages (SOLICIT, ADVERTISE, REQUEST, and REPLY).

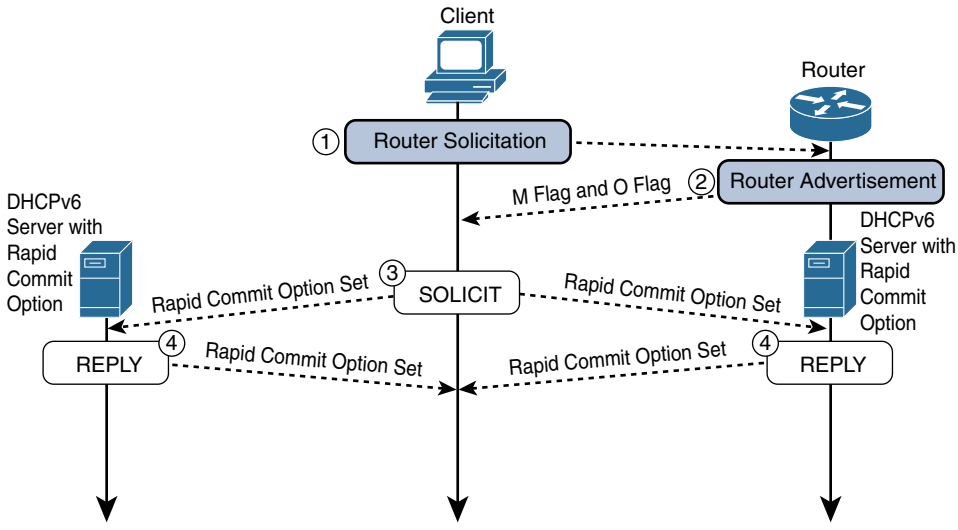


Figure 10-3 *DHCPv6 with Rapid-Commit Option*

When the rapid-commit option is used, the server does not expect nor receive any confirmation from the client that it has received the REPLY message. Therefore, if more than one server responds to a SOLICIT message that includes the rapid-commit option, all servers except one commit addresses that end up not being used by the client. Configuring only one DHCPv6 server with the rapid-commit option can minimize this issue of unused addresses. If a client sends a SOLICIT message with the rapid-commit option, it prefers REPLY messages from servers with the rapid-commit option set over messages from servers that do not have it set. If the client does not receive any REPLY messages with the rapid-commit option, it can accept the server's ADVERTISE message and proceed with the normal four-message exchange.

Configuring the Rapid-Commit Option

You configure a router with the rapid-commit option by including the **rapid-commit** parameter in the **ipv6 server dhcp server** interface command. In Example 10-6, R1's previous DHCPv6 configuration is modified to include the rapid-commit two-message exchange. The **show ipv6 dhcp interface** command is used to confirm that this option has been enabled.

Example 10-6 *Configuring and Verifying the Rapid-Commit Option*

```
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 dhcp server STATELESS-DHCPv6 rapid-commit
R1(config-if)# end
R1#
R1# show ipv6 dhcp interface gigabitethernet 0/0
GigabitEthernet0/0 is in server mode
  Using pool: STATELESS-DHCPv6
  Preference value: 0
  Hint from client: ignored
  Rapid-Commit: enabled
R1#
```

Example 10-7 shows the DHCPv6 and other relevant commands in R1's running config.

Example 10-7 *R1's Running Config*

```
R1# show running-config
!
ipv6 unicast-routing
!
ipv6 dhcp pool STATELESS-DHCPv6
 dns-server 2001:DB8:CAFE:1::8888
 domain-name example.com
!
interface GigabitEthernet0/0
 no ip address
 duplex auto
 speed auto
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:1::1/64
 ipv6 nd other-config-flag
 ipv6 dhcp server STATELESS-DHCPv6 rapid-commit
!
<output omitted for brevity>
```

Relay Agent Communications

At times, the DHCPv6 server is on a different network than the client requesting the addressing and other configuration parameters. Routers or relay agents can be configured to forward DHCPv6 messages between clients and servers on different networks. If you are familiar with DHCPv4 and Cisco routers, you are most likely familiar with the **ip helper-address** command used with DHCPv4. However, routers or relay agents forward DHCPv6 messages slightly differently than they do DHCPv4 messages.

Figure 10-4 illustrates the DHCPv6 process using a relay agent. From the perspective of the client, nothing changes. Although only one router is shown, more than one relay agent can be in the path between the client and the server, as shown in the following steps:

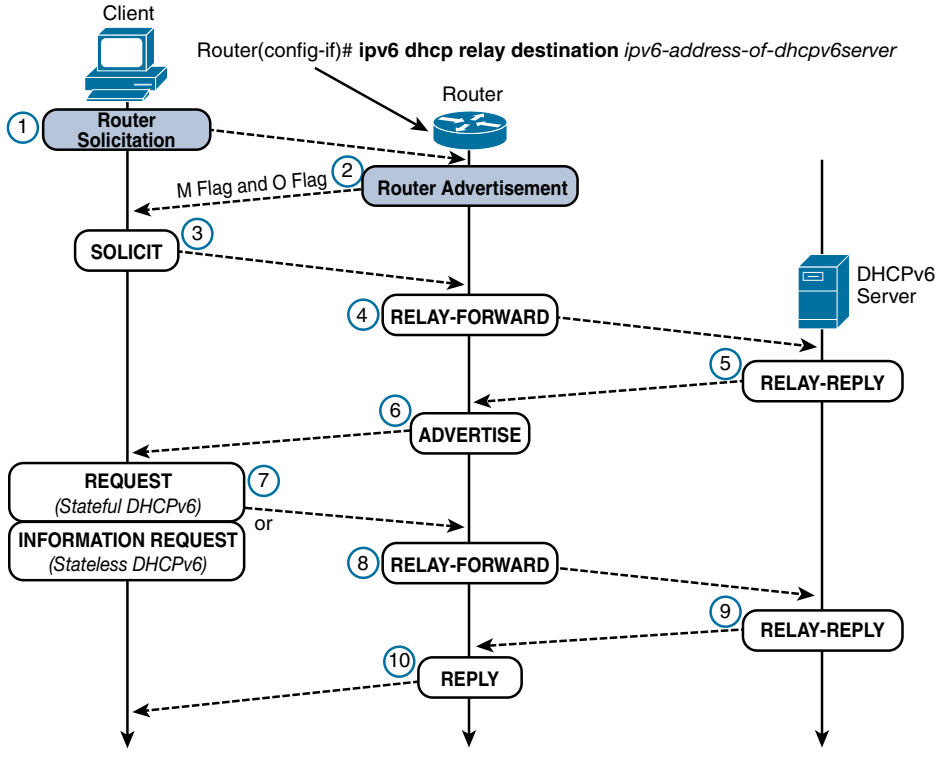


Figure 10-4 *DHCPv6 Relay Agent Communications*

Steps 1–3. These steps are the same as for the normal DHCPv6 communication process described previously. The client's SOLICIT message is sent to the multicast all-DHCP relay agents and servers address, ff02::1:2, which, as indicated by the name, includes both DHCPv6 servers and relay agents.

Step 4. The relay agent creates a RELAY-FORWARD message that contains the original SOLICIT message from the client and forwards the message to a server using the all-DHCP servers multicast address ff05::1:3 with site-local scope. The relay agent can also be configured to use a unicast address for the DHCPv6 server.

Step 5. The server returns a RELAY-REPLY message to a relay agent containing (encapsulating) the ADVERTISE message that the final relay agent uses to deliver to the client. (If the rapid-commit option is being used, the RELAY-REPLY contains a REPLY message with the required addressing and/or other configuration parameters.)

Step 6. The client-facing relay agent (there can be more than one relay agent in the path) receives the RELAY-REPLY. After decapsulating the message, it forwards the ADVERTISE message on to the client.

Step 7. The client sends a REQUEST or INFORMATION-REQUEST message to the server, depending on whether stateful or stateless DHCPv6 is being used. This is the same process as discussed in the previous section.

Step 8. The relay agent creates another RELAY-FORWARD message, this time containing (encapsulating) the REQUEST or INFORMATION-REQUEST message from the client, and forwards this message on to the server.

Step 9. The server responds with a RELAY-REPLY message to a relay agent containing the REPLY message.

Step 10. The relay agent receives the RELAY-REPLY. The client-facing relay agent decapsulates the message and forwards the REPLY message on to the client.

DHCPv6 Relay Agent Configuration Commands

To configure a router as a DHCPv6 relay agent, the **ipv6 dhcp relay destination** command is used on the client-facing interface. When the relay service is enabled on an interface, DHCPv6 messages received on that interface are forwarded to all configured relay destinations. The incoming DHCPv6 message can come from a client on that interface or can be relayed by another relay agent. Table 10-3 shows the syntax for enabling a router as a DHCPv6 relay agent.

Table 10-3 *DHCPv6 Relay Agent Commands*

Command	Description
Router(config)# **interface** *type number*	Specifies an interface type and number and places the router in interface configuration mode. This is the DHCPv6 client-facing interface.
Router(config-if)# **ipv6 dhcp relay destination** *ipv6-address* [*interface-type interface-number*]	Specifies a destination address to which client packets are forwarded and enables DHCPv6 relay service on the interface. The *ipv6-address* parameter defines the relay destination address. There are two types of relay destination address: ■ Link-scoped unicast or multicast IPv6 address. A user must specify an output interface for this kind of address. ■ Global or site-scoped unicast or multicast IPv6 address.

If the **ipv6 dhcp relay destination** command does not include an output interface; the IPv6 routing table determines the exit interface.

Configuring a Unicast DHCPv6 Relay Agent

Figure 10-5 shows the DHCPv6 client WinPC and the DHCPv6 server on separate networks.

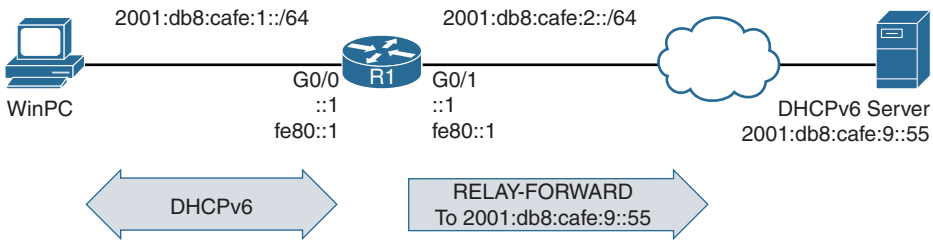


Figure 10-5 *Forwarding DHCPv6 Messages*

Example 10-8 demonstrates the configuration of router R1 as a DHCPv6 relay agent, using a unicast address. Notice that the example disables R1 as a DHCPv6 server with the command **no ipv6 dhcp server STATELESS-DHCPv6 rapid-commit**. Otherwise, IOS would issue the warning "% Interface is in DHCP server mode."

Example 10-8 *Configuring R1 as a DHCPv6 Relay Agent*

```
R1(config)# interface gigabitethernet 0/0
R1(config-if)# no ipv6 dhcp server STATELESS-DHCPv6 rapid-commit
R1(config-if)# ipv6 dhcp relay destination 2001:db8:cafe:9::55
R1(config-if)# end

R1# show ipv6 dhcp interface gigabitethernet 0/0
GigabitEthernet0/0 is in relay mode
  Relay destinations:
    2001:DB8:CAFE:9::55
R1#
```

The **show ipv6 dhcp interface gigabitethernet 0/0** command in Example 10-8 verifies that the interface is operating in relay mode and displays the address of the DHCPv6 server.

If the IPv6 address of the DHCPv6 server is a link-local address, an exit-interface is required following the destination address. Link-local addresses are not in the router's routing table, and the server may exist on any of the router's interfaces. For example:

```
R1(config-if)# ipv6 dhcp relay destination fe80::55 gigabitethernet 0/1
```

Configuring a DHCPv6 Relay Agent Using a Multicast Address

The previous example shows how to configure a relay agent to use the specific unicast address of the DHCPv6 server. Instead of using a unicast address, the relay agent can

be configured to use an all-DHCPv6 server multicast address, which can be either of the following:

- **ff02::1:2**—All-DHCPv6 server multicast address with *link-local scope*. The DHCPv6 messages are not routed beyond the relay link, and therefore the DHCPv6 server must be located on that segment.
- **ff05::1:3**—All-DHCPv6 server multicast address with *site-local scope*. In this case, the DHCPv6 messages can be routed beyond the relay link, but IPv6 multicast routing must be enabled using the **ipv6 multicast-routing** command.

The fourth digit in the multicast address represents the scope: 2 for link-local scope or 5 for site-local scope. Example 10-9 shows the configuration of R1 as a DHCPv6 relay agent using a multicast address with site-local scope.

Example 10-9 *Configuring R1 as a DHCPv6 Relay Agent Using Multicast*

```
R1(config)# ipv6 multicast-routing
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 dhcp relay destination ff05::1:3
R1(config-if)# end

R1# show ipv6 dhcp interface gigabitethernet 0/0
GigabitEthernet0/0 is in relay mode
  Relay destinations:
    FF05::1:3
R1#
```

Note There is much more to DHCPv6 that is beyond the scope of this book, including security considerations (for example, DHCPv6 spoofing) and configuring other client/server options. The list of references at the end of this chapter includes several RFCs and other resources pertaining to DHCPv6. These will help if you are interested in learning more about configuring DHCPv6 or the intricacies of the protocol itself.

Summary

This chapter focuses on stateless DHCPv6, which is one of three methods used for dynamic address allocation. The other two methods are SLAAC and stateful DHCPv6. Everything you learned in Chapter 9 regarding SLAAC also applies to SLAAC with stateless DHCPv6.

Implementing stateless DHCPv6 requires configuring the router's RA message to set the O flag (Other Configuration flag) to 1 and configuring a stateless DHCPv6 server.

The O flag set to 1 informs receiving devices that other configuration information is available from a stateless DHCPv6 server. The A flag (Address Autoconfiguration flag)

remains at its default setting of 1, so the receiving device continues to use SLAAC to create its GUA addresses. The M flag (Managed Address Configuration flag) stays set to 0, indicating that stateful DHCPv6 is not needed.

A Cisco router can be configured as a stateless DHCPv6 server. The **ipv6 dhcp pool** *poolname* creates a DHCPv6 pool and enters DHCPv6 pool configuration mode. The **dns-server** *ipv6-address* and **domain-name** *domain* commands are used to configure a list of DNS servers and domain names, respectively. The interface command **ipv6 dhcp server** *poolname* command is used to enable the DHCPv6 service to the interface.

The client uses SLAAC to create its global unicast address, obtain its default gateway address, and get other link parameters, such as the MTU and on-link prefix. Using stateless DHCPv6, the client can obtain other configuration information, such as a DNS server list or a domain name. The client has a single DUID (DHCP Unique Identifier) that uniquely identifies the device. Each interface has an IAID (Interface Association Identifier) that identifies a specific interface on the device.

Stateless and stateful DHCPv6 include two options: the rapid-commit option and the DHCPv6 relay agent. The rapid-commit option reduces the number of DHCPv6 messages from four to two. You use it by including the **rapid-commit** parameter in the **ipv6 server dhcp server** interface command.

When the DHCPv6 clients and server are on different networks, the router can act as a DHCPv6 relay agent. You can use the **ipv6 dhcp relay destination** command on the client-facing interface to forward all client DHCPv6 messages to the DHCPv6 server. The forwarding destination can be either the unicast address of a specific server or a multicast address. The all-DHCPv6 servers multicast address can be ff02::1:2 with link-local scope or ff05::1:3 with site-local scope. The site-local scope requires enabling IPv6 multicast with the **ipv6 multicast-routing** command.

Review Questions

1. SLAAC with stateless DHCPv6 requires which flags in the RA to be set to 1?
   a. A flag and M flag
   b. A flag and O flag
   c. O flag and M flag
   d. A flag only
   e. O flag only
   f. M flag only
2. Which flag or flags in the RA are set to 1 by default?
   a. A flag and M flag
   b. A flag and O flag
   c. O flag and M flag
   d. A flag only
   e. O flag only
   f. M flag only

3. What is the interface command for setting the O flag to 1?
4. Which of the following can be obtained from a stateless DHCPv6 server? (Choose two.)
   a. Global unicast address
   b. Default gateway address
   c. DNS server address
   d. Domain name
5. What system identifier uniquely identifies every DHCPv6 client and server?
6. What system identifier identifies a specific DHCPv6 interface?
7. Which DHCPv6 messages are sent between client and server when the rapid-commit option is used? (Choose two.)
   a. ADVERTISE
   b. REPLY
   c. REQUEST
   d. SOLCIT
8. When is a router configured as a DHCPv6 relay agent?
9. What is required in the configuration of the DHCPv6 relay agent when the link-local address of the DHCPv6 server is used?
10. Which two all-DHCPv6 server multicast addresses can be used in the **ipv6 dhcp relay destination** command?

References

RFCs

RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, R. Droms, Cisco Systems, www.ietf.org/rfc/rfc3315, July 2003.

RFC 3736, *Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6*, R. Droms, Cisco Systems, www.ietf.org/rfc/rfc3736, April 2004.

Websites

Cisco IPv6 Access Services: DHCPv6 Relay Agent, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_dhcp/configuration/xe-3s/dhcp-xe-3s-book/ip6-dhcp-rel-agent-xe.pdf.

IPv6 Implementation Guide, Cisco IOS Release 15.2M&T, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-dhcp.html#GUID-DB359FCB-5AEB-44AD-B2BD-3527A2148872.

Cisco IOS IPv6 Command Reference, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book.html.

Chapter 11

Stateful DHCPv6

Previous chapters discuss the basics of dynamic IPv6 address allocation and the first two methods of dynamically allocating address information: SLAAC and stateless DHCPv6. This chapter focuses on the third method, stateful DHCPv6. As a reminder, these are the three methods:

- **Method 1:** Stateless Address Autoconfiguration (SLAAC)
- **Method 2:** SLAAC and a stateless DHCPv6 server
- **Method 3:** Stateful DHCPv6 server

Unlike the first two methods, stateful DHCPv6 does not utilize SLAAC to generate a global unicast address. Stateful DHCPv6 is similar to the DHCP services provided for IPv4. A stateful DHCPv6 server provides IPv6 GUA addresses to clients and keeps track of (that is, maintains state for) which devices have been allocated which IPv6 addresses.

Note Although less common, a stateful DHCPv6 server can also provide ULA addresses to clients.

A significant difference between stateful DHCPv6 and DHCPv4 is the advertising of the default gateway address. In IPv4, the DHCPv4 server usually provides the default gateway address. In IPv6, only the router transmitting the ICMPv6 Router Advertisement can provide the address of the default gateway dynamically. There is no option within DHCPv6 to provide a default gateway address. Besides, there is no better device to provide this address than the router itself.

Note DHCPv6 is described in RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*.

The Prefix Delegation option for DHCPv6 (DHCPv6-PD) provides a method for delegating a globally routable IPv6 prefix from a service provider to a customer. Most customer IPv4 networks rely on NAT to translate from a private IPv4 address to a limited number of public IPv4 addresses. DHCPv6-PD provides the customer with more than enough global unicast address space to make NAT unnecessary. A customer may receive anywhere from 1 to 4.3 billion /64 subnets. This chapter discusses the process and examines sample configurations of both the service provider and customer routers.

Stateful DHCPv6 Messages and Process

Chapter 8, "Basics of Dynamic Addressing in IPv6," discusses DHCPv6 messages for both stateless and stateful DHCPv6. Figure 11-1 provides an overview of the stateful DHCPv6 messages and the steps used in this process.

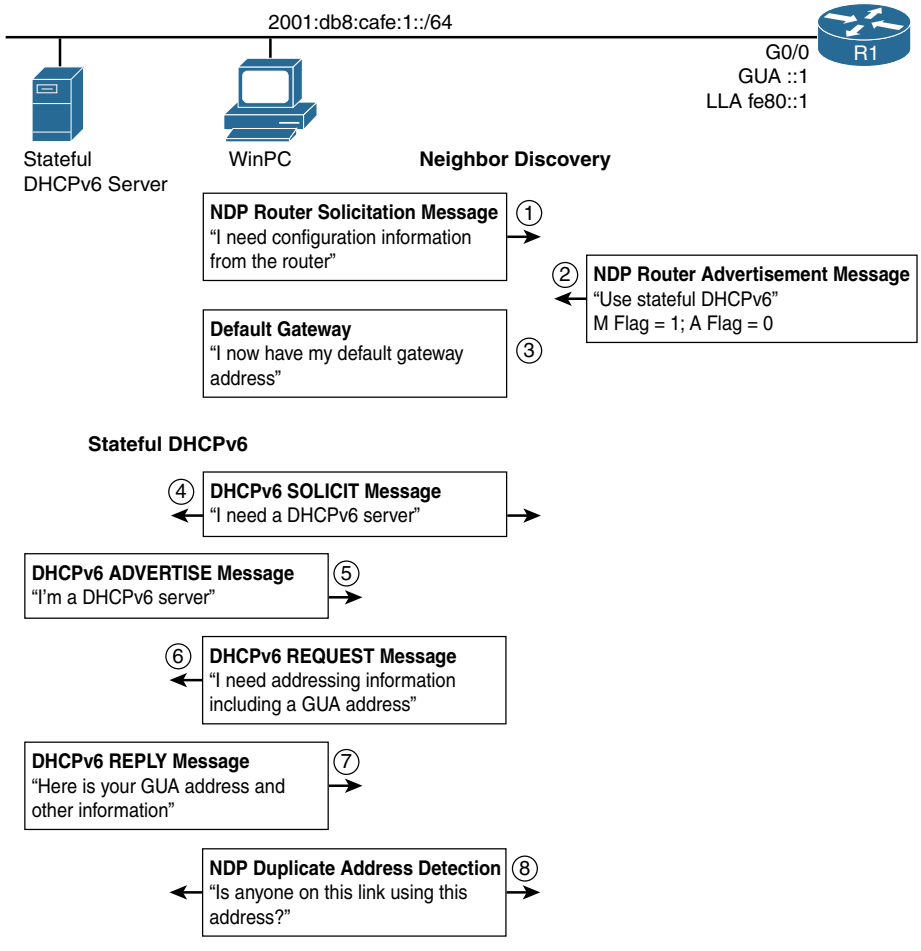


Figure 11-1 *Stateful DHCPv6 Operations*

As illustrated in Figure 11-1, these are the steps used for stateful DHCPv6:

Step 1. WinPC sends an ICMPv6 Router Solicitation message, requesting a Router Advertisement message.

Step 2. Router R1 sends the requested Router Advertisement message with the M flag (Managed Address Configuration flag) set to 1. This suggests to hosts that they use stateful DHCPv6 for addressing and other configuration information. The A flag (Address Autoconfiguration flag) is set to 0, informing hosts that SLAAC is not needed. (We discuss the significance of setting the A flag to 0 later in this chapter.)

The O flag (Other Configuration flag) is set to its default of 0, which is irrelevant because the M flag is set to 1. (The contents of the RA message are displayed using Wireshark in the next section.)

Step 3. Upon receiving the RA, WinPC uses the source IPv6 address of the RA, fe80::1, as its default gateway address. Because the A flag is set to 0, WinPC does not perform SLAAC.

Step 4. The RA message's M flag is set to 1, suggesting that addressing and other configuration information is available from a stateful DHCPv6 server. WinPC sends out a DHCPv6 SOLICIT message to ff02::1:2, the all-DHCPv6 servers multicast address, searching for DHCPv6 service.

Step 5. One or more DHCPv6 servers respond with a DHCPv6 ADVERTISE message, indicating that they are available for DHCPv6 service.

Step 6. WinPC responds to the selected server by sending a REQUEST message asking for addressing and other configuration information.

Step 7. The stateful DHCPv6 server responds with a REPLY message that contains a global unicast address and other configuration information.

Step 8. WinPC performs DAD on the address received from the stateful DHCPv6 server to ensure that this address is unique.

Note The details of DHCPv6 messages are discussed in Chapter 8.

Implementing Stateful DHCPv6

This section examines the process and configuration of implementing stateful DHCPv6. Implementing stateful DHCPv6 requires the following:

- Setting the router's RA message's M flag to 1 and the A flag to 0
- Configuring a stateful DHCPv6 server

Using the topology in Figure 11-2, the following sections demonstrate both tasks, including configuring router R1 as a stateful DHCPv6 server.

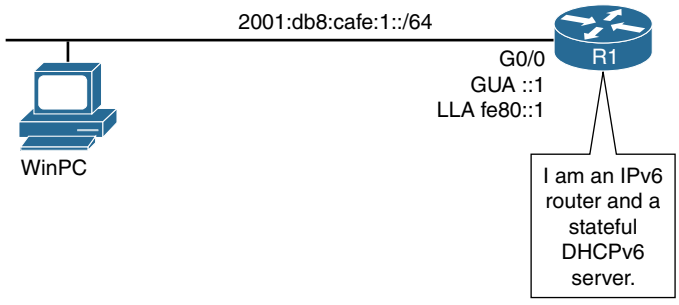


Figure 11-2 *Topology for Stateful DHCPv6 Configuration*

Configuring the RA Message M Flag and A Flag

By default, the M flag (Managed Address Configuration flag) is set to 0. To set the M flag to 1, use the interface command:

```
Router(config-if)# ipv6 nd managed-config-flag
```

The M flag suggests to the receiving host that it use the services of a stateful DHCPv6 server. The **no** option of this command sets this flag to its default of 0.

Setting the M Flag to 1 with an A Flag Set to 1

Before we modify the A flag (Autonomous Address Configuration flag) to 0, lets see what the effect is when you configure the M flag to 1 but leave the A flag at its default setting of 1.

Even though the M flag is set to 1, when the A flag is also set to 1, some operating systems, such as Windows and Ubuntu Linux, obtain their GUA address from the stateful DHCPv6 server but also use SLAAC to generate an additional address. This means the host will have *at least* two global unicast addresses—one from the stateful DHCPv6 server and another it creates for itself using SLAAC. Assuming that router R1 has been configured as a stateful DHCPv6 server, Example 11-1 shows that WinPC has three global unicast addresses:

- **2001:db8:cafe:1:d0f8:9ff6:4201:7086**—A GUA address created using SLAAC. This is a *public* address.
- **2001:db8:cafe:1:deed:3b2f:a6bc:ef77**—A GUA address provided by a stateful DHCPv6 server.
- **2001:db8:cafe:1:f8b6:2536:ce2c:c53a**—A GUA address created using SLAAC. This is a *temporary* address created because WinPC implements the privacy extension for SLAAC.

Example 11-1 *WinPC with GUA Addresses from SLAAC and Stateful DHCPv6*

```
WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  .  :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:deed:3b2f:a6bc:ef77
   Temporary IPv6 Address. . . . . . : 2001:db8:cafe:1:f8b6:2536:ce2c:c53a
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
   Default Gateway . . . . . . . . . : fe80::1
<output omitted for brevity>
```

Many, if not most, environments that implement stateful DHCPv6 have an addressing need or policy whereby devices obtain their address only from the stateful DHCPv6 server. It is common in these networks to use the services of a *stateful* DHCP server in order to manage and track IP addresses on the network. In these cases, you don't want devices arbitrarily creating their own IP addresses. Therefore, you want to eliminate the SLAAC process by setting the A flag to 0.

You might think that preventing the router from sending the RA message on the interface or removing the prefix from the RA message will prevent the host from using SLAAC. Although this is true, you will see that there may be consequences that you didn't intend.

Note If the A flag is set to 1 and the host uses SLAAC to create additional GUA addresses, DAD is performed on these addresses to ensure that they are unique.

Consequences of Disabling the RA Message or Omitting the Prefix

In Example 11-1, WinPC has obtained an address from the stateful DHCPv6 server but has also created two additional GUA addresses for itself by using SLAAC. To prevent the device from using SLAAC, best practice is to set the A flag to 0. You will see how to do this in a moment.

Why not just disable router R1 from sending the RA message on its G0/0 interface? Remember that the RA message provides two important pieces of information for devices that can't be obtained dynamically from a DHCPv6 server:

- **Default gateway address:** This is the address used to forward all packets with a destination IPv6 address on another link or subnet (off-link). Without this information, devices cannot communicate with devices that are on a remote network.
- **On-link prefix:** The L flag (On-Link flag) indicates that a specific prefix is on this link or subnet. Without this information, all packets are sent to the default gateway.

As you can see, disabling the RA message can have undesired consequences. What happens if you send the RA message but omit the prefix so the device receives the default gateway but doesn't use SLAAC?

Sending the RA message but omitting the prefix can also cause problems because of the relationship between the prefix and the L flag. If the prefix is omitted from the RA message, the device cannot receive on-link prefix information, resulting in all packets being sent to the default gateway. When the prefix is sent in the RA message, by default the L flag is set to 1, indicating to devices that this prefix is on-link, on their local subnet. Other devices that use this prefix can be reached directly without sending these packets to the router (default gateway).

If the prefix is omitted along with its associated L flag, devices send all packets to the router. For any packets that the router receives and sends back out the same interface (destinations that are on the same link as the source, the result of omitting the prefix and its associated L flag), the router sends a redirect message back to the source, informing the host that it can reach the destination directly. This can have an additional ill effect of having the router generate a large number of redirect messages on the network.

In short, you should not disable the RA message on an interface or omit the prefix from being advertised unless there are specific reasons to do so and you clearly understand the consequences.

Note Chapter 9 further discusses the L flag and suppressing the RA message.

Setting the M Flag to 1 and Modifying the A Flag to 0

As mentioned earlier, the best practice (or at least common practice) when implementing stateful DHCPv6 is to set the M flag to 1 *and* set the A flag to 0.

You have already seen how to set the M flag to 1, suggesting that the host use the services of a stateful DHCPv6 server. To set the A flag (Address Autoconfiguration flag) to 0, use this command:

```
Router(config-if)#  ipv6 nd prefix ipv6-prefix/prefix-length no-autoconfig
```

Example 11-2 shows the configuration of R1's G0/0 interface with the M flag set to 1 and the A flag set to 0. The **show ipv6 interface gigabitethernet 0/0** command is used to verify the change in the RA message.

Example 11-2 *Configuring R1's G0/0 Interface M Flag to 1 and A Flag to 0*

```
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 nd managed-config-flag
R1(config-if)# ipv6 nd prefix 2001:db8:cafe:1::/64 no-autoconfig
R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::1
  No Virtual link-local address(es):
```

```
 Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
 Joined group address(es):
    FF02::1
    FF02::2
    FF02::D
    FF02::16
    FF02::FB
    FF02::1:2
    FF02::1:FF00:1
    FF05::1:3
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  Output features: MFIB Adjacency
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 30000)
  ND advertised reachable time is 0 (unspecified)
  ND advertised retransmit interval is 0 (unspecified)
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  ND advertised default router preference is Medium
  Hosts use DHCP to obtain routable addresses.
R1#
```

This line of the output in the **show ipv6 interface gigabitethernet 0/0** command indicates how hosts will obtain their addressing information:

```
Hosts use DHCP to obtain routable addresses.
```

It indicates that the M flag is set to 1, suggesting to receiving devices that they use a stateful DHCPv6 server for its global unicast address and other configuration information.

The output from R1's **debug ipv6 nd** command shown in Example 11-3 verifies that the M flag has been set to 1 and the A flag has been set to 0. The highlighted "Managed address" indicates that the M flag has been enabled (set to 1). The absence of the A flag ([A]) in the output indicates that it has been disabled (set to 0).

Example 11-3 *Output from R1's* **debug ipv6 nd** *Command*

```
R1# debug ipv6 nd
  ICMP Neighbor Discovery events debugging is on
*Dec 28 20:02:35.490: ICMPv6-ND: (GigabitEthernet0/0,FE80::1) send RA to FF02::1
*Dec 28 20:02:35.490: ICMPv6-ND:   Managed address
*Dec 28 20:02:35.490: ICMPv6-ND: (GigabitEthernet0/0,FE80::1) Sending RA (1800/M) to
                                 FF02::1
*Dec 28 20:02:35.490: ICMPv6-ND:   MTU = 1500
*Dec 28 20:02:35.490: ICMPv6-ND:   prefix 2001:DB8:CAFE:1::/64 [L]
                                   2592000/604800
R1# undebug all
```

Wireshark Analysis of Router Advertisement: Stateful DHCPv6

The Wireshark analysis of R1's ICMPv6 Router Advertisement message in Example 11-4 confirms the output in the previous debug output. This is similar to the RA message presented in previous chapters. Notice that the M flag is set to 1 and the A flag is set to 0, both highlighted in the example.

Also notice in the RA message in Example 11-4 that the prefix 2001:db8:cafe:1:: is included, and the On-Link (L flag) is set to 1.

The ICMPv6 Router Advertisement message is encapsulated in an IPv6 header (not shown in Example 11-4):

- **Source Address:** fe80::1 (This is the link-local address of R1.)
- **Destination Address:** ff02::1 (This is an all-IPv6 devices multicast group or can also be a solicited unicast.)
- **Next Header:** 0x3a (Next Header is an ICMPv6 header, 58 in decimal.)

Example 11-4 *Wireshark Analysis of R1's Router Advertisement*

```
Internet Control Message Protocol v6
    Type: Router Advertisement (134)
    Code: 0
    Checksum: 0x796a [correct]
    Cur hop limit: 64
    Flags: 0xc0
        1... .... = Managed address configuration: Set
        .0.. .... = Other configuration: Not set
        ..0. .... = Home Agent: Not set
        ...0 0... = Prf (Default Router Preference): Medium (0)
        .... .0.. = Proxy: Not set
        .... ..0. = Reserved: 0
```

```
Router lifetime (s): 1800
Reachable time (ms): 0
Retrans timer (ms): 0
ICMPv6 Option (Source link-layer address : 58:ac:78:93:da:00)
     Type: Source link-layer address (1)
     Length: 1 (8 bytes)
     Link-layer address: 58:ac:78:93:da:00 (58:ac:78:93:da:00)
 ICMPv6 Option (MTU : 1500)
     Type MTU (5)
     Length: 1 (8 bytes)
     Reserved
     MTU: 1500
 ICMPv6 Option (Prefix information : 2001:db8:cafe:1::/64)
     Type: Prefix information (3)
     Length: 4 (32 bytes)
     Prefix Length: 64
     Flag: 0xc0
          1... .... = On-link flag(L): Set
          .0.. .... = Autonomous address-configuration flag(A): Not set
          ..0. .... = Router address flag(R): Not set
          ...0 0000 = Reserved: 0
     Valid Lifetime: 2592000
     Preferred Lifetime: 604800
     Reserved
     Prefix: 2001:db8:cafe:1:: (2001:db8:cafe:1::)
```

Configuring a Router as a Stateful DHCPv6 Server

The steps to configure a Cisco IOS router as a stateful DHCPv6 server are similar to the steps for configuring it as a stateless DHCPv6 server:

Step 1. Configure a DHCPv6 server pool name with configuration parameters.

Step 2. Enable the DHCPv6 server pool on an interface.

Table 11-1 shows the commands used in the first step to create the DHCPv6 pool and specifies some of the configuration parameters available. The difference in configuration between a stateful and stateless DHCPv6 server is the addition of the **address prefix** command. This command will be examined more closely later in this section.

Table 11-1 *Stateless DHCPv6 Configuration Pool Commands*

Command	Description
Router(config)# **ipv6 dhcp pool** *poolname*	Creates a DHCPv6 pool and enters DHCPv6 pool configuration mode.
Router(config-dhcp)# **address prefix** *ipv6-prefix/prefix-length [* ***lifetime*** *{ valid-lifetime preferred-lifetime |* ***infinite*** *}]*	Specifies the IPv6 prefix that will be used to allocate IPv6 addresses. **lifetime** (optional) specifies a time interval (in seconds) that an IPv6 address prefix remains in the valid state. If the **infinite** keyword is specified, the time interval does not expire.
Router(config-dhcp)# **dns-server** *ipv6-address*	Specifies the IPv6 DNS servers available to a DHCPv6 client.
Router(config-dhcp)# **domain-name** *domain*	Configures a domain name for a DHCPv6 client.

Note For complete DHCPv6 configuration options and commands, refer to the chapter "Implementing DHCP for IPv6" in *Cisco IPv6 Implementation Guide*, at www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-dhcp.html.

In the second step, you enable the DHCPv6 service on an interface by using the **ipv6 dhcp server** interface command. This is the same command you used in Chapter 10, "Stateless DHCPv6," for stateless DHCPv6. Table 11-2 shows the commands used to associate a DHCPv6 pool with an interface.

Table 11-2 *Associating the DHCPv6 Pool with an Interface*

Command	Description
Router(config)# **interface** *type number*	Specifies an interface type and number and places the router in interface configuration mode.
Router(config-if)# **ipv6 dhcp server** *poolname* [**rapid-commit**]	Enables DHCPv6 service on an interface. *poolname* (optional) is a user-defined name for the local prefix pool. The pool name can be a symbolic string (such as Engineering) or an integer (such as 0). **rapid-commit** (optional) allows the two-message exchange method for prefix delegation.

The commands to configure router R1 as a stateful DHCPv6 server are shown in Example 11-5, with the *poolname* highlighted. Notice that we have used a *poolname* that describes the type of DHCPv6 service.

Example 11-5 *Stateful DHCPv6 Configuration on R1*

```
! Configure the stateful DHCPv6 server pool
R1(config)# ipv6 dhcp pool STATEFUL-DHCPv6
R1(config-dhcpv6)# address prefix 2001:db8:cafe:1:deed::/80
R1(config-dhcpv6)# dns-server 2001:db8:cafe:1::8888
R1(config-dhcpv6)# domain-name example.com
R1(config-dhcpv6)# exit

! Set the M flag to 1, the A flag to 0 and enable DHCPv6 service on the interface
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 nd managed-config-flag
R1(config-if)# ipv6 nd prefix 2001:db8:cafe:1::/64 no-autoconfig
R1(config-if)# ipv6 dhcp server STATEFUL-DHCPv6
R1(config-if)#
```

The **ipv6 dhcp pool STATEFUL-DHCPv6** command creates the DHCPv6 pool STATEFUL-DHCPv6 and enters DHCPv6 pool configuration mode. The *poolname* is just a name and doesn't indicate whether the router is acting as a stateful or stateless server. The **address prefix** command is the command that causes the router to be a stateful DHCPv6 server and allocate IPv6 addresses. This command is discussed in the next section. Notice that the **address prefix** command uses an /80 prefix length in this example. The prefix length indicates the number of available addresses in the pool. This command is explained in the next section.

The **dns-server 2001:db8:cafe:1::8888** command specifies the address of the DNS server, and the **example.com** command configures a domain name for the DHCPv6 clients.

The interface command **ipv6 dhcp server STATEFUL-DHCPv6** enables the DHCPv6 service on the client-facing interface and associates it with the pool STATEFUL-DHCPv6. The configuration of the M flag to 1 and the A flag to 0 are also shown.

The **Address Prefix** Command

If you are familiar with configuring a Cisco router as an IPv4 DHCP server, you might recall that IOS does not specify an IPv4 prefix. Cisco IOS implementation of DHCPv4 uses the prefix assigned to the router's interface as the prefix in the addresses allocated. If you want to exclude an IPv6 address from being included in the allocation, you use the **ip dhcp excluded-address** *start-IPv4-address end-IPv4-address* command. This ensures that the router doesn't allocate addresses that have been manually assigned to other devices, such as servers and printers.

Cisco IOS as a DHCPv6 server does things slightly differently. The **address prefix** *ipv6-prefix/prefix-length* command is used to implement stateful DHCPv6 services on the router. The *ipv6-prefix/prefix-length* specifically defines the prefix to be used in the allocation. The server dynamically assigns the remaining bits after the *prefix-length*.

For example, examine the following command:

```
R1(config-dhcpv6)# address prefix 2001:db8:cafe:1::/64
```

In this example, router R1 allocates addresses beginning with the prefix 2001:db8:cafe:1::/64. The remaining 64 bits are assigned by the IOS as a DHCPv6 server.

Unlike with DHCPv4, with DHCPv6 there is no mechanism to specifically indicate which address to exclude from the allocation. Instead, you need to be specific in the prefix used in the **address prefix** command. You can think of it this way:

- With an IOS router as a DHCPv4 server you specify which addresses to *exclude*, and all other addresses are *included*.
- With an IOS router as a DHCPv6 server you specify which addresses to *include*, and all other addresses are *excluded*.

Is there a possibility that using the **address prefix 2001:db8:cafe:1::/64** command will conflict with a manually assigned address? A /64 prefix gives you 18 quintillion addresses, so the chances of any conflicts are remote at best. DAD (Duplicate Address Detection) is performed on the unicast addresses to ensure their uniqueness. But it's still best to avoid any potential problems and at the same time make it easier to identify addresses assigned by the stateful DHCPv6 server.

Notice that Example 11-5 uses an /80 prefix:

```
R1(config-dhcpv6)# address prefix 2001:db8:cafe:1:deed::/80
```

This is done primarily to easily identify the addresses allocated from this DHCPv6 server and at the same time avoid even the slightest potential for conflict. Using the /80 prefix assigns addresses with the first 80 bits of 2001:db8:cafe:1:deed. The remaining 48 bits are dynamically assigned, ranging from 2001:db8:cafe:1:deed:0:0:0 to 2001:db8:cafe:1:deed:ffff:ffff:ffff.

Note The prefix length associated with the interface is still /64. The /80 only indicates the available pool of addresses.

As you can see, in DHCPv6 you specifically indicate the prefix, which can include the leading bits of the Interface ID. You will see how to verify the client addresses in the next section.

Verifying Stateful DHCPv6 on a Windows Client

The **ipconfig /all** command in Example 11-6 shows WinPC's address and other configuration information. Notice that there is a single IPv6 global unicast address with the first 80 bits of 2001:db8:cafe:1:deed, the /80 prefix configured in Example 11-5. The output also includes when the lease was obtained and when the lease expires. The A flag is set to 0, so WinPC does not generate any addresses using SLAAC.

Other information includes the IPv6 DNS server address 2001:db8:cafe:1::8888 and the DNS suffix list (domain name) example.com. The DHCPv6 IAID and DUID are also displayed, as discussed in Chapter 10.

Example 11-6 *WinPC* **ipconfig /all** *Command*

```
WinPC> ipconfig /all
<output omitted for brevity>
Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Network Connection
   Physical Address. . . . . . . . . : 00-50-56-AF-97-68
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:deed:3b2f:a6bc:ef77
                                       <Preferred>
   Lease Obtained  . . . . . . . . . : Wednesday, December 28, 2016, 2:00:38 PM
   Lease Expires . . . . . . . . . . : Friday, December 30, 2016, 9:57:16 AM
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11<Preferred>
   Default Gateway . . . . . . . . . : fe80::1%11
   DHCPv6 IAID . . . . . . . . . . . : 234901590
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-1A-5F-DA-B3-00-50-56-8C-C0-45
   DNS Servers . . . . . . . . . . . : 2001:db8:cafe:1::8888
   Connection-specific DNS Suffix Search List : example.com
```

Verifying the Router as a Stateful DHCPv6 Server

The **show ipv6 dhcp** and **show ipv6 dhcp interface** commands in Example 11-7 verify the DHCPv6 services on R1. The **show ipv6 dhcp** command displays the DUID of the router. The **show ipv6 dhcp interface** command is used to display DHCPv6 information on an interface, including the associated DHCPv6 pools and whether the rapid-commit option is being used.

Example 11-7 *Verifying DHCPv6 Services on R1*

```
R1# show ipv6 dhcp
This device's DHCPv6 unique identifier(DUID): 0003000158AC7893DA00
R1#

R1# show ipv6 dhcp interface gigabitethernet 0/0
GigabitEthernet0/0 is in server mode
  Using pool: STATEFUL-DHCPv6
  Preference value: 0
  Hint from client: ignored
  Rapid-Commit: disabled
R1#
```

The **show ipv6 dhcp pool** command in Example 11-8 displays the information provided by the router as a DHCPv6 server. It includes the DHCPv6 prefix and prefix length, along with the number of active clients. The single active client is WinPC. The address of the DNS server and domain name are also displayed.

Example 11-8 **show ipv6 dhcp pool** *Command*

```
R1# show ipv6 dhcp pool
DHCPv6 pool: STATEFUL-DHCPv6
  Address allocation prefix: 2001:DB8:CAFE:1:DEED::/80 valid 172800 preferred 86400
    (1 in use, 0 conflicts)
  DNS server: 2001:DB8:CAFE:1::8888
  Domain name: example.com
  Active clients: 1
R1#
```

The **show ipv6 dhcp binding** command in Example 11-9 displays the client bindings in the DHCPv6 server binding table. This DHCPv6 client information includes the following:

- **Client: fe80::d0f8:9ff6:4201:7086—**This is the link-local address of the DHCPv6 client. fe80::d0f8:9ff6:4201:7086 is WinPC's link-local address, shown in the **ipconfig /all** command in Example 11-6.
- **DUID: 000100011A5FDAB30050568CC045—**This is the client's DHCP Unique Identifier, used to uniquely identify the client. 000100011A5FDAB30050568CC045 is WinPC's DUID, shown in the **ipconfig /all** command in Example 11-6.
- **Address: 2001:db8:cafe:1:deed:3b24:a6bc:ef77—**This is the address assigned to the DHCPv6 client. 2001:db8:cafe:1:deed:3b24:a6bc:ef77 is WinPC's GUA address, shown in the **ipconfig /all** command in Example 11-6.

Example 11-9 **show ipv6 dhcp binding** *Command*

```
R1# show ipv6 dhcp binding
Client: FE80::D0F8:9FF6:4201:7086
  DUID: 000100011A5FDAB30050568CC045
  Username : unassigned
  VRF : default
  IA NA: IA ID 0x0E005056, T1 0, T2 0
    Address: 2001:DB8:CAFE:1:DEED:3B24:A6BC:EF77
            preferred lifetime 86400, valid lifetime 172800
            expires at Dec 30 2016 03:59 PM (154342 seconds)
R1#
```

DHCPv6 Options

Much like stateless DHCPv6, stateful DHCPv6 includes the following two options:

- **rapid-commit:** This option reduces the number of DHCPv6 messages from four to two. By default, a DHCPv6 message exchanged between a client and a server requires four messages (SOLICIT, ADVERTISE, REQUEST, and REPLY). The rapid-commit option reduces this from four to two messages (SOLICIT and REPLY). The client sends a SOLICIT message to the all-DHCP relay agents and servers multicast address, requesting the assignment of addresses and other configuration information. This message includes an indication that the client is willing to accept an immediate REPLY message from the server and commit to the addresses provided. This is done by adding the **rapid-commit** parameter in the **ipv6 server dhcp server interface** command.
- **relay agent:** This option enables access to DHCPv6 services on another network. Much like the **ip helper** command used in DHCPv4, the **ipv6 dhcp relay destination** command forwards DHCPv6 messages from the client-facing interface toward the DHCPv6 server on another network.

Both the rapid-commit option and relay agent option are discussed in Chapter 10.

Note For more information on deploying DHCPv6 and host roles in an IPv6 environment, I highly recommend the excellent Cisco Press LiveLessons video series *IPv6 Design and Deployment LiveLessons*, by Tim Martin (www.ciscopress.com/store/ipv6-design-and-deployment-livelessons-9780134655512).

IPv6 Prefix Delegation Options for DHCPv6

In the world of IPv4, most internal networks use a private IPv4 address space for internal devices and NAT at the edge router to translate an address to a globally routable public IPv4 address. This is a common mechanism for most home networks and is partly responsible for keeping IPv4 alive for so many years. Figure 11-3 illustrates the common use of NAT on a home network.

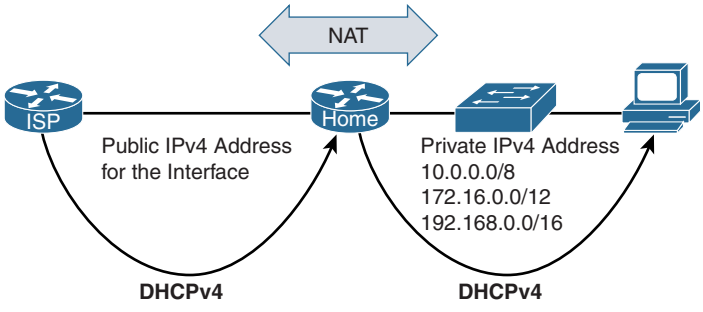


Figure 11-3 *NAT and Private IPv4 Address Space on a Home Network*

Avoiding the complications and problems with address translation, IPv6 uses a different technique. One of the methods IPv6 uses is DHCPv6 with the Prefix Delegation option (sometimes referred to as DHCPv6-PD or DHCPv6 Prefix Delegation), which provides a mechanism for automated delegation of a globally routable IPv6 prefix from a provider's router to a customer's router using DHCPv6. DHCPv6-PD is described in RFC 3633, *IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) Version 6*.

As shown in Figure 11-4, there are two routers involved in this process:

- **Requesting Router (RR):** This is the router that acts as the DHCPv6 client, requesting the prefix(es) to be assigned.
- **Delegating Router (DR):** This is the router that acts as the DHCPv6 server, responding to the requesting router's IPv6 prefix request.

Figure 11-4 shows the DHCPv6 message exchange between the Requesting Router and Delegating Router. Although this is similar to a stateful DHCPv6 message exchange, you can see differences in the SOLICIT and REPLY messages.

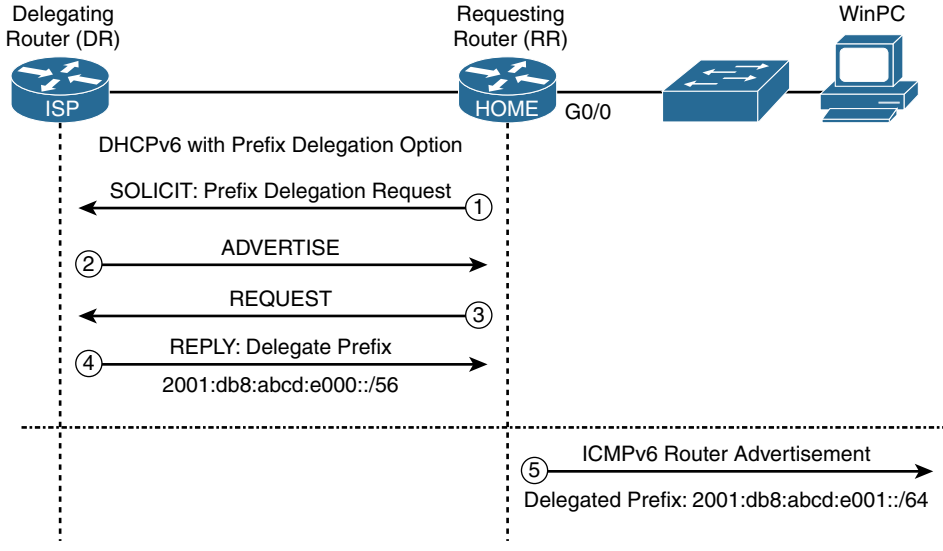


Figure 11-4 *DHCPv6 with Prefix Delegation Option*

The Requesting Router, the HOME router, initiates DHCPv6-PD in its SOLICIT message by including a request for an IPv6 prefix. The REPLY message from the Delegating Router, the ISP router, includes the IPv6 prefix. This is the prefix the Requesting Router can use for its own internal network—a prefix that the HOME router can use to allocate

addresses to its clients. If the prefix delegated from the DR is shorter than a /64, the RR can allocate multiple subnets from the delegated prefix.

In Figure 11-4, the RR router includes the Prefix Delegation option in its initial DHCPv6 SOLICIT message. The DR includes the prefix 2001:db8:abcd:e000::/56 in its REPLY message back to the RR. This prefix is for the RR to use for its own internal networks. Because the prefix delegated is a /56, this allows for an 8-bit Subnet ID that the RR can use to create 256 individual /64 subnets, as shown in Figure 11-5. The individual /64 prefixes can then be advertised by the HOME router in its Router Advertisement messages on its client-facing interfaces. In this example, the HOME router advertises the prefix 2001:db8:abcd:e001::/64 in an RA message out its G0/0 interface.

RFC 6177, *IPv6 Address Assignment to End Sites*, recommends /56 prefixes for many end sites, including home networks. As shown in Figure 11-5, a /56 prefix provides the end site an 8-bit Subnet ID, which allows for 256 /64 subnets.

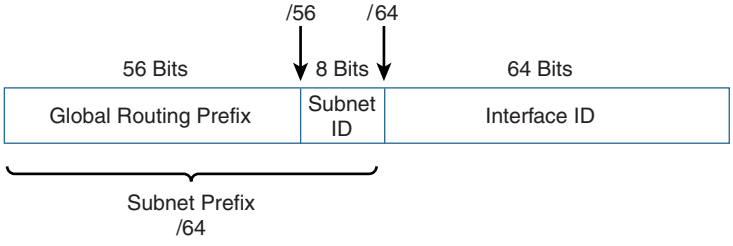


Figure 11-5 */56 Global Routing Prefix and /64 Subnet Prefix*

Sample Configuration: Prefix Delegation with DHCPv6

This section examines sample configurations for both the Requesting Router (HOME) and the Delegating Router (ISP). First, let's look at the process both routers use to distribute the addressing information.

Note Stateful DHCPv6 can also be used on the Requesting Router to assign addresses to downstream clients.

DHCPv6-PD Process

Figure 11-6 displays the topology used for the sample configuration of DHCPv6-PD. The figure illustrates the sequence in which addressing information is distributed using DHCPv6-PD, SLAAC, and stateless DHCPv6.

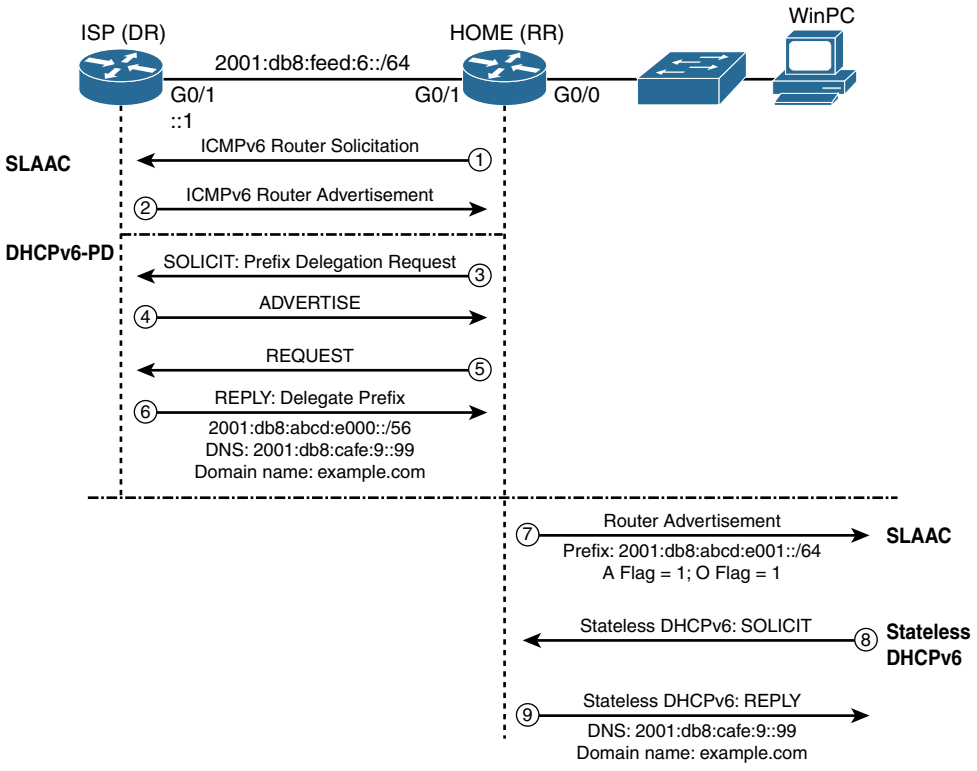


Figure 11-6 *Sequence of Addressing Information Using DHCPv6-PD, SLAAC, and Stateless DHCPv6*

This is the process by which addressing information is distributed using the sample configuration:

SLAAC (HOME Router)

Step 1. The HOME router sends an ICMPv6 Router Solicitation message to the ISP router.

Step 2. The ISP router returns an ICMPv6 Router Advertisement message. Upon receiving the RA message, the HOME router uses SLAAC and EUI-64 to assign the 2001:db8:feed:6:5aac:78ff:fe93:da01 to its ISP-facing G0/1 interface. Later in this section, the **show ipv6 interface brief** command in Example 11-12 displays the address.

DHCPv6 with Prefix Delegation Option

Step 3. The HOME router is configured to initiate DHCPv6-PD and sends a DHCPv6 SOLICIT message with the Prefix Delegation option.

Steps 4 and 5. The ISP and HOME routers exchange ADVERTISE and REQUEST messages.

Step 6. The ISP router sends a REPLY message, including a delegated 2001:db8:abcd:e000::/56 prefix. This is the prefix the HOME router can use to allocate addresses for its own internal networks. The DNS address 2001:db8:cafe:9::99 and domain name example.com are also included in the REPLY. The HOME router uses this information in its stateless DHCPv6 services to clients, as described in step 9.

SLAAC (HOME LAN)

Step 7. After receiving the delegated prefix 2001:db8:abcd:e000::/56, the HOME router does the following:

- It uses the 2001:db8:abcd:e001::/64 for its GigabitEthernet 0/0 LAN.
- It assigns the IPv6 GUA address 2001:db8:abcd:e001::1/64 to its GigabitEthernet 0/0 interface.
- It sends an ICMPv6 Router Advertisement out its GigabitEthernet 0/0 interface with the prefix 2001:db8:abcd:e001::/64. Both the A flag (Address Autoconfiguration flag) and the O flag (Other Configuration flag) are set to 1.
- Because the A flag is set to 1, WinPC uses the 2001:db8:abcd:e001::/64 prefix and a randomized identifier to create the GUA address 2001:db8:abcd:e001:d0f8:9ff6:4201:7086. Because WinPC is enabled with the privacy extension, it also creates a temporary address using the same prefix 2001:db8:abcd:e001:7541:2c5c:13a4:2d2d. You will see the verification of these addresses later in this section, in Example 11-15.
- WinPC uses the RA message's source IPv6 address fe80::5aac:78ff:fe93:da00 as its default gateway, as shown in Example 11-5.

Stateless DHCPv6 (WinPC)

Step 8. The RA message sent from the HOME router and received by WinPC has the O flag set to 1. WinPC sends a DHCPv6 SOLICIT message, searching for a DHCPv6 server.

Note The ADVERTISE and INFORMATION REQUEST messages are not shown.

Step 9. The HOME router is configured as a stateless DHCPv6 server and responds with a DHCPv6 REPLY message to WinPC. The REPLY includes the DNS address 2001:db8:cafe:9::99 and domain name example.com, and also verified later in this section in Example 11-15. This is the information the HOME router received from ISP, using DHCPv6-PD in step 6.

HOME Router (Requesting Router) Configuration and Verification

Example 11-10 shows the configuration of HOME, the Requesting Router. The HOME router begins by using SLAAC to obtain an IPv6 address for its ISP-facing interface, G0/1. Then it initiates DHCPv6-PD to obtain a prefix and other configuration information

for its LAN-facing interface, G0/0. The HOME router is also configured as a stateless DHCPv6 server for its internal LAN networks.

Example 11-10 *HOME Router, Requesting Router Configuration*

```
HOME(config)# ipv6 unicast-routing
HOME(config)# interface gigabitethernet 0/1
HOME(config-if)# ipv6 address autoconfig default
HOME(config-if)# ipv6 dhcp client pd DHCPV6-PREFIX-FROM-ISP
HOME(config-if)# exit

HOME(config)# interface gigabitethernet 0/0
HOME(config-if)# ipv6 address DHCPV6-PREFIX-FROM-ISP 0:0:0:1::1/64
HOME(config-if)# ipv6 nd other-config-flag
HOME(config-if)# ipv6 dhcp server IPV6-STATELESS
HOME(config-if)# exit

HOME(config)# ipv6 dhcp pool IPV6-STATELESS
HOME(config-dhcpv6)# import dns-server
HOME(config-dhcpv6)# import domain-name
```

The HOME router configuration commands in Example 11-10 are described as follows:

Global Configuration Mode

- **ipv6 unicast-routing:** Enables the forwarding of IPv6 unicast packets and sending of ICMPv6 Router Advertisements on its interfaces.

GigabitEthernet 0/1 Interface

- **interface gigabitethernet 0/1:** Configures the GigabitEthernet 0/1 interface and enters interface configuration mode. G0/1 is the ISP-facing interface.
- **ipv6 address autoconfig default:** Enables automatic configuration of an IPv6 address using SLAAC on the G0/1 interface. The **default** keyword installs a default route in HOME's routing table, forwarding packets to ISP. HOME's IPv6 routing table is shown in Example 11-11.

  HOME uses the prefix in the RA message sent by ISP to create a GUA address on its ISP-facing G0/1 interface. Using the RA's 2001:db8:feed:6::/64 prefix and EUI-64, HOME creates the address 2001:db8:feed:6:5aac:78ff:fe93:da01 on the G0/1 interface, as shown in Example 11-12.
- **ipv6 dhcp client pd DHCPV6-PREFIX-FROM-ISP:** Enables the DHCPv6 client process and enables the request for Prefix Delegation through this interface. This command initiates the DHCPv6 Prefix Delegation process between HOME and ISP. It also uses the prefix received from the Delegating Router (ISP) for creating an address on a local interface. The prefix associated with the DHCPV6-PREFIX-

FROM-ISP pool is used in the LAN-facing G0/0 interface. The optional **rapid-commit** parameter can be used to reduce the number of messages from four to two.

GigabitEthernet 0/0 Interface

- **interface gigabitethernet 0/0:** Configures the GigabitEthernet 0/0 interface and enters interface configuration mode. G0/0 is the LAN-facing interface.
- **ipv6 address DHCPV6-PREFIX-FROM-ISP 0:0:0:1::1/64:** Configures an IPv6 address on the G0/0 interface defined in a general prefix learned from DHCPv6-PD. This is the information associated with the **ipv6 dhcp client pd DHCPV6-PREFIX-FROM-ISP** command. ISP delegates the prefix 2001:db8:abcd:e000::/56 to be used by HOME in its internal networks. The address 2001:db8:abcd:e001::1/64 is assigned to G0/0, shown in Example 11-12. Figure 11-7 illustrates how this general prefix is used to assign an address to the G0/0 interface.
- **ipv6 nd other-config-flag:** Sets the O flag (Other Configuration flag) to 1 in the ICMPv6 Router Advertisement message to suggest stateless DHCPv6 services.
- **ipv6 dhcp server IPV6-STATELESS:** Enables DHCPv6 service on the interface, using the information in the pool IPV6-STATELESS.

DHCPv6 Pool: IPV6-STATELESS

- **ipv6 dhcp pool IPV6-STATELESS:** Configures a DHCPv6 server information pool and enters DHCPv6 pool configuration mode. This is the information the HOME router uses in its stateless DHCPv6 services for its G0/0 LAN.
- **import dns-server:** Imports a DNS recursive name server option to a DHCPv6 client. The address of the DNS server was received during the DHCPv6-PD process.
- **import domain-name:** Imports a domain name option to a DHCPv6 client. The domain name was received during the DHCPv6-PD process.

Example 11-11 *HOME IPv6 Routing Table*

```
HOME# show ipv6 route
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, I1 - ISIS L1, I2 - ISIS L2
       IA - ISIS interarea, IS - ISIS summary, D - EIGRP, EX - EIGRP external
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, a - Application
ND  ::/0 [2/0]
     via FE80::FEED:1, GigabitEthernet0/1
S   2001:DB8:ABCD:E000::/56 [1/0]
     via Null0, directly connected
```

```
C   2001:DB8:ABCD:E001::/64 [0/0]
     via GigabitEthernet0/0, directly connected
L   2001:DB8:ABCD:E001::1/128 [0/0]
     via GigabitEthernet0/0, receive
NDp 2001:DB8:FEED:6::/64 [2/0]
     via GigabitEthernet0/1, directly connected
L   2001:DB8:FEED:6:5AAC:78FF:FE93:DA01/128 [0/0]
     via GigabitEthernet0/1, receive
L   FF00::/8 [0/0]
     via Null0, receive
HOME#
```

Example 11-11 shows the NDp (Neighbor Discovery prefix) entry for the prefix 2001:db8:feed:6::/64 in HOME's IPv6 routing table. This prefix was received from ISP using SLAAC and is used by the HOME router to create its GUA address on G0/1, its ISP-facing interface. This address is displayed in Example 11-12, using the **show ipv6 interface brief** command on the HOME router.

Example 11-12 *HOME Interface IPv6 Addresses*

```
HOME# show ipv6 interface brief
GigabitEthernet0/0      [up/up]
    FE80::5AAC:78FF:FE93:DA00
    2001:DB8:ABCD:E001::1
GigabitEthernet0/1      [up/up]
    FE80::5AAC:78FF:FE93:DA01
    2001:DB8:FEED:6:5AAC:78FF:FE93:DA01
HOME#
```

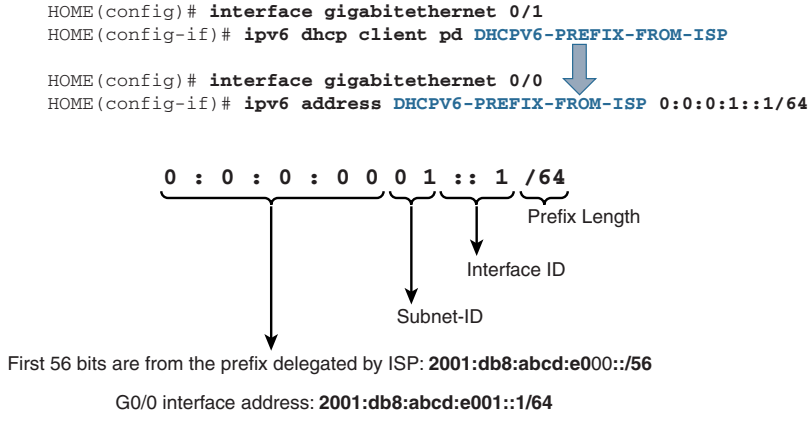


Figure 11-7 ipv6 address *General Prefix Command*

ISP Router (Delegating Router) Configuration and Verification

Example 11-13 shows the configuration of ISP, the Delegating Router. ISP sends an RA message on its HOME-facing interface that provides the information HOME needs to perform SLAAC. ISP is also the DHCPv6-PD Delegating Router and provides prefix and other configuration information for the HOME router's internal LAN networks.

Example 11-13 *ISP Delegating Router Configuration*

```
ISP(config)# ipv6 unicast-routing
ISP(config)# interface gigabitethernet 0/1
ISP(config-if)# ipv6 address 2001:db8:feed:6::1/64
ISP(config-if)# ipv6 address fe80::feed:1 link-local
ISP(config-if)# ipv6 dhcp server DHCPV6-CLIENT-ADDRESS
ISP(config-if)# exit

ISP(config)# ipv6 dhcp pool DHCPV6-CLIENT-ADDRESS
ISP(config-dhcpv6)# prefix-delegation pool DHCPV6-PD-POOL
ISP(config-dhcpv6)# dns-server 2001:db8:cafe:9::99
ISP(config-dhcpv6)# domain-name example.com
ISP(config-dhcpv6)# exit

ISP(config)# ipv6 local pool DHCPV6-PD-POOL 2001:db8:abcd:e000::/52 56
```

The ISP router configuration commands in Example 11-13 are described as follows:

Global Configuration Mode

- **ipv6 unicast-routing:** Enables the forwarding of IPv6 unicast packets and sending of ICMPv6 Router Advertisements on interfaces.

GigabitEthernet 0/1 Interface

- **interface gigabitethernet 0/1:** Configures the GigabitEthernet 0/1 interface and enters interface configuration mode. G0/1 is the HOME-facing interface.
- **ipv6 address 2001:db8:feed:6::1/64:** Configures a global unicast address on the interface.
- **ipv6 address fe80::feed:1 link-local:** Configures a link-local address on the interface.
- **ipv6 dhcp server DHCPV6-CLIENT-ADDRESS:** Enables DHCPv6 services on the interface, using the information in the pool DHCPV6-CLIENT-ADDRESS.

DHCPv6 Pool: DHCPV6-CLIENT-ADDRESS

- **ipv6 dhcp pool DHCPV6-CLIENT-ADDRESS:** Configures a DHCPv6 server information pool and enters DHCPv6 pool configuration mode. This pool contains the information the ISP router will use in its DHCPv6 services to the HOME router.
- **prefix-delegation pool DHCPV6-PD-POOL:** Specifies a named IPv6 local prefix pool from which prefixes are delegated to DHCPv6 clients. This command references

a prefix delegation pool **ipv6 local pool DHCPV6-PD-POOL** that defines how prefixes are delegated.

- **dns-server 2001:db8:cafe:9::99:** Specifies the address of the DNS server distributed by ISP as a DHCPv6 server.
- **domain-name example.com:** Specifies the domain name distributed by ISP as a DHCPv6 server.

Prefix Delegation Pool: DHCPV6-PD-POOL

- **ipv6 local pool DHCPV6-PD-POOL 2001:db8:abcd:e000::/52 56:** Configures a local prefix pool. ISP reserves the /52 prefix **2001:db8:abcd:e**000::**/52**, from which it allocates /56 prefixes for its DHCPv6-PD clients: **2001:db8:abcd:e0**00::/56 to **2001:db8:abcd:ef**00::/56. HOME receives the first /56 prefix from this pool, 2001:db8:abcd:e000::/56. This process is illustrated in Figure 11-8.

The **show ipv6 local pool** and **show ipv6 local pool DHCPV6-PD-POOL** commands in Example 11-14 verify that ISP has a total of 16 /56 prefixes that it can allocate to customers (15 free and 1 in use).

```
ISP(config)# ipv6 local pool DHCPV6-PD-POOL 2001:db8:abcd:e000::/52 56
```

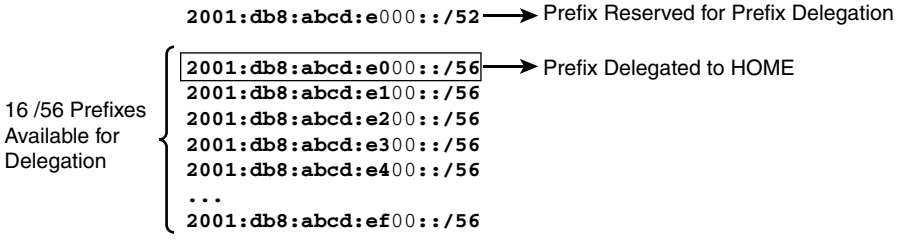


Figure 11-8 *ISP's Local Prefix Pool*

Example 11-14 *Verification of Address Pool on ISP*

```
ISP# show ipv6 local pool
Pool                 Prefix                        Free  In use
DHCPV6-PD-POOL       2001:DB8:ABCD:E000::/52         15       1
ISP#

ISP# show ipv6 local pool DHCPV6-PD-POOL
Prefix is 2001:DB8:ABCD:E000::/52 assign /56 prefix
1 entries in use, 15 available, 0 rejected
0 entries cached, 1000 maximum
User                 Prefix                        Interface
0003000158AC7893DA0000040001
                     2001:DB8:ABCD:E000::/56

ISP#
```

Verifying Prefix Delegation with DHCPv6 on WinPC

Final verification is done by examining the addressing information on WinPC using the **ipconfig /all** command in Example 11-15. WinPC is now configured with the following information, highlighted in the example:

- **IPv6 Address 2001:db8:abcd:e001:d0f8:9ff6:4201:7086—**This is the public IPv6 address generated using SLAAC. The /64 prefix is part of the /56 prefix initially distributed by ISP to HOME, using DHCPv6-PD. The HOME router advertised the 2001:db8:abcd:e001::/64 prefix in its RA message.
- **Temporary IPv6 Address 2001:db8:abcd:e001:7541:2c5c:13a4:2d2d—**This is the temporary IPv6 address generated using SLAAC. It is the same prefix used by WinPC's public IPv6.
- **Default Gateway fe80::5aac:78ff:fe93:da00—**This is the default gateway address obtained from the source IPv6 address of the HOME router's RA message.
- **DNS Servers 2001:db8:cafe:9::99—**This is the DNS server address received from the HOME router as a stateless DHCPv6 server.
- **Connection-specific DNS Suffix Search List example.com—**This is the domain name received from the HOME router as a stateless DHCPv6 server.

Example 11-15 *WinPC* **ipconfig /all** *Command*

```
WinPC> ipconfig /all
<output omitted for brevity>
Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : example.com
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Network Connection
   Physical Address. . . . . . . . . : 00-50-56-AF-97-68
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   IPv6 Address. . . . . . . . . . . : 2001:db8:abcd:e001:d0f8:9ff6:4201:7086
                                       (Preferred)
   Temporary IPv6 Address. . . . . . : 2001:db8:abcd:e001:7541:2c5c:13a4:2d2d
                                       (Preferred)
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11(Preferred)
   Default Gateway . . . . . . . . . : fe80::5aac:78ff:fe93:da00%11
   DHCPv6 IAID . . . . . . . . . . . : 234901590
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-1A-5F-DA-B3-00-50-56-8C-C0-45
   DNS Servers . . . . . . . . . . . : 2001:db8:cafe:9::99
   Connection-specific DNS Suffix Search List : example.com
```

Note For more information on DHCPv6 Prefix Delegation, see the Cisco Press book *Deploying IPv6 Networks* by Popoviciu, Levy-Abegnoli, and Grossetete. Again, for more information on deploying DHCPv6 and host roles in an IPv6 environment, see the excellent Cisco Press LiveLessons video series *IPv6 Design and Deployment LiveLessons* by Tim Martin (www.ciscopress.com/store/ipv6-design-and-deployment-livelessons-9780134655512).

Summary

This chapter focuses on stateful DHCPv6, a method similar to DHCP for IPv4. It also discusses the Prefix Delegation option for DHCPv6, which service providers typically use to allocate IPv6 addressing information to customers.

This chapter looks at the process and messages used for stateful DHCPv6. The router sends the ICMPv6 Router Advertisement message with the M flag (Managed Address Configuration flag) set to 1. This suggests to hosts that the global unicast address and other configuration information can be obtained from a stateful DHCPv6 server. The M flag is set to 1 using this interface command:

```
Router(config-if)# ipv6 nd managed-config-flag
```

However, unlike DHCP for IPv4, DHCPv6 does not provide the default gateway address. That address can only be provided dynamically from the source IPv6 address of the RA message.

The addressing policy in many *stateful* DHCPv6 environments is to only obtain the IPv6 address from a stateful DHCPv6 server. This allows for easier management and tracking of IPv6 addresses in the network. In these cases, it is important to ensure that devices aren't creating additional address using SLAAC. This can be accomplished by configuring the Router Advertisement's A flag (Autonomous Address Configuration flag) to 0, using the following interface command:

```
Router(config-if)#  ipv6 nd prefix ipv6-prefix/prefix-length no-autoconfig
```

This chapter discusses other options for preventing devices from using SLAAC, including omitting the prefix in the RA message and not sending the RA message at all. However, there can be unintended effects in doing either of these things. As mentioned previously, devices can only obtain the address of the default gateway from the Router Advertisement, so blocking RA messages from being sent can prevent a host from reaching other networks.

Omitting the prefix from the RA also omits the L flag (On-Link flag), which tells devices their local link or subnet. This has the effect of the host sending all packets to the router. The router forwards any packets local to the subnet back on the link, along with an ICMPv6 Redirect message. This could cause an undesirable number of Redirect messages on the network.

Configuration of a Cisco IOS router as a stateful DHCP6 server is similar to configuration of a stateless DHCPv6 server. The difference is the addition of the **address prefix** command in DHCPv6 pool configuration mode. This command specifies the prefix that will be used to create the address, with the rest of the address dynamically assigned by IOS.

Service providers can use the Prefix Delegation option for DHCPv6 (DHCPv6-PD) to allocate IPv6 prefixes and other configuration information to customers. There are two routers involved in this process:

- **Requesting Router (RR):** The router that acts as the DHCPv6 client, requesting the prefix(es) to be assigned
- **Delegating Router (DR):** The router that acts as the DHCPv6 server, responding to the requesting router's IPv6 prefix request

The DHCPv6-PD process is initiated by the client, the Requesting Router. The Delegating Router provides a prefix that the Requesting Router can use in its Router Advertisement messages on its internal networks. If the prefix is less than a /64, such as a /56, the Requesting Router can subnet the prefix for different /64 prefixes on multiple internal networks.

The router may also act as a stateless DHCPv6 server for information not provided in the Router Advertisement, such as the DNS server address or a domain name.

Review Questions

1. Which of a router's RA flag(s) must you modify from the default setting(s) to suggest to clients that they use stateful DHCPv6 and not create an additional address using SLAAC?
   a. A flag and M flag
   b. A flag an O flag
   c. O flag and M flag
   d. A flag only
   e. O flag only
   f. M flag only
2. What is the interface command to set the M flag to 1?
3. What is the interface command to set the A flag to 1?
4. When you omit the prefix from the Router Advertisement message, what additional effect occurs?

5. Which of the following DHCPv6 server pool commands would a router use to provide addresses with the first 80 bits of 2001:db8:face:b00c:1eaf::/80?
   a. address 2001:db8:face:b00c:1eaf::/80
   b. address prefix 2001:db8:face:b00c:1eaf::/80
   c. ipv6 address 2001:db8:face:b00c:1eaf::/80
   d. ipv6 dhcp address 2001:db8:face:b00c:1eaf::/80
6. Which of the following statements are true regarding stateful DHCPv6? (Choose two.)
   a. With an IOS router as a stateful DHCPv6 server you specify which addresses to *exclude*, and all other addresses are *included*.
   b. With an IOS router as a stateful DHCPv6 server you specify which addresses to *include*, and all other addresses are *excluded*.
   c. With an IOS router as a stateful DHCPv4 server you specify which addresses to *include*, and all other addresses are *excluded*.
   d. With an IOS router as a stateful DHCPv4 server you specify which addresses to *exclude*, and all other addresses are *included*.
7. Which router in DHCPv6-PD acts as the DHCPv6 client, requesting the prefix(es) to be assigned?
   a. Delegating Router
   b. Designated Router
   c. Prefix-Delegating Router
   d. Requesting Router
8. Which router in DHCPv6-PD acts as the DHCPv6 server, responding to the Requesting Router's IPv6 prefix request?
   a. Delegating Router
   b. Designated Router
   c. Prefix-Delegating Router
   d. Requesting Router
9. If the Requesting Router received the prefix 2001:db8:beef::/48, what would be its IPv6 address if it were configured with the interface command **ipv6 address MY-PREFIX 0:0:0:99::99/64**?
   a. 2001:db8:beef:99::1/48
   b. 2001:db8:beef:99::1/64
   c. 2001:db8:beef:99::99/48
   d. 2001:db8:beef:99::99/64
10. How many prefixes could a Delegating Router provide with a local prefix pool command **ipv6 local pool DHCPV6-PD-POOL 2001:db8:ab00::/40 48**? What would be the prefix length of the prefixes provided?
    a. 16 /40 prefixes
    b. 16 /48 prefixes
    c. 16 /64 prefixes

d. 256 /40 prefixes
e. 256 /48 prefixes
f. 256 /64 prefixes

The 256 prefixes the DR can provide range from 2001:db8:ab00::/48 to 2001:db8:abff::/48.

References

RFCs

RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, R. Droms, Cisco Systems, www.ietf.org/rfc/rfc3315, July 2003.

RFC 3633, *IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6*, O. Troan, Cisco systems, www.ietf.org/rfc/rfc3633, December 2003.

RFC 3736, *Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6*, R. Droms, Cisco Systems, www.ietf.org/rfc/rfc3736, April 2004.

RFC 6177, *IPv6 Address Assignment to End Sites*, T. Narten, IBM, tools.ietf.org/html/rfc6177, March 2011.

Websites

DHCPv6 Using the Prefix Delegation Feature Configuration Example, www.cisco.com/c/en/us/support/docs/ip/ip-version-6-ipv6/113141-DHCPv6-00.html.

IP Addressing: DHCP Configuration Guide, Cisco IOS XE Release 3S, Chapter: IPv6 Access Services: DHCPv6 Prefix Delegation, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_dhcp/configuration/xe-3s/dhcp-xe-3s-book/ip6-dhcp-prefix-xe.html.

Cisco IOS IPv6 Command Reference, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book.html.

Cisco Press LiveLessons: IPv6 Design and Deployment, by Tim Martin, www.ciscopress.com/store/ipv6-design-and-deployment-livelessons-9780134655512.

This page intentionally left blank

Part IV

ICMPv6 and ICMPv6 Neighbor Discovery

This page intentionally left blank

Chapter 12

ICMPv6

If you are familiar with Internet Control Message Protocol (ICMP) for IPv4, you will find ICMPv6 very similar; however, ICMPv6 is more than just ICMP for IPv6. ICMPv6 is a more robust protocol, containing new features and improving on similar functionality in ICMPv4. These enhancements to ICMPv6 are part of ICMPv6 Neighbor Discovery Protocol, which is discussed in Chapter 13, "ICMPv6 Neighbor Discovery."

This chapter focuses on the ICMPv6 protocol, the counterpart to ICMPv4. ICMPv6 is described in RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*.

ICMP is one of the core protocols of the TCP/IP suite of protocols. It is used by operating systems to send messages—either informational or error messages—between devices. ICMP is also used by various applications such as ping and traceroute to test network connectivity between two devices.

This chapter explores the general message format for ICMPv6, which uses similar Type and Code fields to those used in ICMPv4. This chapter examines the two types of ICMPv6 messages: error messages and informational messages.

These ICMPv6 error messages are similar to ICMPv4 error messages:

- Destination Unreachable
- Packet Too Big
- Time Exceeded
- Parameter Problem

The following ICMPv6 informational messages used by ping are also similar to those in ICMPv4:

- Echo Request
- Echo Reply

There are three ICMPv6 informational messages used for Multicast Listener Discovery (see RFC 2710 and RFC 3810), discussed in Chapter 7, "Multicast Addresses":

- Multicast Listener Query
- Multicast Listener Report
- Multicast Listener Done

Chapter 13 discusses ICMPv6 Neighbor Discovery, which includes address resolution (similar to ARP in IPv4), Duplicate Address Detection (DAD), and Neighbor Unreachability Detection (NUD). Neighbor Discovery uses the following ICMPv6 informational messages (see RFC 4861):

- Router Solicitation
- Router Advertisement
- Neighbor Solicitation
- Neighbor Advertisement
- Redirect

General Message Format

All ICMPv6 messages have the same general format, and the general format of an ICMPv6 message is similar to that of an ICMPv4 message. As shown in Figure 12-1, an IPv6 header with a Next Header value of 58 precedes every ICMPv6 message. (In IPv4, the Protocol field is set to 1 to indicate an ICMPv4 message.) The preceding header does not have to be the main IPv6 header. It could also be one of the IPv6 extension headers.

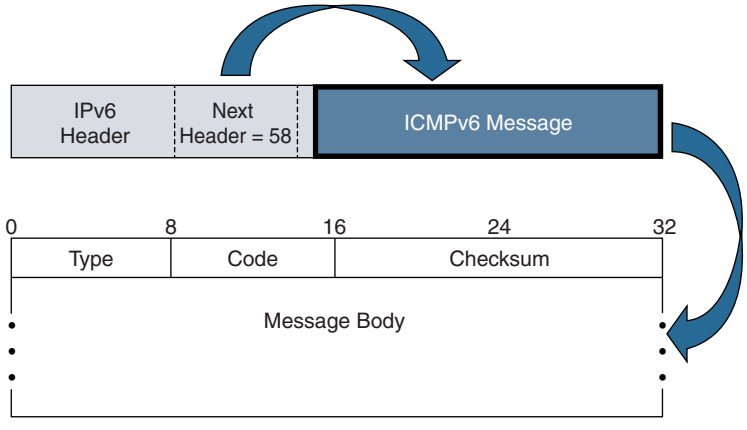


Figure 12-1 *General Format of an ICMPv6 Message*

Figure 12-1 shows three fields in the message:

- **Type (8 bits):** Indicates the type of ICMPv6 message, such as Echo Request, Destination Unreachable, or Packet Too Big.
- **Code (8 bits):** Provides more granularity for the Type field. Its meaning depends on the message type. For example, if the message type is Destination Unreachable, the Code field gives the specific reason the packet was not able to reach its destination; for example, the router did not have a route to a host's network in its routing table.
- **Checksum (16 bits):** Used to detect data corruption in the ICMPv6 message and parts of the IPv6 header.

The Type field is used to group ICMPv6 messages into two classes:

- **Error messages:** Type = 0 to 127
- **Informational messages:** Type = 128 to 255

Error messages are identified by a high-order bit of 0 (0xxxxxx) in the message Type field. This results in an error message having a Type value from 0 to 127. Therefore, informational messages with a high-order bit of 1 have a Type value of 128 to 255.

ICMPv6 error messages notify a device why a packet it sent could not be delivered; for example, the hop limit was reached, it was decremented to 0, and then the packet was dropped by the router.

ICMPv6 informational messages are not used to report errors but to provide information required for various testing, diagnostics, and support functions. Two common informational messages in both ICMP for IPv4 and IPv6 are the Echo Request and Echo Reply messages used by the **ping** command.

Table 12-1 provides an overview of the different types of ICMPv6 error messages, and Table 12-2 provides an overview of the different types of ICMPv6 informational messages.

Table 12-1 *ICMPv6 Error Messages*

Type	Type Description	Code and Code Description
1	Destination Unreachable	0: No route to destination
		1: Communication with destination administratively prohibited
		2: Beyond scope of source address
		3: Address unreachable
		4: Port unreachable
		5: Source address failed ingress/egress policy
		6: Reject route to destination

Type	Type Description	Code and Code Description
2	Packet Too Big	0: Ignored by receiver
3	Time Exceeded	0: Hop limit exceeded in transit
		1: Fragment reassembly time exceeded
4	Parameter Problem	0: Erroneous header field encountered
		1: Unrecognized Next Header type encountered
		2: Unrecognized IPv6 option encountered
101	Private Experimentation	
107	Private Experimentation	
127	Reserved for expansion of ICMPv6 error messages	

Table 12-2 *ICMPv6 Informational Messages*

Type	Type Description	Code and Code Description
Used by the **ping** *command (RFC 4443)*		
128	Echo Request	0: Ignored by receiver
129	Echo Reply	0: Ignored by receiver
Used for Multicast Listener Discovery (RFC 2710)		
130	Multicast Listener Query	0: Ignored by receiver
131	Multicast Listener Report	0: Ignored by receiver
132	Multicast Listener Done	0: Ignored by receiver
Used by Neighbor Discovery (RFC 4861)		
133	Router Solicitation	0: Ignored by receiver
134	Router Advertisement	0: Ignored by receiver
135	Neighbor Solicitation	0: Ignored by receiver
136	Neighbor Advertisement	0: Ignored by receiver
137	Redirect message	0: Ignored by receiver

Table 12-3 lists ICMPv6 informational messages that are beyond the scope of this book but are included here for completeness and to give you a better idea of how ICMPv6 is used.

Table 12-3 *Additional ICMPv6 Informational Messages*

Type	Description
Type 138 Router Renumbering	Used for router renumbering, the mechanisms for informing a set of routers of renumbering operations they are to perform. (RFC 2894)
Type 139 Node Information Query **Type 140** Node Information Reply	Requests an IPv6 node to supply certain network information, such as its host name or fully qualified domain name. (RFC 4620)
Type 141 Inverse Neighbor Discovery Solicitation message **Type 142** Inverse Neighbor Discovery Advertisement message	Allows a node to determine and advertise an IPv6 address corresponding to a given link layer address—similar to IPv4 Inverse ARP used with Frame Relay. (RFC 3122)
Type 143 Version 2 Multicast Listener Report message	MLDv2 adds the ability for a node to report interest in listening to packets with a particular multicast address only from specific source addresses or from all sources except for specific source addresses. (RFC 3810)
Type 144 ICMP Home Agent Address Discovery Request message **Type 145** ICMP Home Agent Address Discovery Reply message **Type 146** ICMP Mobile Prefix Solicitation message **Type 147** ICMP Mobile Prefix Advertisement message	Supports mobile IPv6. (RFC 3775)

Type	Description
Type 148 Certification Path Solicitation message **Type 149** Certification Path Advertisement message	Provides security mechanisms for Neighbor Discovery Protocol.
Type 151 Advertisement Packet **Type 152** Solicitation Packet **Type 153** Termination Packet	Allows the discovery of multicast routers. (RFC 4286)
Type 200 **Type 201**	Used for private experimentation. (RFC 4443)

ICMP Error Messages

Layer 3 devices, such as hosts and routers, use ICMPv6 error messages to notify the sender as to why a packet could not be delivered. As shown in Table 12-1, there are four types of error messages:

- Destination Unreachable
- Packet Too Big
- Time Exceeded
- Parameter Problem

Note An ICMPv6 error message must never be sent as a result of a previous ICMPv6 error message because this could lead to confusion and a never-ending exchange of error messages.

Destination Unreachable

An ICMPv6 Destination Unreachable message is sent when a packet cannot be delivered to its destination for reasons other than congestion. Instead of having a source just re-send packets without any idea why they were not received, these feedback messages

are used to help provide some useful information to the sender about the reason why. A router or a firewall usually generates these Destination Unreachable messages.

Note It can be argued that a firewall or any other security device should not send Destination Unreachable messages.

Figure 12-2 shows the format of a Destination Unreachable message. Notice that the Type field is set to 1.

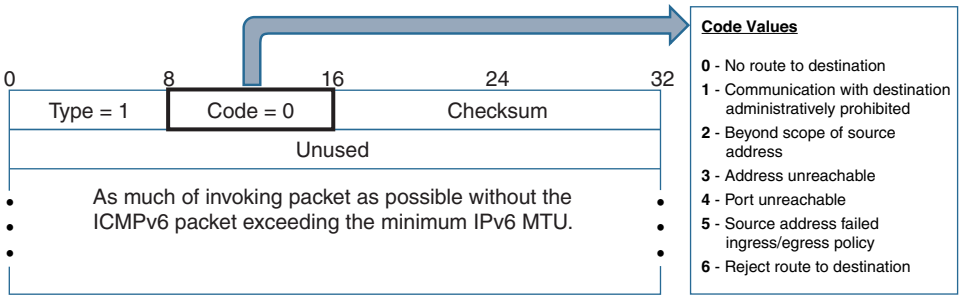


Figure 12-2 *ICMPv6 Destination Unreachable Message*

There are several reasons why the destination might be unreachable. The Code field is used to provide more detailed information about why the packet was not delivered. There are seven codes:

- **Code = 0, No route to destination:** The packet could not be delivered because the router did not have a route to this destination. This can only occur if the router does not have a default route in its routing table. This message is equivalent to the Network Unreachable message in ICMPv4.
- **Code = 1, Communication with destination administratively prohibited:** The packet was blocked because of an access control list or other packet filtering.
- **Code = 2, Beyond scope of source address:** This error message is generated when the source address is a link-local address and the destination address is a global unicast address.
- **Code = 3, Address unreachable:** This error message indicates that there was a problem delivering the packet because the host specified in the destination address could not be reached. This can occur if the destination address could not be resolved to its corresponding data-link address (MAC address on LANs) or the destination address is incorrect. This is equivalent to the ICMPv4 Host Unreachable message.
- **Code = 4, Port unreachable:** This error message occurs because the destination port specified in the TCP or UDP header does not exist or the destination is not listening on that port. For example, if a packet is sent with a TCP destination port 80, but the receiving host is not running the HTTP web service, a Port Unreachable message is transmitted.

- **Code = 5, Source address failed ingress/egress policy:** This error message indicates that the packet with this source address is blocked because of an access control list or other packet filtering. Code 5 is a subset of Code 1.
- **Code = 6, Reject route to destination:** This error message occurs when packets with a specific prefix are blocked by an access control list or other packet filtering. Code 6 is a subset of Code 1.

Example 12-1 shows a Wireshark capture of an ICMPv6 Destination Unreachable message. The error message was caused by a failed **ping** attempt to 2a03:2880:f122:face:b00c::1. For brevity, Example 12-1 displays only the ICMPv6 Destination Unreachable message. However, notice that the Data field of the error message contains the IPv6 header and ICMPv6 message of the original ICMPv6 Echo Request that generated this Destination Unreachable message.

Example 12-1 *Wireshark Capture of an ICMPv6 Destination Unreachable Message*

```
C:\> ping -6 2a03:2880:f122:face:b00c::1
Destination host unreachable.

<Wireshark capture: ICMPv6 Message>

Internet Control Message Protocol v6
    Type: Destination Unreachable (1)
    Code: 3 (Address unreachable)
    Checksum: 0xfec5 [correct]
    Reserved: 00000000
    Data:
    Internet Protocol Version 6,
        Version: 6
        Traffic class: 0x00000000
          Differentiated Services Field: Default (0x00000000)
          ECN-Capable Transport (ECT): Not set
          ECN-CE: Not set
        Flowlabel: 0x00000000
        Payload length: 40
        Next header: ICMPv6 (58)
        Hop limit: 119
        Source: 2601:648:cd01:a683:8cd3:96f5:3ff9:3b48
        Destination: 2a03:2880:f122:face:b00c::1
    Internet Control Message Protocol v6
        Type: Echo (ping) request (128)
        Code: 0
        Checksum: 0xa795 [correct]
        Identifier: 0x0001
        Sequence: 9
        Data (32 bytes)

C:\ >
```

Packet Too Big

A significant change to IPv6 is related to packet fragmentation and reassembly. In IPv4, routers can fragment a packet when the maximum transmission unit (MTU) of the outgoing link is smaller than the size of the packet. The destination device is responsible for reassembling the fragmented packets.

Although it might be convenient to have routers fragment packets when needed, it is also inefficient for a router to manage the fragmentation. IPv6 removes this task from the router, allowing only the source of the packet to perform fragmentation. When an IPv6 router receives a packet larger than the MTU of the egress interface, the router drops the packet and sends an ICMPv6 Packet Too Big message back to the source. The Packet Too Big message includes the MTU size of the link in bytes so that the source can change the size of the packet for retransmission.

Note A router can fragment an IPv6 packet if it is the source of the packet. Fragmentation in IPv6, including the use of extension header 44 (Fragment), is discussed in Chapter 3, "Comparing IPv4 and IPv6."

As shown in Figure 12-3, an ICMPv6 Packet Too Big message has the Type set to 2 and the Code equal to 0. The MTU is the maximum transmission unit of the next-hop link. This ICMPv6 error message is also used as part of Path MTU Discovery.

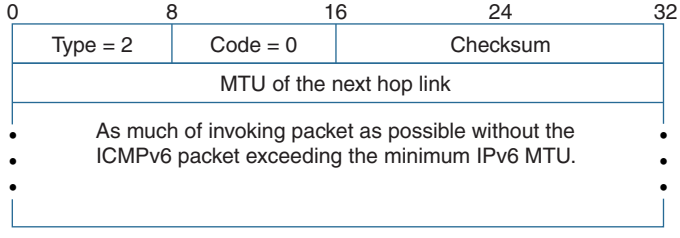


Figure 12-3 *ICMPv6 Packet Too Big Message*

Path MTU Discovery

Path MTU Discovery is defined in RFC 1981, *Path MTU Discovery for IP version 6*. When a device has a large number of packets to transmit, it is preferable for these packets to be as large as possible so that fewer packets need to be sent. This requires the source to know the minimum link MTU (the smallest MTU) of all the links in the path to the destination. The sender can then transmit the largest packet size possible, without the risk of a router dropping the packet along the path because the MTU of its outgoing link is too small. The size of this packet is referred to as the *Path MTU* (*PMTU*).

Note IPv6 requires that every link on the Internet have a minimum MTU of 1280 bytes, compared to 68 bytes for IPv4. This improves the ratio of payload to header length and reduces the need for fragmentation by the source.

The Path MTU Discovery process works as illustrated in Figure 12-4 and as described in the following steps:

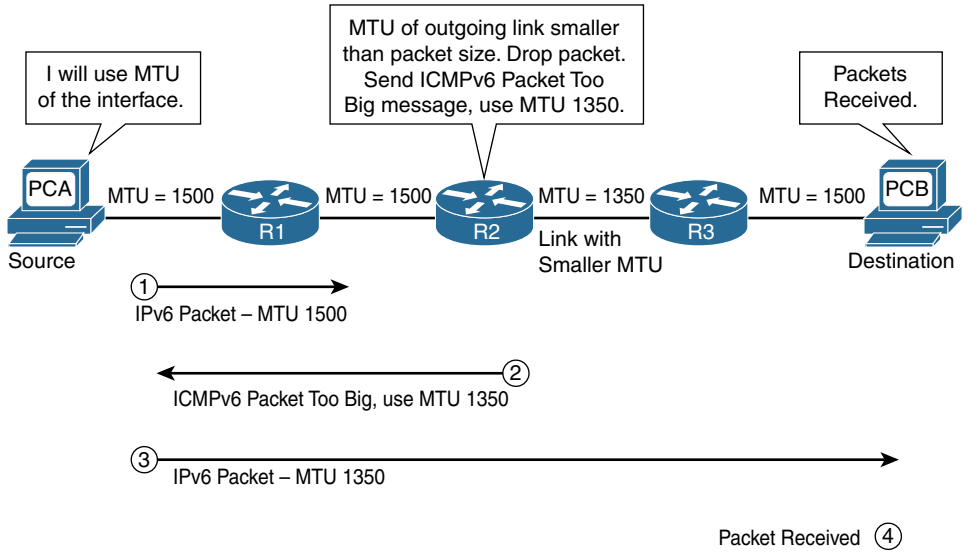


Figure 12-4 *Path MTU Discovery*

Step 1. The source device assumes that the Path MTU (PMTU) size of the packet is equal to the MTU of its outgoing link to the first-hop router. In Figure 12-4, PC-A sets the PMTU to 1500 bytes, the MTU of the Ethernet link.

Step 2. If the packet size is larger than the MTU of the router's next-hop link, the packet is dropped, and an ICMPv6 Packet Too Big message is sent back to the source. Included in the message is the MTU of the next-hop link. In Figure 12-4, router R2 drops the packet and sends a Packet Too Big message back to PC-A, with an MTU of 1350 bytes.

Step 3. The source device uses the information in the Packet Too Big message to reduce the packet size to match the MTU contained in the message. The source device sends further packets using this smaller MTU. PC-A in Figure 12-4 transmits further packets with the new MTU of 1350. The MTU size will never go below 1280 bytes, the minimum link MTU for IPv6. (In Figure 12-4, PC-A would resend the original 1500 byte packet as two 1350 byte packets, using extension header 44 [Fragment].)

Step 4. This process of routers sending ICMPv6 Packet Too Big messages and the source reducing its packet size continues until the packet reaches its destination. In Figure 12-4, the new MTU was sufficient for sending the packet to its destination PC-B.

Because the path from a specific source to a given destination can change, so might the PMTU. So, it is possible that source devices might have to dynamically modify the PMTU size of their packets. Devices are not required to implement Path MTU, but doing so is recommended in RFC 4443. Path MTU Discovery supports multicast as well as unicast destinations.

Time Exceeded

To avoid the possibility of packets traveling endlessly between networks due to a routing loop or another problem, there is a prevention mechanism for both IPv4 and IPv6.

Before a router forwards an IPv6 packet, it decrements the Hop Limit field by 1. When a router receives a packet with a Hop Limit of 0, or a router decrements a packet's Hop Limit to 0, it must discard the packet and send an ICMPv6 Time Exceeded message to the source of the packet. This indicates either a routing loop or an initial Hop Limit value that is too small. The Time Exceeded field is identical to the TTL field in IPv4, but its name better reflects its function.

An ICMPv6 Time Exceeded message has a Type value of 3 and Code value of 0, as shown in Figure 12-5.

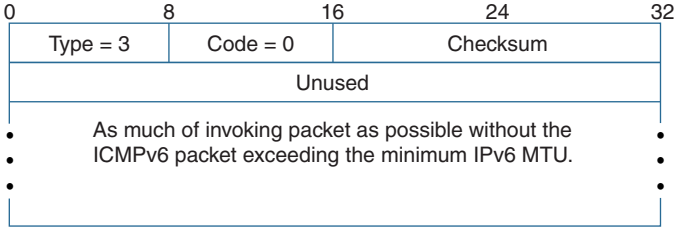


Figure 12-5 *ICMPv6 Time Exceeded Message*

The traceroute utility uses the Time Exceeded message to help determine the path of routers to the destination. In the first step of the traceroute utility, the program sends a sequence of ICMPv6 Echo Request messages with the Hop Limit field of 1. If the source receives an ICMPv6 Time Exceeded message, it displays the source address of the packet, increments the Hop Limit by 1, and sends another Echo Request. This continues until the Echo Request reaches the destination.

Example 12-2 shows an example of using traceroute (**tracert** command) on Windows. The **-6** parameter is used to ensure IPv6 with ICMPv6 is used instead of IPv4 with ICMPv4. Notice the creative use of hexadecimal in some of the addresses (2a03:2880:f122:83:*face:b00c*:0:25de and 2620:0:1cff:*dead:beef*::1019). For Linux and Mac OS, the **traceroute6** command is used with IPv6.

Example 12-2 *Windows* **traceroute** *Using IPv6 and ICMPv6*

```
C:\> tracert -6 www.facebook.com

Tracing route to star-mini.c10r.facebook.com [2a03:2880:f122:83:face:b00c:0:25de]
over a maximum of 30 hops:

  1     6 ms     4 ms     4 ms  2601:647:cd00:a683:481d:70ff:fe6f:9503
  2    12 ms    16 ms    13 ms  2001:558:4000:94::1
  3    14 ms    14 ms    14 ms  te-0-7-0-6-sur04.scotts.ca.sfba.comcast.net
                                   [2001:558:82:172::1]
  4    18 ms    15 ms    18 ms  be-321-ar01.hayward.ca.sfba.comcast.net
                                   [2001:558:80:45a::1]
  5    20 ms    22 ms    16 ms  hu-0-0-0-0-ar01.santaclara.ca.sfba.comcast.net
                                   [2001:558:80:38::1]
  6    15 ms    17 ms    19 ms  be-33651-cr01.sunnyvale.ca.ibone.comcast.net
                                   [2001:558:0:f697::1]
  7    59 ms    24 ms    20 ms  be-10925-cr01.9greatoaks.ca.ibone.comcast.net
                                   [2001:558:0:f5e7::2]
  8    21 ms    18 ms    20 ms  hu-0-13-0-1-pe03.11greatoaks.ca.ibone.comcast.net
                                   [2001:558:0:f8e2::2]
  9    25 ms    24 ms    18 ms  as32934-2-c.11greatoaks.ca.ibone.comcast.net
                                   [2001:559::136]
 10    19 ms    19 ms    25 ms  po131.asw04.sjc1.tfbnw.net
                                   [2620:0:1cff:dead:beef::11fa]
 11    24 ms    19 ms    20 ms  po241.psw03.sjc2.tfbnw.net
                                   [2620:0:1cff:dead:beef::1019]
 12    17 ms    29 ms    24 ms  po3.msw1aj.01.sjc2.tfbnw.net
                                   [2a03:2880:f022:ffff::7b]
 13    26 ms    19 ms    18 ms  edge-star-mini6-shv-01-sjc2.facebook.com [2a03:2880:
                                   f122:83:face:b00c:0:25de]

Trace complete.
C:\ >
```

The Cisco IOS **traceroute** command uses UDP segments with an initial destination port number of 33434, or as specified in the extended **traceroute** command. The IPv6 hop limit of the first packet is 1. As long as an ICMP Time Exceeded message is received by the source, this cycle repeats, with incremental hop limits and destination port numbers. When the source receives an ICMPv6 Port Unreachable message, it knows that the destination has been reached.

Note RFC 1393, *Traceroute Using an IP Option*, defines a more efficient way to perform a traceroute. The source device sends a single packet to the destination, containing a special Traceroute IP option. Each router in the path recognizes the option as the test message and responds to the original source with an ICMP Traceroute message.

The same IOS **traceroute** command is used for both IPv4 and IPv6 when tracing the path to a destination IP address. When the destination is a hostname (domain name), to have IOS prefer IPv6, use the optional **ipv6** parameter with the **traceroute** command:

```
Router# traceroute ipv6 hostname
```

Using the topology in Figure 12-6, Example 12-3 shows the IOS **traceroute** command when using both an IPv6 address and a hostname. Much like the **ip host** command in IPv4, the **ipv6 host** command is used to map a hostname to an IPv6 address.

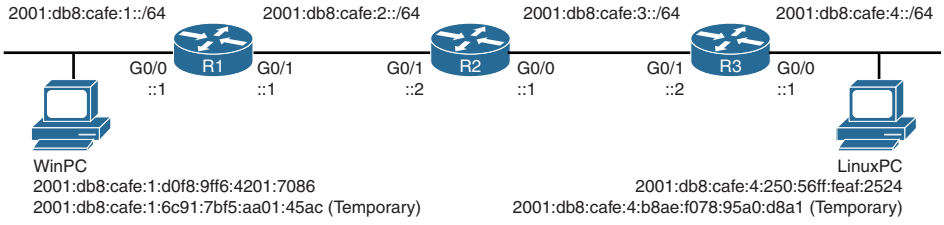


Figure 12-6 *Topology for ICMPv6 Messages*

Figure 12-6 shows that WinPC and LinuxPC both have two global unicast addresses. As discussed in Chapter 9, "Stateless Address Autoconfiguration (SLAAC)," the Windows and Ubuntu Linux operating systems use the privacy extension when deriving their GUA addresses using SLAAC. The first address is the public address, and the second address is the temporary address. A temporary address has a short lifetime and is used as the source address when the device initiates the conversation.

Example 12-3 *Cisco IOS* **traceroute** *Using IPv6 and ICMPv6*

```
R1# traceroute 2001:db8:cafe:4:250:56ff:feaf:2524
Type escape sequence to abort.
Tracing the route to LinuxPC (2001:DB8:CAFE:4:250:56FF:FEAF:2524)

  1 2001:DB8:CAFE:2::2 0 msec 0 msec 0 msec
  2 2001:DB8:CAFE:3::2 0 msec 0 msec 4 msec
  3 LinuxPC (2001:DB8:CAFE:4:250:56FF:FEAF:2524) 0 msec 0 msec 0 msec
R1#

R1# config t
R1(config)# ipv6 host LinuxPC 2001:db8:cafe:4:250:56ff:feaf:2524
R1(config)# end
R1# traceroute ipv6 LinuxPC
Type escape sequence to abort.
Tracing the route to LinuxPC (2001:DB8:CAFE:4:250:56FF:FEAF:2524)

  1 2001:DB8:CAFE:2::2 4 msec 0 msec 0 msec
  2 2001:DB8:CAFE:3::2 0 msec 0 msec 0 msec
 3 LinuxPC (2001:DB8:CAFE:4:250:56FF:FEAF:2524) 8 msec 0 msec 0 msec
R1#
```

Note Example 12-3 uses only IPv6 addresses, so you could just use the command **traceroute LinuxPC** to get the same result. In a dual-stacked environment, the same *hostname* can be used for both the **ip host** and **ipv6 host** commands, in which case you may need to use **traceroute ipv6** *hostname* when preferring IPv6. To troubleshoot a protocol in a dual-stacked environment, you need to specify the proper IP protocol.

An ICMPv6 Type 3 with Code equal to 1 is a result of the destination device not receiving all the fragmented packets after attempting to reassemble the original message. After receiving the first fragment, instead of waiting indefinitely, a timer is set, with a specific amount of time allotted for all the fragments to reach this destination. If they do not all arrive in the appropriate time, an ICMPv6 Time Exceeded message is sent to the source.

Parameter Problem

An ICMPv6 Parameter Problem error message is generated when a device processing a packet finds a problem with a field in the main IPv6 header or an extension header. This means that the receiving device didn't understand the information in the IPv6 header and had to discard it. Figure 12-7 shows a packet with an invalid value in the Next Header field. The receiving device would generate an ICMPv6 Parameter Problem message with Code set to 1 (see Figure 12-8). This problem occurs if the arguments in the extension header are invalid or if the extension header is unsupported by the device.

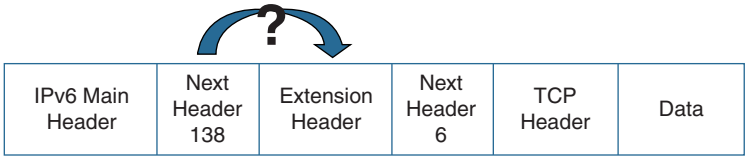


Figure 12-7 *Packet with Unrecognized Next Header Value*

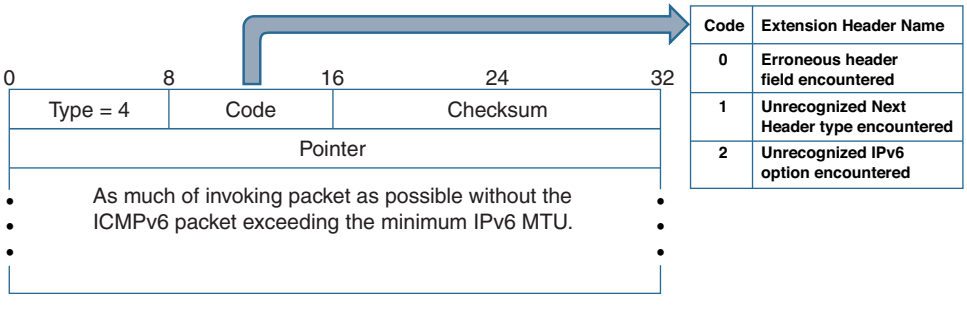


Figure 12-8 *ICMPv6 Parameter Problem Message*

ICMP Informational Messages

As we saw earlier, devices use ICMPv6 error messages to let senders know why packets could not be delivered. ICMPv6 informational messages are used to help devices discover and share information with each other. As shown in Table 12-2 and listed as follows, there are several types of informational messages:

- ICMPv6 informational messages used by ping:
  - Echo Request
  - Echo Reply
- ICMPv6 informational messages used for Multicast Listener Discovery (see Chapter 7):
  - Multicast Listener Query
  - Multicast Listener Report
  - Multicast Listener Done
- ICMPv6 informational messages used by Neighbor Discovery (RFC 4861) (see Chapter 13):
  - Router Solicitation message
  - Router Advertisement message
  - Neighbor Solicitation message
  - Neighbor Advertisement message
  - Redirect message

Echo Request and Echo Reply

Echo Request and Echo Reply are two ICMP messages used by ping, a very common TCP/IP utility and a staple of every system/network administrator. The **ping** command is commonly used to test network layer connectivity between two devices. Its name is derived from active sonar terminology, which sends pulses of sound and then waits for a reflection (echo).

A device sends an Echo Request to prompt the destination to return an Echo Reply to verify network layer connectivity. If the sender does not receive a corresponding Echo Reply, it does not necessarily mean that the destination could not be reached. It is possible that internetwork devices in the path are discarding the Echo Request or the Echo Reply messages. It is also possible that the destination itself is not accepting or responding to Echo Request messages.

Figure 12-8 shows the format of ICMPv6 Echo Request and Echo Reply messages. The structure of an Echo Request and Echo Reply is identical except for the value in the Type field. An Echo Request has the Type field set to 128, whereas an Echo Reply has a Type value of 129. The Code field is always set to 0. The rest of the fields are shown in Figure 12-9.

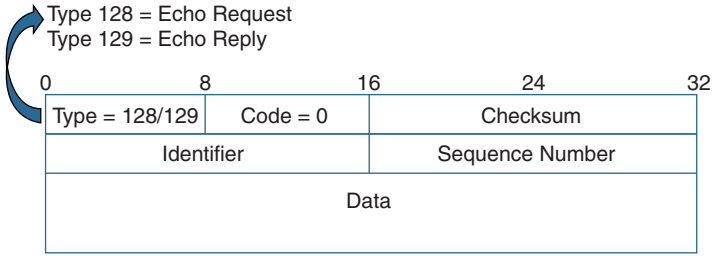


Figure 12-9 *ICMPv6 Echo Request and Echo Reply Messages*

Pinging a Global Unicast Address

Example 12-4 shows the **ping** command issued from WinPC and sent to the global unicast address of router R1. Except for the IPv6 addresses, this looks identical to pinging an IPv4 address.

Example 12-4 *Global Unicast Address* **ping** *from WinPC to R1*

```
WinPC> ping 2001:db8:cafe:1::1

Pinging 2001:db8:cafe:1::1 with 32 bytes of data:

Reply from 2001:db8:cafe:1::1: time=1ms
Reply from 2001:db8:cafe:1::1: time<1ms
Reply from 2001:db8:cafe:1::1: time<1ms
Reply from 2001:db8:cafe:1::1: time<1ms
Ping statistics for 2001:db8:cafe:1::1:

    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

WinPC>
```

The **ping -6** *target-name* command can be used on dual-stacked Windows devices to ensure IPv6 is used instead of IPv4. In Linux and Mac OS, the **ping6** command must be used to ping an IPv6 address or target.

Example 12-5 examines the ICMPv6 Echo Request sent from WinPC to Router R1.

Example 12-5 *Echo Request from WinPC to R1*

```
Ethernet II, Src: 00:50:56:af:97:68, Dst: 58:ac:78:93:da:00

Internet Protocol Version 6
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
    Next header: ICMPv6 (58)
    Hop limit: 128
    Source: 2001:db8:cafe:1:6c91:7bf5:aa01:45ac
    Destination: 2001:db8:cafe:1::1

Internet Control Message Protocol v6
    Type: 128 (Echo (ping) request)
    Code: 0 (Should always be zero)
    Checksum: 0x0b48 [correct]
    ID: 0x0001
    Sequence: 0
    Data (32 bytes)
```

Notice the source IPv6 address of the Echo Request. The destination address in the **ping** command is a global unicast address, so the source address is also a global unicast address. As we mentioned earlier, Windows uses the privacy extension when deriving its address using SLAAC. Because WinPC initiates the ping conversation with R1 in this example, it uses its temporary address for the source IPv6 address.

Example 12-6 shows similar information for the corresponding ICMPv6 Echo Reply. Router R1 uses the source IPv6 address of the Echo Request for its destination IPv6 address in its Echo Reply.

Note Default source address selection is discussed in Chapter 9.

In both IPv6 packets, the value in the Next Header field is decimal 58, indicating that an ICMPv6 header follows the IPv6 header. Some versions of Wireshark use hexadecimal values, such as 0x3a, for ICMPv6.

Example 12-6 *Echo Reply from R1 to WinPC*

```
Ethernet II, Src: 58:ac:78:93:da:00, Dst: 00:50:56:af:97:68

Internet Protocol Version 6
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
     Next header: ICMPv6 (58)
    Hop limit: 64
    Source: 2001:db8:cafe:1::1
    Destination: 2001:db8:cafe:1:6c91:7bf5:aa01:45ac

Internet Control Message Protocol v6
    Type: 129 (Echo (ping) reply)
    Code: 0 (Should always be zero)
    Checksum: 0x0a48 [correct]
    ID: 0x0001
    Sequence: 0
    Data (32 bytes)
```

Looking at the ICMPv6 header in both Examples 12-5 and 12-6 gives you a better understanding of these messages:

- **Type:** In the Echo Request, Type is set to 128, and in the Echo Reply, Type is set to 129.
- **Code:** Ignored by the receiver, Code is always 0 for both an Echo Request and an Echo Reply.
- **Checksum:** Checksum validates the ICMPv6 header.
- **Identifier:** This field is used to help match a series of Echo Request messages with their corresponding Echo Reply messages. Notice that the Echo Request and the Echo Reply have the same value. This value remains constant for the entire sequence of Echo Request messages and the Echo Reply messages generated by this instance of the **ping** command. In the examples, Identifier is set to 1 for the series of ICMPv6 messages.
- **Sequence:** This field also helps match Echo Request and Echo Reply messages. The Sequence field provides more granularity by matching individual Echo Reply messages with their specific Echo Request messages. An Echo Request is generated with a Sequence number, and its corresponding Echo Reply includes that same Sequence number. The next Echo Request increments the Sequence number by 1, and the receiver uses that same value in its returning Echo Reply. In Examples 12-5

and 12-6, the Sequence number is 0 for both the Echo Request and the Echo Reply. The next Echo Request and Echo Reply from the same **ping** command increments the Sequence number to 1.

- **Data:** The Echo Request adds zero or more bytes of arbitrary data. The receiving device's Echo Reply copies this data into the returned Echo Request.

Pinging a Link-Local Address

In Example 12-7, another **ping** command is issued, this time pinging the link-local address from router R1 to WinPC. Remember that a link-local address has to be unique only on that link and is never routed off the link. (The examples in this section is a review of the same information covered in Chapter 6, "Link-Local Unicast Address.")

Example 12-7 *Pinging Link-Local Address from R1 to WinPC*

```
R1# ping fe80::d0f8:9ff6:4201:7086
Output Interface: g 0/0
% Invalid interface. Use full interface name without spaces (e.g. Serial0/1)
Output Interface: gigabitethernet0/0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FE80::D0F8:9FF6:4201:7086, timeout is 2 seconds:
Packet sent with a source address of FE80::1%GigabitEthernet0/0
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R1#
```

In Example 12-8, notice that you cannot ping a link-local address of WinPC without first specifying the exit interface, or *output interface*, of the router. This is because link-local addresses are not included in the router's routing table, so the router doesn't know which exit interface to use. As you can see in the failure of the first **ping** command to a link-local address, Cisco IOS requires the use of the full interface name, without any spaces.

Examples 12-8 and 12-9 show the Echo Request and Echo Reply, using the link-local addresses.

Example 12-8 *Echo Request to a Link-Local Address from R1 to WinPC*

```
Ethernet II, Src: 58:ac:78:93:da:00, Dst: 00:50:56:af:97:68

Internet Protocol Version 6
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 60
    Next header: ICMPv6 (58)
```

```
   Hop limit: 64
     Source: fe80::1
     Destination: fe80::d0f8:9ff6:4201:7086

Internet Control Message Protocol v6
    Type: 128 (Echo (ping) request)
    Code: 0 (Should always be zero)
    Checksum: 0xb9b3 [correct]
    ID: 0x18b5
    Sequence: 0
    Data (52 bytes)
```

Example 12-9 *Echo Reply to a Link-Local Address from WinPC to R1*

```
Ethernet II, Src: 00:50:56:af:97:68, Dst: 58:ac:78:93:da:00

Internet Protocol Version 6
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 60
    Next header: ICMPv6 (58)
    Hop limit: 64
     Source: fe80::d0f8:9ff6:4201:7086
     Destination: fe80::1

Internet Control Message Protocol v6
    Type: 129 (Echo (ping) reply)
    Code: 0 (Should always be zero)
    Checksum: 0xb8b3 [correct]
    ID: 0x18b5
    Sequence: 0
     Data (52 bytes)
```

Notice that both devices use their link-local addresses as the IPv6 source address of the packet. The remainder of the message is similar to that of the previous ICMPv6 messages. Again, link-local addresses are confined to that link. Therefore, WinPC is not able to ping the link-local address of LinuxPC, which is on a different network or link.

In Windows, when you need to ping the link-local address *from* the host, WinPC, which has only a single interface, you do not have to include an exit interface, as shown in Example 12-10.

Example 12-10 *Pinging a Link-Local Address from WinPC to R1*

```
WinPC> ping fe80::1

Pinging fe80::1 with 32 bytes of data:

Reply from fe80::1: time<1ms
Reply from fe80::1: time<1ms
Reply from fe80::1: time<1ms
Reply from fe80::1: time<1ms

Ping statistics for fe80::1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

WinPC>
```

If a Windows device has multiple interfaces, the Zone ID is required. Example 12-11 shows an example of determining the Zone ID and using it with the **ping** command.

Example 12-11 *IPv6 Configuration on WinPC and Pinging with Zone ID*

```
WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:db8:cafe:1:d0f8:9ff6:4201:7086
   Temporary IPv6 Address. . . . . . : 2001:db8:cafe:1:6c91:7bf5:aa01:45ac
   Link-local IPv6 Address . . . . . : fe80::d0f8:9ff6:4201:7086%11
   Default Gateway . . . . . . . . . : fe80::1%11
<output omitted for brevity>

WinPC> ping fe80::1%11

Pinging fe80::1 with 32 bytes of data:

Reply from fe80::1: time<1ms
Reply from fe80::1: time<1ms
<output omitted for brevity>
```

With Linux and Mac OS, the exit interface must always be used with the **ping6** command. Example 12-12 shows two options using the **ping6** command to ping a link-local address, using both *%interface* at the end of the address and the **–I** *interface* parameter.

Example 12-12 *Pinging a Link-Local Address from Linux OS*

```
LinuxPC$ ping6 fe80::3
Connect: Invalid argument

LinuxPC$ ping6 fe80::3%eth0
PING fe80::3%eth0(fe80::3) 56 data bytes
64 bytes from fe80::3: icmp_seq=0 ttl=64 time=0.552 ms
64 bytes from fe80::3: icmp_seq=1 ttl=64 time=0.429 ms
<output omitted for brevity>

LinuxPC$ ping6 -I eth0 fe80::3
PING fe80::3%eth0(fe80::3) 56 data bytes
64 bytes from fe80::3: icmp_seq=0 ttl=64 time=0.552 ms
64 bytes from fe80::3: icmp_seq=1 ttl=64 time=0.551 ms
<output omitted for brevity>
```

Summary

This chapter examines ICMPv6. ICMPv6 has many similarities to ICMPv4 but is a more robust protocol that contains new functionality and many improvements.

Two types of ICMPv6 messages are discussed: error messages and informational messages.

This chapter explores the ICMPv6 error messages:

- **Destination Unreachable:** These messages are sent when a packet cannot be delivered to its destination for reasons other than congestion.
- **Packet Too Big:** An IPv6 router sends an ICMPv6 Packet Too Big message back to the source when it receives a packet larger than the MTU of the egress interface. The router drops the packet. This message is also used for Path MTU Discovery.
- **Time Exceeded:** Before a router forwards an IPv6 packet, it decrements the Hop Limit field by 1, much as the TTL field does in IPv4. If the Hop Limit results in a 0, the packet is dropped, and an ICMPv6 Time Exceeded message is sent back to the source.
- **Parameter Problem:** This message is generated when a device processing a packet finds a problem with a field in the main IPv6 header or an extension header and must discard it.

This chapter also covers ICMPv6 informational messages, including these two messages that are commonly used by the **ping** command:

- **Echo Request:** A device sends an Echo Request to prompt the destination to return an Echo Reply to verify network layer connectivity.
- **Echo Reply:** An Echo Reply is sent in response to an Echo Request.

Review Questions

1. Classify each of the ICMPv6 messages as either an error message or an informational message.
   a. Multicast Listener Query
   b. Time Exceeded
   c. Echo Request
   d. Multicast Listener Done
   e. Destination Unreachable
   f. Packet Too Big
   g. Echo Reply
   h. Multicast Listener Report
2. Which ICMPv6 message is sent when a packet cannot be delivered to its destination for reasons other than congestion?
   a. Time Exceeded
   b. Packet Too Big
   c. Echo Reply
   d. Echo Request
   e. Parameter Problem
   f. Destination Unreachable
3. What does a router do with a packet it receives that is larger than the MTU of the exit interface?
   a. It fragments the packet and sends an ICMPv6 Packet Too Big message back to the source.
   b. It fragments the packet and sends an ICMPv6 Packet Too Big message to the previous-hop router.
   c. It drops the packet and sends an ICMPv6 Packet Too Big message back to the source.
   d. It drops the packet and sends an ICMPv6 Packet Too Big message to the previous-hop router.

4. Which ICMPv6 message is sent back to the source, indicating either a routing loop or an initial Hop Limit value that is too small?
   a. Time Exceeded
   b. Packet Too Big
   c. Echo Reply
   d. Echo Request
   e. Parameter Problem
   f. Destination Unreachable
5. Which ICMPv6 message is sent back to the source when a device processing a packet finds a problem with a field in the main IPv6 header or an extension header?
   a. Time Exceeded
   b. Packet Too Big
   c. Echo Reply
   d. Echo Request
   e. Parameter Problem
   f. Destination Unreachable
6. A device implementing the privacy extension has generated two global unicast addresses using SLAAC. What does it use as its source IPv6 address when pinging a public GUA address of another device?
7. What address does a device use as its source IPv6 address when pinging a link-local address of another device?
8. Besides the destination IPv6 address, what else is required when pinging a link-local address from a Cisco router?
   a. The source IPv6 address
   b. The outgoing interface
   c. The zone Identifier
   d. Echo Request
9. What field is used to match individual Echo Request messages with their specific Echo Reply?
   a. Type
   b. Code
   c. Identifier
   d. Sequence
10. What field is used to match a sequence of Echo Request messages with their corresponding Echo Reply messages and remains constant for the entire sequence of messages?
    a. Type
    b. Code
    c. Identifier
    d. Sequence
11. Why do operating systems require an exit interface when pinging a link-local address?

References

RFCs

RFC 1393, *Traceroute Using an IP Option*, G. Malkin, Xylogics, Inc., www.ietf.org/rfc/rfc1393.txt, January 1963.

RFC 1981, *Path MTU Discovery for IP Version 6*, J. McCann, DEC, www.ietf.org/rfc/rfc1981.txt, August 1996.

RFC 2464, *Transmission of IPv6 Packets over Ethernet Networks*, M. Crawford, Fermilab, www.ietf.org/rfc/rfc2464.txt, December 1998.

RFC 2711, *IPv6 Router Alert Option*, C. Partridge, BBN, www.ietf.org/rfc/rfc2711.txt, October 1999.

RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, A. Conta, Transwitch, www.ietf.org/rfc/rfc4443.txt, March 2006.

This page intentionally left blank

Chapter 13

ICMPv6 Neighbor Discovery

As you saw in Chapter 12, "ICMPv6," ICMPv6 has its roots in ICMPv4. ICMPv6 informational and error messages are very similar to those in ICMPv4. However, the addition of ICMPv6 Neighbor Discovery Protocol (ND) makes ICMPv6 a much more robust protocol, with new features and enhancements not found in its ICMPv4 counterpart.

ICMPv6 ND is defined in RFC 4861, *Neighbor Discovery for IP Version 6 (IPv6)*. Neighbor Discovery includes similar processes as in IPv4, such as address resolution, router discovery, and redirect, but also with some significant differences. ICMPv6 ND also includes new functionality such as prefix discovery, Duplicate Address Detection (DAD), and Neighbor Unreachability Detection (NUD).

Neighbor Discovery uses five ICMPv6 messages:

- Router–device messages used for dynamic address allocation:
  - Router Solicitation (RS) message
  - Router Advertisement (RA) message
- Device–device messages used for address resolution:
  - Neighbor Solicitation (NS) message
  - Neighbor Advertisement (NA) message
- Router–device messages used for better first-hop selection:
  - Redirect message

The first four messages are new with ICMPv6. Only the Redirect message has a similar counterpart in ICMPv4.

IPv6 devices can be either routers or nodes or both. (*Nodes* are also referred to as *devices*.) Router Solicitation and Router Advertisement messages have to do with on-link communications between routers and devices. Neighbor Solicitation and Neighbor

Advertisement messages are used for on-link communications between any two devices, including a router.

Devices, including routers, use ICMPv6 Neighbor Discovery for the following:

- **Router and prefix discovery:** Router Solicitation and Router Advertisement messages assist a device in automatically determining its network prefix, default gateway, and other configuration information, known as Stateless Address Autoconfiguration (SLAAC). Devices that obtain their addressing information from a stateful DHCPv6 server also use the RA message for default gateway information. Hosts use RS and RA messages to actively search for an alternative when a router or a path to the router fails.
- **Address resolution:** Neighbor Solicitation and Neighbor Advertisement messages assist a device in determining the Layer 2 data link address (typically Ethernet) of another device on its network when it knows its IPv6 address.
- **Duplicate Address Detection (DAD):** Neighbor Solicitation and Neighbor Advertisement messages are used to determine whether a manually or dynamically configured unicast address is already in use by another device.
- **Neighbor Unreachability Detection (NUD):** Neighbor Solicitation and Neighbor Advertisement messages are used to determine whether a neighbor is reachable from the perspective of the device.
- **Redirection:** Routers send Redirect messages to redirect the sender of a packet to a better first-hop or next-hop router. Redirect messages also inform the sender that the destination is on its same subnet and the packet can be sent directly to the destination.

Note Some of the information in this chapter has been discussed in previous chapters and is included in this chapter for completeness.

Neighbor Discovery Options

The five ICMPv6 Neighbor Discovery messages may include one or more options, some of which may appear multiple times in the same message. These options help provide information associated with the various purposes associated with Neighbor Discovery. There are five ICMPv6 Neighbor Discovery options:

- **Type 1—Source Link-Layer Address:** This option contains the Layer 2 address (typically Ethernet) of the sender of the packet. It is used in the Neighbor Solicitation, Router Solicitation, and Router Advertisement messages.
- **Type 2—Target Link-Layer Address:** This option contains the Layer 2 address (typically Ethernet) of the intended target. It is used in Neighbor Advertisement and Redirect messages.
- **Type 3—Prefix Information:** The Prefix Information option provides hosts with prefixes and other information for SLAAC. The Prefix Information option appears in Router Advertisement messages.

- **Type 4—Redirect Header:** This option is used in Redirect messages to contain all or part of the packet that is being redirected.
- **Type 5—MTU:** The MTU option is used in Router Advertisement messages to help ensure that all devices on a link use the same MTU.

You will see how these Neighbor Discovery options are implemented by ICMPv6 messages throughout this chapter.

Default Router and Prefix Determination

Router Solicitation and Router Advertisement messages along with dynamic address allocation have been discussed in previous chapters, most notably in Chapter 9, "Stateless Address Autoconfiguration (SLAAC)." As mentioned previously, IPv6 uses ICMPv6 Router Advertisement messages to suggest to devices how to obtain their IPv6 address information dynamically. Router Advertisement messages are sent periodically by routers or upon receiving a Router Solicitation message. (Chapter 17, "Deploying IPv6 in the Network" discusses using RS and RA messages for transparent fail over at the default gateway.)

This chapter focuses on the message formats of the RS and RA messages.

Router Solicitation Message

A host sends a Router Solicitation message when it needs to know how to dynamically obtain its addressing information. This typically occurs during startup and is the default on most host operating systems.

In Figure 13-1, WinPC sends a Router Solicitation message requesting a Router Advertisement message.

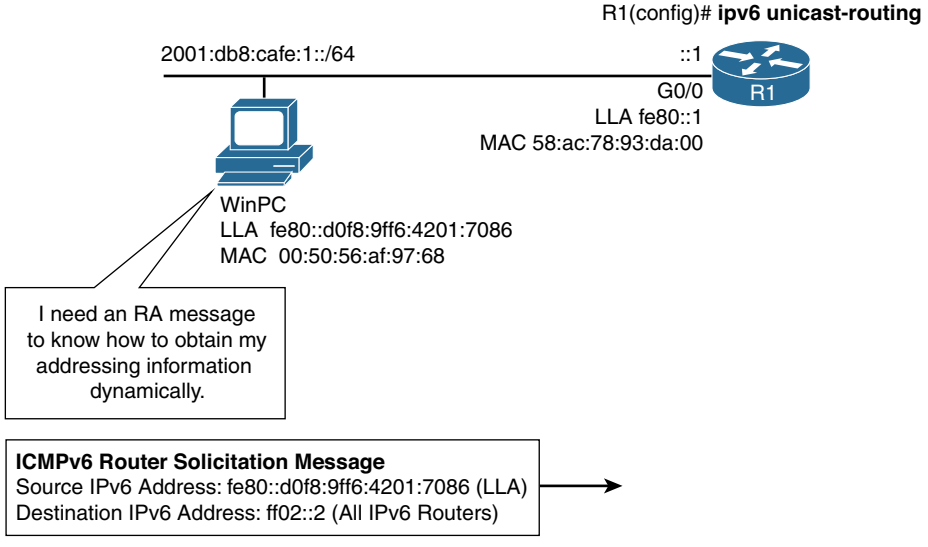


Figure 13-1 *Router Solicitation Message from WinPC*

Figure 13-2 shows the format of a Router Solicitation message. Using Wireshark, Example 13-1 displays an analysis of the RS message sent from WinPC.

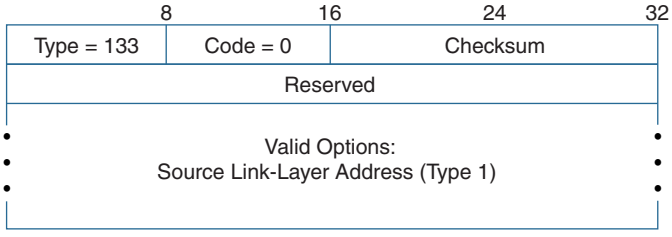


Figure 13-2 *ICMPv6 ND Router Solicitation Message*

Example 13-1 *Router Solicitation from PC1*

```
Ethernet II
  Destination: 33:33:00:00:00:02 (IPv6mcast)
  Source: 00:50:56:af:97:68

Internet Protocol Version 6
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 16
    Next header: ICMPv6 (58)
    Hop limit: 255
    Source: fe80::d0f8:9ff6:4201:7086
    Destination: ff02::2

Internet Control Message Protocol v6
    Type: 133 (Router solicitation)
    Code: 0
    Checksum: 0x3277 [correct]
    ICMPv6 Option (Source link-layer address)
        Type: Source link-layer address (1)
        Length: 1 (8 bytes)
        Link-layer address: 00:50:56:af:97:68
```

By default, WinPC is configured to obtain its addressing information dynamically. WinPC sends a Router Solicitation message to ff02::2, the all-routers multicast address, requesting a Router Advertisement message from any routers on the link. The RS message includes the ICMPv6 Source Link-Layer Address option, which includes the Ethernet source MAC address of WinPC. The router can use this Layer 2 address when responding to WinPC with its Router Advertisement message.

The following describes the Ethernet, IPv6, and ICMPv6 headers in WinPC's Router Solicitation message in Example 13-1.

- Ethernet header:
  - **Destination MAC address (33:33:00:00:00:02):** This is an Ethernet multicast address. 33:33 is the Ethernet multicast address for IPv6. The lower 32 bits, 00:00:00:02, are mapped from the destination IPv6 multicast address, ff02::2.
  - **Source MAC address (00:50:56:af:97:68):** This is the source MAC address of WinPC.
  - **Type field (0x86dd)—not shown in Example 13-1 output:** 0x86dd indicates that the Ethernet data (payload) is an IPv6 packet.
- IPv6 header (significant fields only):
  - **Next Header (58):** A Next Header of 58 indicates that the data (payload) portion of the IPv6 packet is an ICMPv6 message.
  - **Hop Limit (255):** Hop Limit is always set to 255, indicating that the router cannot forward this packet. A router drops an RS message with a Hop Limit other than 255.
  - **Source IPv6 address (fe80::d0f8:9ff6:4201:7086):** The source IPv6 address can be either an IPv6 address already assigned to the sending interface or an unspecified address if no address is assigned. This is typically the host's link-local address.
  - **Destination IPv6 address (ff02::2):** The destination address is the all-routers multicast address, ff02::2. The rightmost 32 bits of the IPv6 multicast address are mapped to the rightmost 32 bits of the Ethernet MAC address, with 33:33 appended to the address.
- ICMPv6 header:
  - **Type (133):** The Type field is set to 133, indicating that this is a Router Solicitation message.
  - **Code (0):** Code is set to 0 and ignored by the receiver.
  - **Checksum (0x3277):** This validates the ICMPv6 header.
  - **Reserved—not shown in Example 13-1 output:** This field is unused.
- Option Type: Source Link-Layer Address
  - **Type (1):** Type field equal to 1 indicates that this is a Source Link-Layer Address option.
  - **Length (1):** This is the length of the option, including the Type and Length fields, in units of 8 octets (bytes). The Source Link-Layer option is 1 unit (or 8 bytes).
  - **Source Link Layer Address (00:50:56:af:97:68):** This is the Layer 2 link-layer (or data link layer) address of the sender. The receiver may use this address when responding to the source of this message. The option is not used when the source IPv6 address is an unspecified address.

Router Advertisement Message

A Cisco router sends Router Advertisement messages every 200 seconds by default, and it also sends an RA message upon receiving a Router Solicitation message from a device. The RA message is a suggestion to devices on the link about how to obtain their addressing information dynamically. This information in the RA message includes prefix, default router, and other configuration information. The RA message is sent to the all-IPv6 devices multicast address, ff02::1. The router can also be configured to send a Router Advertisement message as a unicast when it is in response to a Router Solicitation message.

Before sending RA messages, a router must be configured as an IPv6 router, using the **ipv6 unicast-routing** command. (This also allows the router to enable dynamic IPv6 routing protocols and forward IPv6 packets.)

In Figure 13-3, router R1 sends a Router Advertisement message.

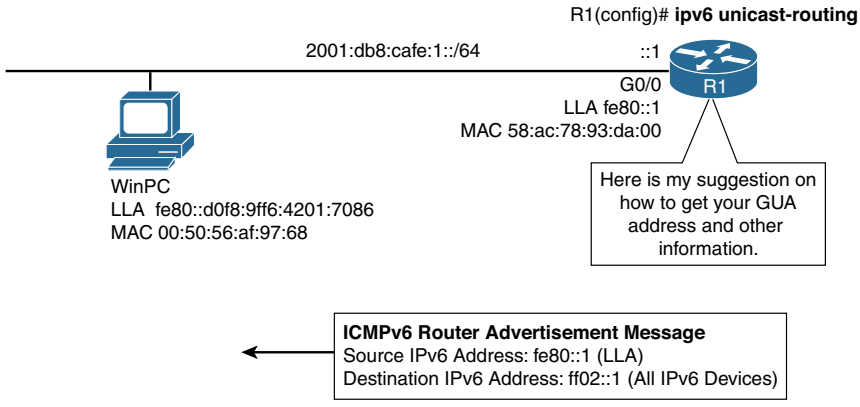


Figure 13-3 *Router Advertisement Message from R1*

Figure 13-4 shows the format of a Router Advertisement, and Example 13-2 provides a Wireshark analysis of router R1's RA message.

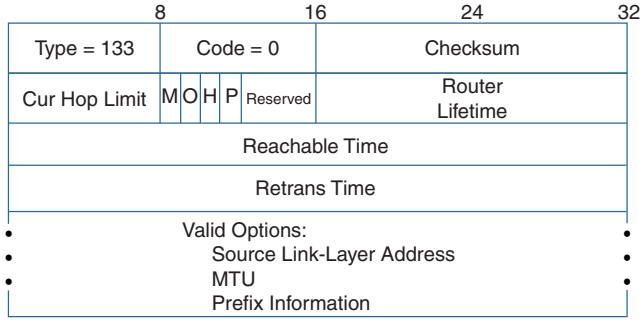


Figure 13-4 *ICMPv6 ND Router Advertisement Message*

Example 13-2 *ND Router Advertisement from Router R1*

```
Ethernet II
  Destination: 33:33:00:00:00:01 (IPv6mcast)
  Source: 58:ac:78:93:da:00


Internet Protocol Version 6
    0110 .... = Version: 6
    .... 1110 0000 .... .... .... .... .... = Traffic class: 0x000000e0
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 64
    Next header: ICMPv6 (58)
    Hop limit: 255
    Source: fe80::1
    Destination: ff02::1

Internet Control Message Protocol v6
    Type: 134 (Router advertisement)
    Code: 0
    Checksum: 0x79aa [correct]
    Cur hop limit: 64
    Flags: 0x00
        0... .... = Managed address configuration: Not set
        .0.. .... = Other configuration: Not set
        ..0. .... = Home Agent: Not set
        ...0 0... = Prf (Default Router Preference): Medium (0)
        .... .0.. = Proxy: Not set
        .... ..0. = Reserved: 0
    Router lifetime (s): 1800
    Reachable time (ms): 0
    Retrans timer (ms): 0
    ICMPv6 Option (Source link-layer address : 58:ac:78:93:da:00)
        Type: Source link-layer address (1)
        Length: 1 (8 bytes)
        Link-layer address: 58:ac:78:93:da:00 (58:ac:78:93:da:00)
    ICMPv6 Option (MTU : 1500)
        Type MTU (5)
        Length: 1 (8 bytes)
        Reserved
        MTU: 1500
    ICMPv6 Option (Prefix information : 2001:db8:cafe:1::/64)
        Type: Prefix information (3)
        Length: 4 (32 bytes)
        Prefix Length: 64
        Flag: 0xc0
```

```
        1... .... = On-link flag(L): Set
         .1.. .... = Autonomous address-configuration flag(A): Set
         ..0. .... = Router address flag(R): Not set
         ...0 0000 = Reserved: 0
      Valid Lifetime: 2592000
      Preferred Lifetime: 604800
      Reserved
      Prefix: 2001:db8:cafe:1:: (2001:db8:cafe:1::)
```

Router R1 responds to WinPC's Router Solicitation message with a Router Advertisement. The RA message is sent to ff02::1, the all-IPv6 devices multicast address. The RA message contains three flags to suggest to WinPC (and other devices on the link) how to obtain IPv6 addressing information:

- **Address Autoconfiguration flag (A flag):** When set to 1 (on), which is the default setting, this flag tells the receiving host to use SLAAC to create its global unicast address. SLAAC allows the host to create its own GUA address by combining the prefix in the RA message with a self-generated Interface ID. The method the host uses to create its own Interface ID depends on the operating system. The two options for creating the Interface ID are:
  - EUI-64 process
  - Random 64-bit value
- **Other Configuration flag (O flag):** When set to 1 (on), this flag tells the host to obtain other addressing information, other than its global unicast address, from a stateless DHCPv6 server. This information may include DNS server addresses and a domain name. The default setting is 0 (off).
- **Managed Address Configuration flag (M flag):** When set to 1 (on), this flag tells a host to use a stateful DHCPv6 server for its global unicast address and all other addressing information. This is similar to DHCP for IPv4. The only information the host uses from the RA message is from the RA's source IPv6 address, which it uses as the default gateway address. The default setting is 0 (off).

Note These flags, along with SLAAC, stateless DHCPv6, and stateful DHCPv6, are discussed in Chapters 9, "Stateless Address Autoconfiguration (SLAAC)," 10, "Stateless DHCPv6," and 11, "Stateful DHCPv6."

In our example, all three flags are set to their default settings. Therefore, WinPC uses the prefix in the RA message to configure its global unicast address using SLAAC. WinPC does not use the services of a stateless or stateful DHCPv6 server.

WinPC uses the source IPv6 address fe80::1 (a link-local address) of the RA message to update its Default Router List, and it uses this address as its default gateway address.

This occurs only if the RA's Router Lifetime field is greater than 0. (The default Router Lifetime is 1800 seconds.) The Default Router List is a list of default gateways that this router can use for reaching devices on networks other than its own.

The following is an explanation of the information in Example 13-2, including the Ethernet header, the IPv6 header, and the ICMPv6 Router Advertisement message. The RA message in this example includes three options—the Source Link-Layer Address, MTU, and Prefix Information options:

- Ethernet header:
  - **Destination MAC address (33:33:00:00:00:01):** The destination MAC address is either a unicast address or a multicast address. If the RA message is in response to an RS message, and the router has been configured to reply to RS messages using unicast, then the destination MAC address is the source MAC address of the device that sent the RS message. (Chapter 9 discusses sending a solicited unicast Router Advertisement.) Otherwise, the destination MAC address is a multicast address, as shown in Example 13-2. The 33:33 indicates that this is an Ethernet multicast address for IPv6. The lower 32 bits, 00:00:00:01, are mapped from the destination IPv6 multicast address, ff02::1.
  - **Source MAC address (58:ac:78:93:da:00):** This is the source MAC address of router R1.
  - **Type field (0x86dd)—not shown in Example 13-2 output:** 0x86dd indicates that the Ethernet data (payload) is an IPv6 packet.
- IPv6 header (significant fields only):
  - **Next Header (58):** A Next Header of 58 indicates that the data (payload) portion of the IPv6 packet is an ICMPv6 message.
  - **Hop Limit (255):** Hop Limit is always set to 255, indicating that the router cannot forward this packet. A router drops an RA message with a Hop Limit other than 255.
  - **Source IPv6 address (fe80::1):** This is the link-local address of the router's egress interface. Devices configured to dynamically receive their IPv6 address information add this address to their Default Router List. WinPC uses this address as its default gateway.
  - **Destination IPv6 address (ff02::1):** The destination address is either the all-devices multicast address, ff02::1, or the source IPv6 address of the device that sent the Router Solicitation message invoking the RA message. (Chapter 9 discusses sending a solicited unicast Router Advertisement.) In this example, router R1 is sending the RA messages using the multicast address. The rightmost 32 bits of this IPv6 multicast address are mapped to the rightmost 32 bits of the Ethernet destination MAC address, and 33:33 is prepended to the address.

- ICMPv6 header:

Note Some of the ICMPv6 header fields are also discussed in Chapter 9 and are described again here for completeness.

- **Type (134):** The Type field is set to 134, indicating that this is a Router Advertisement message.
- **Code (0):** Code is set to 0 and ignored by the receiver.
- **Checksum (0x79aa):** This validates the ICMPv6 header.
- **Cur Hop Limit (64):** The Cur Hop Limit is the value the router recommends for hosts on the network to use as the Hop Limit field in their IPv6 packets. A value of 0 means that the router is not recommending a hop limit and that the host's operating system should determine its own value. The default is 64.
- **Managed Address Configuration flag (M flag) (0):** When set to 1, this tells the host to use stateful configuration (DHCPv6). The default is 0.
- **Other Configuration flag (O flag) (0):** When set to 1, this tells the host that additional information is available from the DHCPv6 server, such as DNS-related information. The default is 0.

Note When both the M and O flags are set to 0, this indicates that there is no information available from a DHCPv6 server.

- **Home Agent (0):** Not set. The Home Agent (H bit) is set in a Router Advertisement to indicate that the router sending this Router Advertisement is also functioning as a Mobile IPv6 home agent on this link.
- **Default Router Preference (medium):** When receiving RA messages from multiple routers, the Default Router Preference (DRP) is used to determine which router to prefer as the default gateway. The preference values are High (01), Medium (00), Low (11), and Reserved (10). If the DRP values are equal, the host uses the source address from the first RA message it received as its default gateway. The default is Medium.
- **Reserved (0):** Not used.
- **Router Lifetime (1800):** This informs a host of the duration, in seconds, that the router should be used as the default gateway. A lifetime of 0 indicates that the router is not a default gateway. The Router Lifetime applies only to the router's function as a default gateway. It does not apply to other information contained in other message fields or options such as prefix and prefix length. The host refreshes its own timer every time it receives a Router Advertisement. The default is 1,800 seconds.

Note A device that is not a router maintains a *Default Router List*. When a device receives a Router Advertisement, it adds the link-local source address of the packet as one of the routers it can use as a default gateway. Each entry has an invalidation timer, the Router Lifetime, extracted from the Router Advertisement; it is used to delete entries that are no longer being advertised.

  - **Reachable Time (0):** This tells a host how long, in milliseconds, it can assume that a neighbor is reachable after receiving reachability confirmation. This is used by Neighbor Unreachability Detection (NUD). A value of 0 means that the router is not specifying a value.
  - **Retrans Timer (0):** This notifies a host of the period of time, in milliseconds, that it should wait before retransmitting Neighbor Solicitation messages. This is used by address resolution and Neighbor Unreachability Detection (NUD).
- ICMPv6 Source Link-Layer Address option:
  - **Type (1):** A Type field 1 indicates that this is a Source Link-Layer Address option.
  - **Length (1):** This is the length of the option, including the Type and Length fields, in units of 8 octets (bytes). The Source Link-Layer Address option is 1 unit (8 bytes).
  - **Source Link Layer Address (58:ac:78:93:da:00):** This is the Layer 2 link layer address of the sender. In Example 13-2, this is the address of router R1's Ethernet MAC address.
- ICMPv6 MTU option:
  - **Type (5):** A Type field equal to 1 indicates that this is an MTU option.
  - **Length (1):** This is the length of the option, including the Type and Length fields, in units of 8 octets (bytes). The MTU option is 1 unit (8 bytes).
  - **MTU (1500):** This informs hosts of the maximum transmission unit (MTU) in bytes for the network. Hosts use this information to maximize the size of the IPv6 packet.
- ICMPv6 Prefix Information option:
  - **Type (3):** A Type field equal to 1 indicates that this is a Prefix Information option.
  - **Length (4):** This is the length of the option, including the Type and Length fields, in units of 8 octets (bytes). The Prefix Information option is 4 units (32 bytes).
  - **Prefix Length (64):** The Prefix Length field provides the information necessary for on-link determination, when combined with the On-Link flag (L flag) in the prefix information option. It indicates the prefix length for on-link prefix determination.
  - **On-Link flag (L flag) (1):** When set to 1, this indicates that the prefix can be used for on-link determination; the prefix advertised in the RA is on this link (subnet). When not set, the advertisement makes no statement about on-link or off-link properties of the prefix. The default is 1.

- **Autonomous Address Configuration flag (A flag):** When set to 1 (on), this flag tells the receiving host to use SLAAC to create its global unicast address. The default is 1.
- **Valid Lifetime (2592000):** This is the length of time an address remains in the valid state. Valid Lifetime must be greater than or equal to Preferred Lifetime. When the Valid Lifetime expires, the address becomes invalid. The default is 2,592,000 seconds (30 days).
- **Preferred Lifetime (604800):** This is the length of time a valid address is preferred until it becomes deprecated. When the Preferred Lifetime expires, the address becomes deprecated. The default is 604,800 seconds (7 days).
- **Prefix (2001:db8:cafe:1::):** This notifies the host of the prefix that is can used for Stateless Address Autoconfiguration.

Note The use of the Preferred Lifetime and Valid Lifetime is discussed in Chapter 9, "Stateless Address Autoconfiguration (SLAAC)."

Address Resolution

Neighbor Solicitation (NS) and Neighbor Advertisement (NA) are two more ICMPv6 Neighbor Discovery messages. These messages are used as follows:

- **Address resolution:** Address resolution in IPv6 is similar to ARP in IPv4. A device sends a Neighbor Solicitation message when it knows the destination IPv6 address but needs to request its Layer 2 address (typically an Ethernet address). This is similar to an ARP Request in IPv4. In response to the Neighbor Solicitation message, the target device sends in a Neighbor Advertisement message, similar to an ARP Reply.

  Address resolution includes **Duplicate Address Detection (DAD)**, which verifies the uniqueness of an address on the link. DAD is much like a gratuitous ARP. The device sends a Neighbor Solicitation message for its own IPv6 address to detect whether another device on the link is using the same address. If a Neighbor Advertisement message is not received, the device knows its address is unique on the link.

- **Neighbor Cache and Neighbor Unreachability Detection (NUD):** IPv6 devices use NS messages and their associated NA messages to build a Neighbor Cache. The Neighbor Cache contains a mapping of IPv6 to Ethernet MAC addresses, similar to an IPv4 ARP cache.

  Neighbor Unreachability Detection (NUD) uses NS and NA messages to detect whether another device is reachable on the link.

Address resolution, DAD, and NUD are similar processes with different purposes. All three involve an understanding of how Neighbor Solicitation and Neighbor Advertisement messages are used for address resolution.

The Address Resolution Process

In Figure 13-5, router R1 has an IPv6 packet to send to WinPC. The destination IPv6 address of the packet is 2001:db8:cafe:1:d0f8:9ff6:4201:7086. Because 2001:db8:cafe:1::/64 is an address on a directly connected network (considered on-link), R1 encapsulates the packet in an Ethernet frame to send it directly to WinPC.

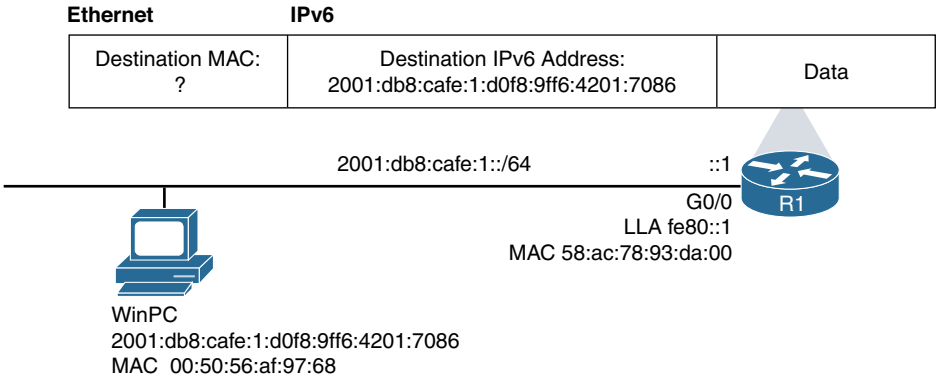


Figure 13-5 *Known IPv6 Address but Unknown Ethernet MAC Address*

Before R1 can send the packet, it must resolve the destination IPv6 address to its Layer 2 address, the Ethernet MAC address of WinPC. R1 puts the packet on hold and performs address resolution using the following steps:

Step 1. R1 checks its Neighbor Cache (or Neighbor Table) to see if there is an entry for the IPv6 address 2001:db8:cafe:1:d0f8:9ff6:4201:7086 and an associated Ethernet MAC address. Similar to an ARP cache, the Neighbor Cache maintains a list of IPv6-to-Ethernet MAC address mappings. In this example, R1 doesn't have this IPv6 address in its Neighbor Cache, so it needs to send a Neighbor Solicitation message, requesting the MAC address from whoever has this IPv6 address. See Figure 13-6.

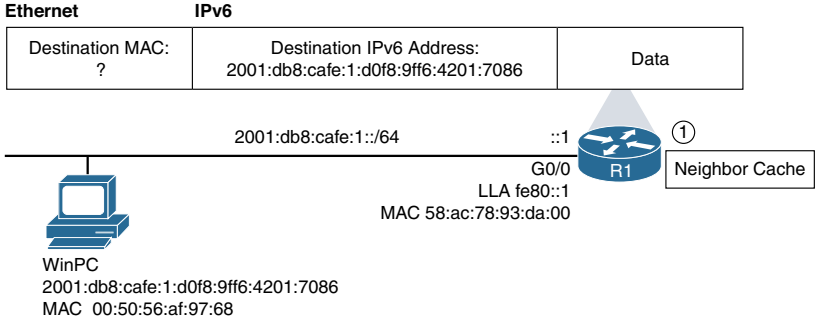


Figure 13-6 *R1 Checks Its Neighbor Cache*

Step 2. R1 sends a Neighbor Solicitation message out its G0/0 interface. The destination IPv6 address is a solicited-node multicast address derived from the Target Address in the ICMPv6 message. The Target Address in the NS message is the destination IPv6 address in the packet, 2001:db8:cafe:1:d0f8:9ff6:4201:7086. The solicited-node multicast address is ff02::1:ff01:7086, which uses the low-order 24 bits of the Target Address. The IPv6 solicited-node address is then mapped to the Ethernet destination MAC address. 33:33 is prepended to the low-order 32 bits of the solicited-node multicast address, which results in an Ethernet multicast address of 33:33:ff:01:70:86. (This mapping is examined in more detail later in this chapter, and you will see the advantage of using an Ethernet multicast address compared to a broadcast address in an IPv4 ARP Request.)

The NS message is a request for whoever has this IPv6 address to reply with their Ethernet MAC address. See Figure 13-7.

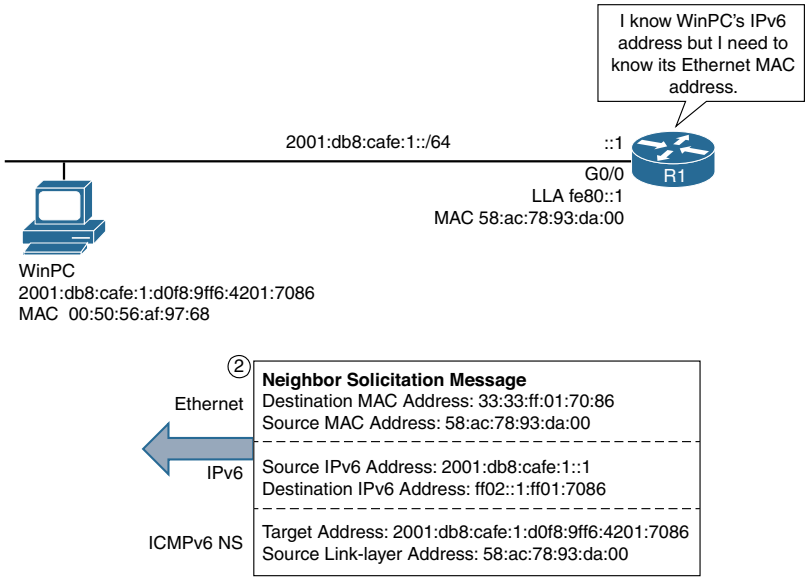


Figure 13-7 *R1 Sends an ICMPv6 Neighbor Solicitation Message*

Step 3. WinPC receives the Neighbor Solicitation message and determines that it is the intended target of the message. It adds the source IPv6 address 2001:db8:cafe:1::1 from the IPv6 header and the link layer address 58:ac:78:93:da:00 from the NS message to its own Neighbor Cache. It then uses this information in its Neighbor Advertisement message to R1. See Figure 13-8.

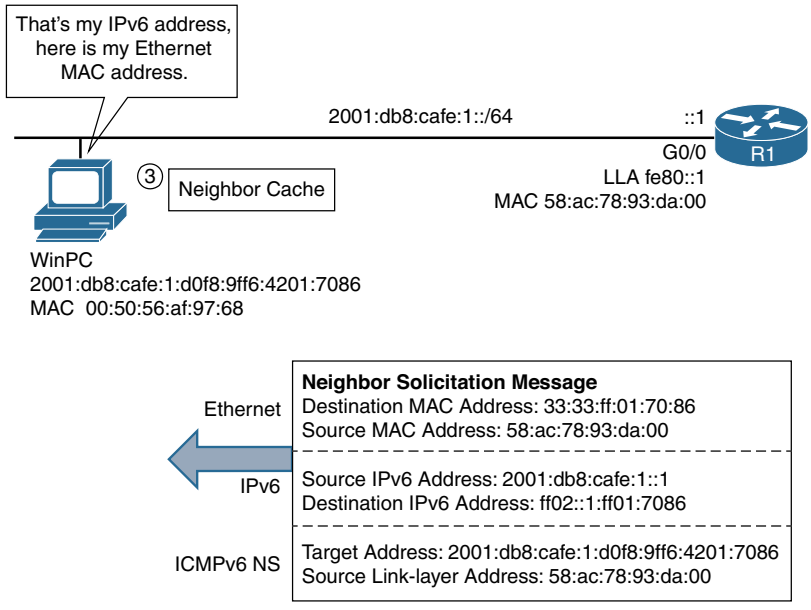


Figure 13-8 *WinPC Is the Target of the NS Message*

Step 4. WinPC responds with a Neighbor Advertisement message. The NA message includes WinPC's IPv6 address, the IPv6 Target Address 2001:db8:cafe:1:d0f8:9ff6:4201:7086, and the Ethernet MAC address 00:50:56:af:97:68. The Neighbor Advertisement is sent as a unicast to R1. See Figure 13-9.

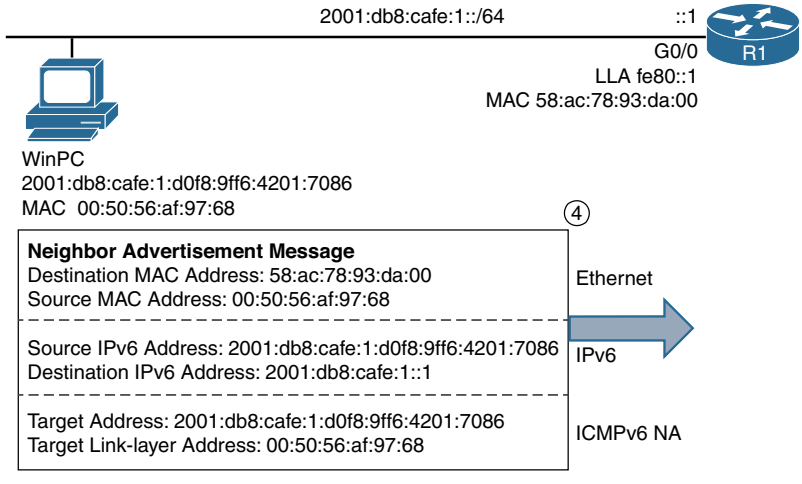


Figure 13-9 *WinPC Responds with an ICMPv6 Neighbor Advertisement Message*

Step 5. R1 receives the Neighbor Advertisement message from WinPC. R1 can now add WinPC's MAC address, 00:50:56:af:97:68, and its associated IPv6 address, 2001:db8:cafe:1:d0f8:9ff6:4201:7086, to its Neighbor Cache. 00:50:56:af:97:68 is included as the destination MAC address in the Ethernet header, and R1 can forward the frame to WinPC. See Figure 13-10.

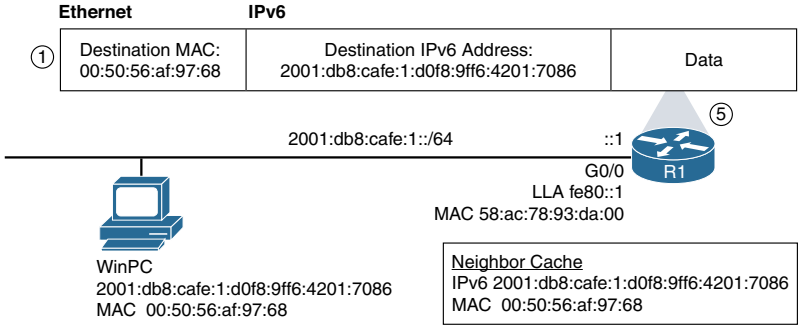


Figure 13-10 *R1 Updates Its Neighbor Cache and Forwards the Frame*

Characteristics of the Neighbor Solicitation Message

As mentioned earlier in this chapter, the Neighbor Solicitation message is similar to an ARP Request in IPv4. Although they have the same purpose, you might have noticed that there are some significant differences. These comparisons are illustrated in Figure 13-11 and listed in Table 13-1.

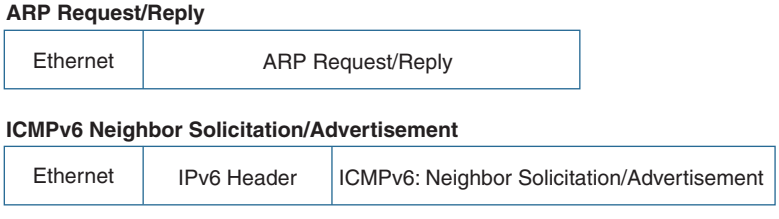


Figure 13-11 *IPv4 ARP and ICMPv6 Address Resolution*

Table 13-1 *Comparing an IPv4 ARP Request and an IPv6 Neighbor Solicitation Message*

	IPv4 ARP Request	IPv6 Neighbor Solicitation
Destination MAC	Broadcast	Multicast
Encapsulated over IP	No	Yes (IPv6)
Destination IP	N/A	Solicited-Node Multicast
Address Resolution Protocol	ARP	ICMPv6

Notice in Figure 13-11 that ARP for IPv4 is carried directly over Ethernet, while ICMPv6 NS/NA messages are carried over IPv6 and then encapsulated in the Ethernet frame. The advantage to using an IPv6 header is that the Neighbor Solicitation message can use an Ethernet multicast address instead of an Ethernet broadcast. Ethernet NICs not only listen for their unicast MAC address (*the burned-in address*) but also listen for any associated multicast MAC addresses.

An ARP Request is an Ethernet broadcast. This means that every device on the subnet must receive and process the message. The Ethernet NIC sees that the destination MAC address is a broadcast, receives the frame, and then passes the data to the ARP process. The ARP process on every device must examine the target IPv4 address in the ARP Request to see if it matches its own IPv4 address. The ARP Request is a message that is intended for only a single device but that must be processed by all devices on the subnet. Only the intended target of the ARP Request needs to respond with an ARP Reply.

Note An excessive number of broadcasts can be detrimental to a network. A broadcast storm can cause extensive disruption and even render the network inoperable. Kevin Wallace has some excellent YouTube videos on broadcast storms and how to prevent them. See Kevin Wallace's YouTube channel.

Multicast allows the Ethernet NIC or, if necessary, the IP process of the device to determine whether it is the intended target of the message in lieu of an upper-layer protocol having to make this decision.

An ICMPv6 Neighbor Solicitation message is encapsulated in an IPv6 header that uses a solicited-node multicast address for the destination IPv6 address. The solicited-node multicast address then maps to an Ethernet multicast MAC address, which allows the Ethernet NIC to filter the frame. Figure 13-12 illustrates the following steps in this process:

Step 1. The IPv6 Target Address in the ICMPv6 Neighbor Solicitation message is the destination IPv6 address of the packet, 2001:db8:cafe:1:d0f8:9ff6:4201:7086. This is the address for which router R1 is in search of the Ethernet MAC address.

Step 2. The low-order 24 bits of the Target Address are copied to a solicited-node multicast address, prepended with the prefix ff02::1:ff00:0000/104. The solicited-node multicast address ff02::1:ff01:7086 is used as the destination IPv6 address in the IPv6 header of the NS message.

Step 3. Here is where we see the benefit of using an IPv6 multicast address. The low-order 32 bits of the destination IPv6 address (the solicited-node IPv6 multicast address) are copied to the low-order 32 bits of the destination MAC address in the Ethernet header. Ethernet prepends the 16 bits 33:33, reserved for IPv6 multicast. This address 33:33:ff:01:70:86 is used as the destination MAC address for the NS message.

Note Solicited-node multicast addresses are covered in Chapter 7, "Multicast Addresses."

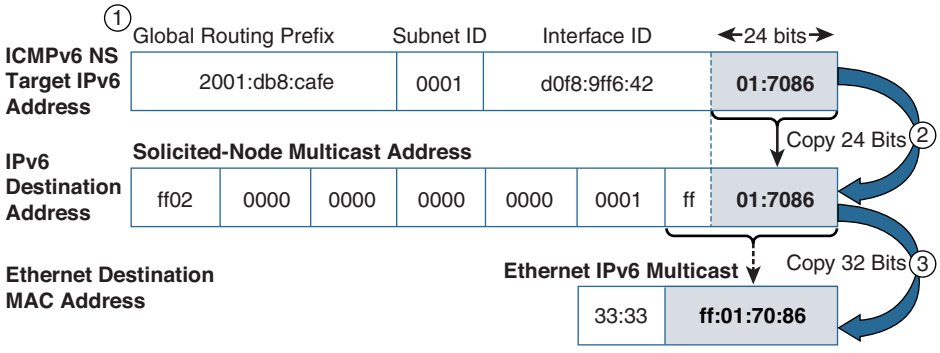


Figure 13-12 *Mapping Unicast Address to Multicast Addresses*

This Ethernet multicast address allows the NIC to determine whether it should pass the encapsulated data to the IPv6 process or whether it can be discarded. As illustrated in Figure 13-13, an Ethernet broadcast must be received by every NIC and then passed to an upper-layer protocol. With an Ethernet multicast, the NIC can accept only Ethernet frames that have a MAC address associated with one of its solicited-node multicast addresses.

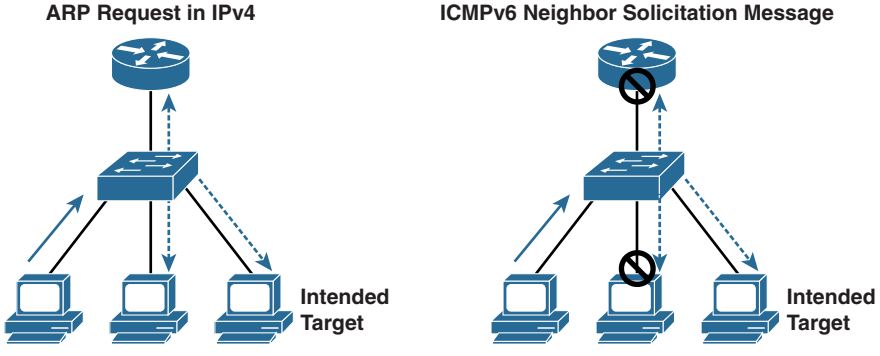


Figure 13-13 *ARP Request Versus ICMPv6 Neighbor Solicitation Message*

Note Every IPv6 unicast address has an associated solicited-node multicast address. Multicast addresses are discussed in Chapter 7.

Note As mentioned in Chapter 7, it is possible that more than one device can have the same solicited-node multicast address; this means that these devices' NICs will accept the same Ethernet multicast addresses. However, only the device that has an IPv6 address that matches the target address in the ICMPv6 message responds with a Neighbor Advertisement.

Format of the Neighbor Solicitation Message

The format of a Neighbor Solicitation message is shown in Figure 13-14. Using the example in Figure 13-7, the Wireshark analysis for router R1's NS message is shown in Example 13-3.

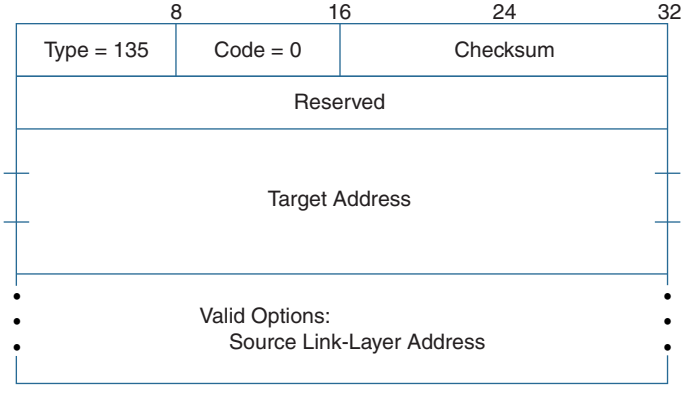


Figure 13-14 *ICMPv6 Neighbor Solicitation Message*

Example 13-3 *ND Neighbor Solicitation from Router R1*

```
Ethernet II
  Destination: 33:33:ff:01:70:86 (IPv6mcast)
  Source: 58:ac:78:93:da:00

Internet Protocol Version 6
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 32
    Next header: ICMPv6 (58)
    Hop limit: 255
    Source: 2001:db8:cafe:1::1
    Destination: ff02::1:ff01:7086

Internet Control Message Protocol v6
    Type: 135 (Neighbor solicitation)
    Code: 0
```

```
Checksum: 0x48ed [correct]
Reserved: 0 (Should always be zero)
Target: 2001:db8:cafe:1:d0f8:9ff6:4201:7086
ICMPv6 Option (Source link-layer address)
    Type: Source link-layer address (1)
    Length: 8 (64 bytes)
    Link-layer address: 58:ac:78:93:da:00
```

The following are the significant fields in the Ethernet, IPv6, and ICMPv6 headers in a Neighbor Solicitation message:

Note The ICMPv6 NS message includes the Source Link-Layer Address option.

- Ethernet header:
  - **Destination MAC address (33:33:ff:01:70:86):** This is an Ethernet multicast address. 33:33 is the Ethernet multicast address for IPv6. The low-order 32 bits ff:01:70:86 are mapped from the low-order 32 bits of the destination IPv6 multicast address, ff02::1:ff01:7086.
  - **Source MAC address (58:ac:78:93:da:00):** This is the source MAC address of R1.
  - **Type field (0x86dd)—not shown in output:** 0x86dd indicates that the Ethernet data (payload) is an IPv6 packet.
- IPv6 header (significant fields only):
  - **Next Header (58):** A Next Header of 58 indicates that the data (payload) portion of the IPv6 packet is an ICMPv6 message.
  - **Hop Limit (255):** Hop Limit is always set to 255, indicating that the router cannot forward this packet. A router drops an NS message with a Hop Limit other than 255.
  - **Source IPv6 address (2001:db8:cafe:1::1):** This is the global unicast address of R1's G0/0 interface. (R1 chooses its GUA address instead of its link-local address on the interface because the NS Target Address is a GUA address.) If DAD is in progress for this address, an unspecified address is used.
  - **Destination address (ff02::1:ff01:7086):** This can be either the solicited-node multicast address corresponding to the target or the target address itself. (NS messages are sent as unicasts during the Neighbor Unreachability Detection [NUD] process. NUD is discussed later in this chapter.) The advantage of a solicited-node multicast address is that it can be mapped to an Ethernet multicast address. In Example 13-3, WinPC is sending it to the solicited-node multicast address. As you saw previously, this address gets mapped to the Ethernet destination MAC address.

- ICMPv6 header:
  - **Type (135):** Set to 135, indicating that this is a Neighbor Solicitation message.
  - **Code (0):** Code is set to 0 and ignored by the receiver.
  - **Checksum (0x48ed):** This validates the ICMPv6 header.
  - **Reserved:** This field is unused.
  - **Target Address (2001:db8:cafe:1:d0f8:9ff6:4201:7086):** This is the IPv6 address that is the target of the solicitation. The sender knows this IPv6 address but not the Layer 2, Ethernet, address. The address cannot be a multicast address. This address is mapped to the IPv6 destination address as a solicited-node multicast address.
- Source Link-Layer Address option:
  - **Type (1):** A Type field of 1 indicates that this is a Source Link-Layer Address option.
  - **Length (1):** This is the length of the option, including the Type and Length fields, in units of 8 octets (bytes). The Source Link-Layer option is 1 unit (8 bytes).
  - **Source Link Layer Address (58:ac:78:93:da:00):** This is the Layer 2, link layer, address of the sender. The receiver may use this address when responding to the source of this message. The option is not used when the source IPv6 address is an unspecified address.

Format of the Neighbor Advertisement Message

The format of a Neighbor Advertisement message is shown in Figure 13-15. Using the example in Figure 13-7, the Wireshark analysis for router R1's NA message is shown in Example 13-4.

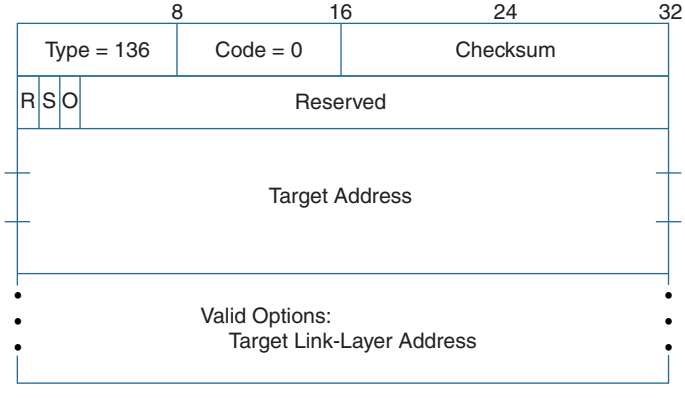


Figure 13-15 *ICMPv6 Neighbor Advertisement Message*

Example 13-4 *ICMPv6 Neighbor Advertisement Message from WinPC*

```
Ethernet II
  Destination: 58:ac:78:93:da:00
  Source: 00:50:56:af:97:68

Internet Protocol Version 6
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 32
    Next header: ICMPv6 (58)
    Hop limit: 255
    Source: 2001:db8:cafe:1:d0f8:9ff6:4201:7086
    Destination: 2001:db8:cafe:1::1

Internet Control Message Protocol v6
    Type: 136 (Neighbor advertisement)
    Code: 0
    Checksum: 0xf621 [correct]
    Flags: 0x60000000
     0... .... .... .... .... .... .... ....= Router: Not set
     .1.. .... .... .... .... .... .... ....= Solicited: Set
     ..1. .... .... .... .... .... .... ....= Override: Set
     ...0 0000 0000 0000 0000 0000 0000 0000= Reserved: 0
    Target: 2001:db8:cafe:1:d0f8:9ff6:4201:7086
    ICMPv6 Option (Target link-layer address)
        Type: Target link-layer address (2)
        Length: 1 (8 bytes)
        Link-layer address: 00:1b:24:04:a2:1e
```

The following are the significant fields in the Ethernet, IPv6, and ICMPv6 headers in a Neighbor Advertisement message. The ICMPv6 NA message includes the Target Link-Layer Address option.

- Ethernet header:
  - **Destination MAC address (58:ac:78:93:da:00):** The destination MAC address is the MAC address of router R1—the originator of the NS message that prompted the NA message.
  - **Source MAC address (00:50:56:af:97:68):** This is the source MAC address of WinPC.
  - **Type field (0x86dd)—not shown in Example 13-4 output:** 0x86dd indicates that the Ethernet data (payload) is an IPv6 packet.

- IPv6 header (significant fields only):
  - **Next Header (58):** A Next Header of 58 indicates that the data (payload) portion of the IPv6 packet is an ICMPv6 message.
  - **Hop limit (255):** Hop Limit is always set to 255, indicating that the router cannot forward this packet. A router drops an NA message with a Hop Limit other than 255.
  - **Source IPv6 address (2001:db8:cafe:1:d0f8:9ff6:4201:7086):** This is the IPv6 address of the sending interface. Because the ICMPv6 Target Address is a global unicast address, WinPC uses its GUA address instead of a link-local address.
  - **Destination address (2001:db8:cafe:1::1):** The destination IPv6 address is the source IPv6 address of R1's NS message that prompted this NA message. If the source address in the corresponding Neighbor Solicitation message was an unspecified unicast address, the destination address is the all-nodes multicast address, ff02::1.
- ICMPv6 header:
  - **Type (136):** This is set to 136, indicating that this is a Neighbor Advertisement message.
  - **Code (0):** Code is set to 0 and ignored by the receiver.
  - **Checksum (0xf621):** This validates the ICMPv6 header.
  - **Router flag (0):** When the Router flag (R-bit) is set to 1, it indicates that the sender is a router. In our example, the NA message was not sent by a router but by the host WinPC. The R-bit is used by Neighbor Unreachability Detection to detect a router that changes to a host.
  - **Solicited flag (1):** When the Solicited flag (S-bit) is set to 1, it indicates that this Neighbor Advertisement message was sent in response to a Neighbor Solicitation. In our example, WinPC is sending the NA message in response to an NS message from router R1. The S-bit is used as a reachability confirmation during Neighbor Unreachability Detection.
  - **Override flag (1):** When the Override flag (O-bit) is set to 1, it indicates that this Neighbor Advertisement should override the existing Neighbor cache entry (equivalent to an IPv4 ARP cache) by updating the cached Layer 2 address for IPv6 address. When it is not set, this advertisement does not update a cached link layer address, but it creates one if one does not already exist. In our example, the receiver of the NA, router R1, updates its neighbor cache.
  - **Reserved (0):** This field is unused.
  - **Target Address (2001:db8:cafe:1:d0f8:9ff6:4201:7086):** When the Neighbor Advertisement is in response to a Neighbor Solicitation, the target address is the IPv6 address found in the Target Address field of the solicitation. In other words, it is the IPv6 address of the device sending this advertisement. For an unsolicited advertisement, the target address is the IPv6 address whose link layer address has changed. In our example, WinPC is including its IPv6 address—the address that R1 knows but for which it needs the associated Ethernet MAC address.

- Target Link-Layer Address option:
  - **Type (2):** A Type field of 2 indicates that this is a Target Link-Layer Address option.
  - **Length (1):** This is the length of the option, including the Type and Length fields, in units of 8 octets (bytes). The Target Link-Layer option is 1 unit (8 bytes).
  - **Target Link-Layer Address (00:50:56:af:97:68):** This is the Layer 2, Ethernet, address of the intended target of the Neighbor Solicitation message. It is the Ethernet MAC address solicited by R1's NS message. In our example, WinPC, as the intended target, is providing its MAC address, 00:50:56:af:97:68.

Router R1 uses the IPv6 Target Address 2001:db8:cafe:1:d0f8:9ff6:4201:7086 and the target link-layer (Ethernet) address 00:50:56:af:97:68 from the ICMPv6 Neighbor Advertisement to update its Neighbor Cache.

Neighbor Cache

The host maintains two caches or tables for each interface:

- Neighbor Cache
- Destination Cache

The *Neighbor Cache* is equivalent to an ARP Cache or an ARP table in IPv4. The Neighbor Cache maintains a list of entries about neighbors to which traffic has recently been sent. Entries include IPv6 unicast addresses and their corresponding Layer 2 addresses, typically Ethernet MAC addresses. The cache also indicates whether the neighbor is a router or a host, the reachability state of the address, whether any queued packets are waiting for address resolution to complete, and the time the next Neighbor Unreachability Detection event is scheduled to take place.

Devices maintain a Neighbor Cache from the information received in Neighbor Advertisement messages. A Neighbor Cache entry can be one of five reachability states (described shortly). The two most common states are Reachable and Stale.

The Reachable state indicates that a packet from this device has recently been received, confirming that this device is reachable. The state transitions from Reachable to Stale when a certain amount of time has elapsed since a packet has been received from this address. In Cisco IOS, the transition time from Reachable to Stale is 30 seconds.

In Example 13-5 you can see this transition in router R1's Neighbor Cache. First, you clear all entries in the Neighbor Cache by using the command **clear ipv6 neighbors**. The **show ipv6 neighbors** command is used to display the Neighbor Cache. After clearing the Neighbor Cache, you can see that the cache is now empty.

Next, you ping WinPC's IPv6 address, 2001:db8:cafe:1:d0f8:9ff6:4201:7086. The Neighbor Cache now includes the entry with WinPC's IPv6 address and its Ethernet MAC address, 0050.56af.9768. The state of the entry is currently Reachable.

The Age column shows the time in minutes that the entry has been in its current state. Continuing with Example 13-5, after 30 seconds, without any further communications, the entry transitions to Stale. After communication is reestablished with WinPC using another **ping** command, the Neighbor Cache state transitions back to Reachable.

Example 13-5 *Displaying the Neighbor Cache*

```
R1# clear ipv6 neighbors

R1# show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface

R1# ping 2001:db8:cafe:1:d0f8:9ff6:4201:7086
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:1:D0F8:9FF6:4201:7086, timeout is 2
  seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/8 ms

R1# show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
2001:DB8:CAFE:1:D0F8:9FF6:4201:7086         0 0050.56af.9768  REACH Gi0/0

R1#

<30 seconds later>

R1# show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
2001:DB8:CAFE:1:D0F8:9FF6:4201:7086         0 0050.56af.9768  STALE Gi0/0

R1# ping 2001:db8:cafe:1:d0f8:9ff6:4201:7086
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:1:D0F8:9FF6:4201:7086, timeout is 2
  seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms

R1# show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
2001:DB8:CAFE:1:D0F8:9FF6:4201:7086         0 0050.56af.9768  REACH Gi0/0
R1#
```

The neighbor cache has five states, as illustrated in Figure 13-16:

- **No Entry Exists (or Deleted):** This is not a Neighbor Cache state. There is no entry in the Neighbor Cache for an IPv6 address and its associated MAC address.
- **Incomplete:** Address resolution is in progress, and the link layer address is not yet known. A Neighbor Solicitation is sent to request the MAC address for the targeted IPv6 address. If three Neighbor Solicitation messages are sent without corresponding Neighbor Advertisements in reply, the Incomplete entry is removed because no entry exists.
- **Reachable:** Packets have recently been received, providing confirmation that this device is reachable. A Neighbor Advertisement message has been received specifying the MAC address associated with the targeted IPv6 address in the previously sent Neighbor Solicitation.
- **Stale:** A certain time period, known as the Reachable time, has elapsed since a packet has been received from this address. The Neighbor Cache also transitions to this state when an unsolicited Neighbor Advertisement has been received. There is no action required by the device.
- **Delay:** Neighbor is pending re-resolution, and traffic might flow to this neighbor. A packet has been sent by the device and is awaiting confirmation by means of a returning packet, such as a TCP Syn/Ack as part of the three-way handshake. If a packet is received within 5 seconds, the entry returns to the Reachable state; otherwise, it transitions to the Probe state.
- **Probe:** Neighbor re-resolution is in progress, and traffic might flow to this neighbor. Similar to an Incomplete state, a Neighbor Solicitation is sent to once again request the MAC address for the targeted IPv6 address. If a Neighbor Advertisement is returned by the solicited device, the entry returns to the Reachable state. If three Neighbor Advertisements are sent without corresponding Neighbor Advertisements in return, the entry is removed.

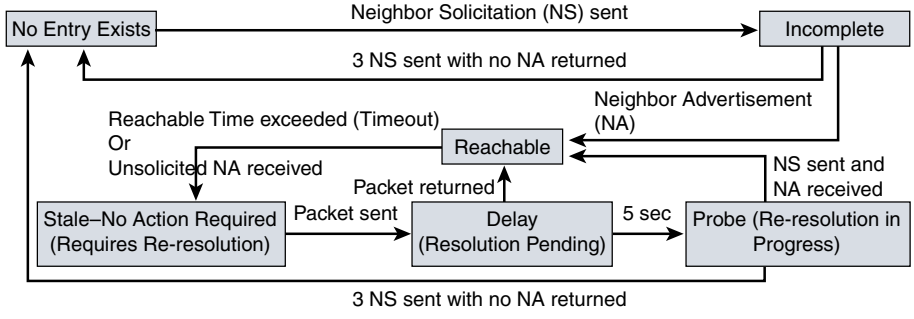


Figure 13-16 *Neighbor Cache States*

You can see the transition for several of these states by using the **debug ipv6 nd** command. Using the same sequence of commands, the debug output in Example 13-6 shows

the transitions from Deleted to Incomplete, Incomplete to Reachable, and then Reachable to Stale. Notice that the corresponding link-layer address (LLA), the MAC address 0050.56af.9768, is also displayed. This is the MAC address associated with the IPv6 address 2001:db8:cafe:1:d0f8:9ff6:4201:7086 in the ping.

Example 13-6 *Displaying the Transition of Neighbor Cache States*

```
R1# clear ipv6 neighbors

R1# show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface

R1# debug ipv6 nd
  ICMP Neighbor Discovery events debugging is on
R1# ping 2001:db8:cafe:1:d0f8:9ff6:4201:7086
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:1:D0F8:9FF6:4201:7086, timeout is 2
  seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/8 ms



*Jan 28 04:38:36.421: ICMPv6-ND: (GigabitEthernet0/0,2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086) Resolution request
*Jan 28 04:38:36.421: ICMPv6-ND: (GigabitEthernet0/0,2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086) DELETE -> INCMP
*Jan 28 04:38:36.421: ICMPv6-ND: (GigabitEthernet0/0,2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086) Sending NS
*Jan 28 04:38:36.421: ICMPv6-ND: (GigabitEthernet0/0,2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086) Queued data for resolution
*Jan 28 04:38:36.429: ICMPv6-ND: (GigabitEthernet0/0,2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086) Received NA from 2001:DB8:CAFE:1:D0F8:9FF6:4201:7086
*Jan 28 04:38:36.429: ICMPv6-ND: (GigabitEthernet0/0,2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086) LLA 0050.56af.9768
*Jan 28 04:38:36.429: ICMPv6-ND: (GigabitEthernet0/0,2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086) INCMP -> REACH

<after 30 seconds>

*Jan 28 04:39:06.579: ICMPv6-ND: (GigabitEthernet0/0,2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086) REACH -> STALE

R1# show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
2001:DB8:CAFE:1:D0F8:9FF6:4201:7086         0 0050.56af.9768  STALE Gi0/0

R1# undebug all
```

To display the neighbor cache on a Windows host, use this command:

```
C:\> netsh interface ipv6 show neighbors
```

To display the neighbor cache on a Linux host, use this command:

```
Linux# ip -6 neighbor show
```

To display the neighbor cache on a MacOS host, use this command:

```
MacOS$ ndp -a
```

In Example 13-7, you can see how the Neighbor Cache state in Figure 13-16 transitions from Stale to being deleted from the cache. In this case, WinPC with the IPv6 address 2001:db8:cafe:1:d0f8:9ff6:4201:7086 has been shut down and can no longer be reached. Router R1 attempts another ping to WinPC.

Notice that the state changes from Stale to Delay pending re-resolution of the address. After 5 seconds, the state changes to Probe, and three Neighbor Solicitation messages are sent in an attempt to resolve this IPv6 address. When R1 doesn't receive a corresponding Neighbor Advertisement, it deletes the entry from the Neighbor Cache. The **show ipv6 neighbors** command verifies that this address is no longer reachable.

Example 13-7 *Displaying the Deletion of a Neighbor Cache Entry*

```
<WinPC, 2001:db8:cafe:1:d0f8:9ff6:4201:7086, has been shut down>


R1# show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
2001:DB8:CAFE:1:D0F8:9FF6:4201:7086         3 0050.56af.9768  STALE Gi0/0


R1# ping 2001:db8:cafe:1:d0f8:9ff6:4201:7086
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:1:D0F8:9FF6:4201:7086, timeout is 2
  seconds:


*Jan 28 15:36:09.791: ICMPv6-ND: (GigabitEthernet0/0, 2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086)
STALE -> DELAY
*Jan 28 15:36:14.855: ICMPv6-ND: (GigabitEthernet0/0, 2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086)
DELAY -> PROBE
*Jan 28 15:36:14.855: ICMPv6-ND: (GigabitEthernet0/0, 2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086)
Sending NS
*Jan 28 15:36:15.943: ICMPv6-ND: (GigabitEthernet0/0, 2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086)
Sending NS
```

```
*Jan 28 15:36:17.031: ICMPv6-ND: (GigabitEthernet0/0, 2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086)
Sending NS
.....
Success rate is 0 percent (0/5)
*Jan 28 15:36:18.119: ICMPv6-ND: PROBE deleted: 2001:DB8:CAFE:1:D0F8:9FF6:4201:7086
*Jan 28 15:36:18.119: ICMPv6-ND: (GigabitEthernet0/0, 2001:DB8:CAFE:1:D0F8:
  9FF6:4201:7086)
PROBE -> DELETE
* Jan 28 15:36:18.119: ICMPv6-ND:  Remove ND cache entry

R1# undebug all

R1# show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface

R1#
```

Note Neighbor Discovery is not performed on serial interfaces, so there is no Neighbor Cache present on these links.

Destination Cache

There is another data structure known as the *Destination Cache*. Like the Neighbor Cache, it maintains a list of the destinations that traffic has been recently sent to, including those on other links or networks. In those cases, the entry is the Layer 2 address of the next-hop router. An IPv6 host maintains a Default Router List from which it selects a router for traffic to off-link destinations. The selected router for a destination is then cached in the Destination Cache.

Example 13-8 shows WinPC pinging an off-link address—the address of one of router R3's interfaces. The **netsh interface ipv6 show destinationcache** command is then used to display the destination cache, which shows the default gateway (next-hop address) that WinPC used to reach destination address.

Example 13-8 *Displaying the Destination Cache on WinPC*

```
WinPC> ping 2001:db8:cafe:4::1
Reply from 2001:db8:cafe:4::1 time=13ms
Reply from 2001:db8:cafe:4::1 time<1ms
Reply from 2001:db8:cafe:4::1 time<1ms
Reply from 2001:db8:cafe:4::1 time<1ms
<output omitted for brevity>

WinPC> netsh interface ipv6 show destinationcache

Interface 11: Local Area Connection

PMTU Destination Address                    Next Hop Address
---- ------------------------------------- ----------------
1500 2001:db8:cafe:4::1                     fe80::1

WinPC>
```

Note The Destination Cache is also updated with information learned from Redirect messages. The Destination Cache can be a subset of the Neighbor Cache. With certain versions of Cisco IOS, the **show ipv6 nd destination** command can be used to display information about IPv6 host-node Destination Cache entries.

Duplicate Address Detection (DAD)

A device can use ICMPv6 Duplicate Address Detection (DAD) to determine whether an address it wants to assign to an interface is already in use by another device. With some exceptions, RFC 4861 recommends that DAD be performed on every unicast address, link-local or global, before it is assigned to an interface—regardless of whether it was assigned using SLAAC, DHCPv6, or manual configuration. There are some exceptions to this behavior, as discussed in RFC 4429, *Optimistic Duplicate Address Detection (DAD) for IPv6*.

If a duplicate address is discovered during this process, it cannot be used on the interface. The procedure for detecting duplicate addresses involves using Neighbor Solicitation and Neighbor Advertisement messages. Figure 13-17 illustrates this process, using WinPC and its link-local unicast address, as described in the following steps:

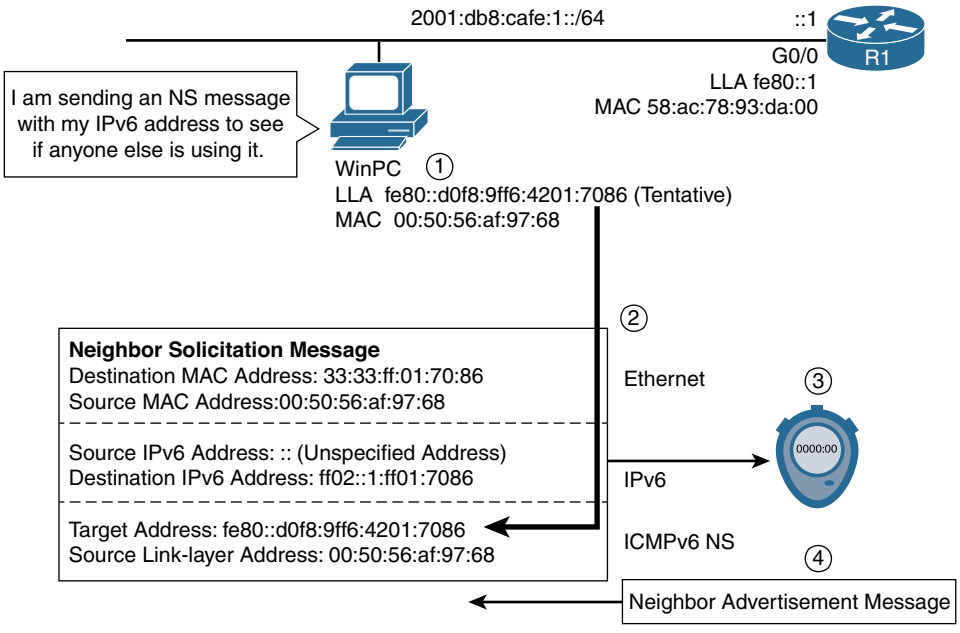


Figure 13-17 *Duplicate Address Detection (DAD)*

Step 1. WinPC automatically creates its own link-local unicast address on its Ethernet interface. The prefix fe80::/10 is prepended to a random 64-bit Interface ID to create the link-local address fe80::d0f8:9ff6:4201:7086. Before using this link-local address, WinPC must determine whether it is unique on the link. The link-local address is put into the tentative state until the DAD procedure is completed.

Step 2. WinPC sends out a Neighbor Solicitation message to see whether anyone else on the link is already using this link-local address. A closer examination of the Neighbor Solicitation message in Figure 13-17 reveals the significant fields:

- Ethernet header:
  - **Destination MAC address:** This is the multicast address 33:33, with the last 32 bits mapped from the 32 low-order bits of the IPv6 destination solicted-node multicast address, ff02::1:ff01:7086. The destination MAC address is, therefore, 33:33:ff:01:70:86.
  - **Source MAC address:** The source MAC address of WinPC is 00:50:56:af:97:68.
- IPv6 header:
  - **Source IPv6 address:** The source IPv6 address is ::, an unspecified source address. When DAD is in progress, the source address is always an unspecified address.

- **Destination IPv6 address:** This is the solicited-node multicast address for WinPC's link-local address, ff02::1:ff01:7086. This address gets mapped to the Ethernet destination MAC address.

- ICMPv6 header (Neighbor Solicitation message):
  - **Target IPv6 address:** This is WinPC's own link-local address, fe80::d0f8:9ff6:4201:7086. If another device is using this address, it needs to respond with a Neighbor Advertisement message.

Step 3. WinPC sets a timer to give an opportunity for another device using this tentative link-local address to respond with a Neighbor Advertisement message. If no message is received after this time expires, the link-local address transitions from tentative to assigned.

Step 4. If another device on the link has the same link-local address, it responds with a Neighbor Advertisement message. The receipt of the NA message tells WinPC that this address is already in use and cannot be assigned. The address is then suspended on WinPC.

Note If the solicited-node multicast address maps to multiple devices (devices with the same low-order 24 bits in their IPv6 unicast address), the devices examine the ICMPv6 Target Address to determine whether they are the intended target. Only the device that has a unicast address that matches the ICMPv6 Target Address needs to respond with a Neighbor Advertisement message.

Note Duplicate Address Detection (DAD) is defined in RFC 4861, *Neighbor Discovery for IP Version 6 (IPv6)*; RFC 4429, *Optimistic Duplicate Address Detection (DAD) for IPv6*; and RFC 7527, *Enhanced Duplicate Address Detection*.

Neighbor Unreachability Detection (NUD)

Neighbor Unreachability Detection (NUD) is defined in RFC 4861, *Neighbor Discovery for IP Version 6 (IPv6)*. Communications between any two IPv6 devices can fail for a number of reasons, such as the host losing power or a faulty cable. Devices actively track the reachability state for the neighbors to which they are sending packets.

The confirmation of reachability can be achieved in one of two ways:

- A Neighbor Advertisement message sent in response to a Neighbor Solicitation message.
- An upper-layer process indicating a successful connection, such as acknowledgments in an active TCP connection.

When a path to a neighbor appears to be failing, the specific recovery procedure depends on how the neighbor is being used. If the neighbor is the ultimate destination, address resolution should be performed again. This involves sending a Neighbor Solicitation message and awaiting a Neighbor Advertisement in response. If the neighbor is a router, it would be more appropriate for the device to use a different default gateway. Neighbor Unreachability Detection is performed only with neighbors where unicast packets are sent.

This chapter has already discussed Neighbor Cache states, and Figure 13-16 illustrates the various states and events that transpire between them. Neighbor Unreachability Detection uses these same states and events to detect and resolve neighbor unreachability issues.

NUD can have a significant impact on the control plane processing of a first-hop router with a large number of devices. The router will have a large Neighbor Cache, which generates a substantial amount of NS and NA traffic that the router's CPU will need to process.

Redirect Message

An ICMPv6 Redirect message is used to inform a device that there is a better first-hop router. As with IPv4, a router informs the originating host of the IPv6 packet if there is another router on the same local link but closer to the destination. Figure 13-18 shows this process:

Step 1. PC-A has several packets to send to a device on a remote network, network X. PC-A sends the first packet to router R1, its default gateway.

Step 2. After checking its routing table, router R1 forwards this packet to router R2.

Step 3. Router R1 sees that it has forwarded the packet to R2 out the same interface on which it received it from PC-A. R1 sends an ICMPv6 Redirect message to PC-A, suggesting that PC-A use the better first-hop router R2.

Step 4. PC-A receives the Redirect message and sends subsequent messages directly to router R2.

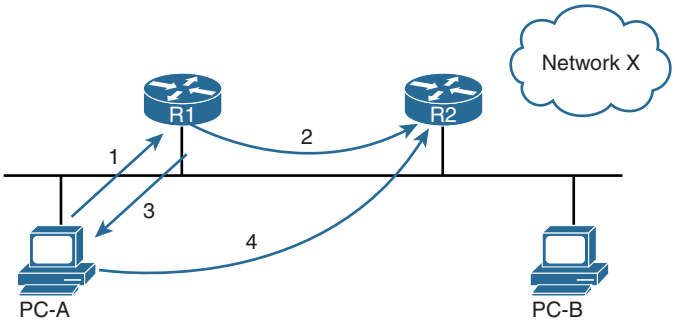


Figure 13-18 *ICMPv6 Redirect Message for an Off-Link Device*

Unlike in IPv4, the router's ICMPv6 Redirect message can also inform the originating host if the destination host (on a different prefix/network) is on the same link as itself.

Figure 13-19 shows an example of router R1 redirecting an IPv6 packet from PC-A for a device on the same link as the source, PC-B. The process is similar to the steps just described.

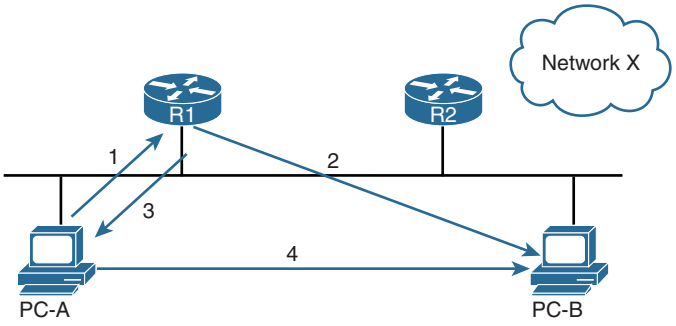


Figure 13-19 *ICMPv6 Redirect Message for an On-Link Device*

Figure 13-20 shows the format of an ICMPv6 Redirect message.

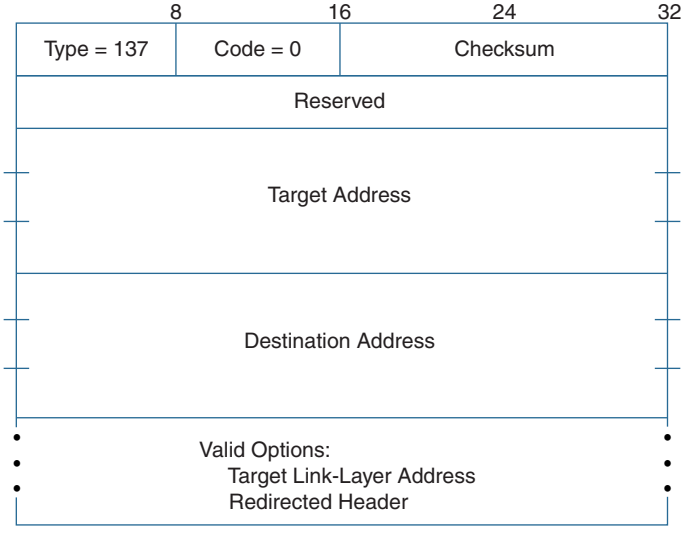


Figure 13-20 *ICMPv6 Redirect Message*

The fields in the IPv6 and ICMPv6 headers are as follows:

- IPv6 header:
  - **Source address:** This is the link-local address assigned to the interface.
  - **Destination address:** This is the source address of the packet that triggered the Redirect message.
  - **Hop limit:** Hop Limit is always set to 255.

- ICMPv6 header:
  - **Type:** This is set to 137, indicating that this is a Redirect message.
  - **Code:** Code is set to 0 and ignored by the receiver.
  - **Checksum:** This validates the ICMPv6 header.
  - **Reserved:** This field is unused.
  - **Target Address:** This is the IPv6 address that is a better first hop to use.
  - **Destination Address:** This is the IPv6 address of the destination that has been redirected to the target address.
- Target Link-Layer Address option:
  - **Type (2):** A Type field set to 2 indicates that this is a Target Link-Layer Address option.
  - **Length (1):** This is the length of the option, including the Type and Length fields, in units of 8 octets (bytes). The Target Link-Layer option is 1 unit (8 bytes).
  - **Target Link Layer Address:** This is the link layer address of the target address, the recommended next-hop router. This prevents the host from having to resolve the link layer address of the other router.
- Redirected Header option:
  - **Type:** A Type field set to 4 indicates that this is a Redirected Header option.
  - **Length:** This is the length of the option, including the Type and Length fields, in units of 8 octets (bytes). The Redirected Header option is 1 byte.
  - **IPv6 Header + Data:** This is the original packet truncated to ensure that the size of the redirect message does not exceed the minimum MTU required to support IPv6, as specified in IPv6.

Summary

ICMPv6 Neighbor Discovery includes processes similar to those used in IPv4, such as ARP, ICMP Router Discovery, and Redirect, but with significant differences. ND also includes new functionality, such as Duplicate Address Detection (DAD) and Neighbor Unreachability Detection (NUD).

Neighbor Discovery uses five ICMPv6 messages:

- **Router Solicitation message:** A host sends a Router Solicitation message when it needs to know how to obtain its addressing information dynamically.
- **Router Advertisement message:** Router Advertisement messages are sent periodically and in response to Router Solicitation messages. This type of message provides hosts with addressing and other configuration information. SLAAC uses the prefix in the RA to create a GUA address and for other network information. The default

gateway address can only be obtained dynamically from the source IPv6 address of the RA message. DHCPv6 does not provide this information.

- **Neighbor Solicitation message:** Neighbor Solicitation and Neighbor Advertisement messages are very similar to ARP Requests and ARP Replies in IPv4. A Neighbor Solicitation is sent to request the Layer 2 address (typically an Ethernet MAC address) of a target device while also providing its own link layer address to the target.
- **Neighbor Advertisement message:** A Neighbor Advertisement is sent in response to a Neighbor Solicitation, providing the Layer 2 address (typically a MAC address) of an IPv6 address.
- **Redirect message:** An ICMPv6 Redirect message is used to inform a device that there is a better first-hop router. It works the same as the Redirect message in IPv4.

These ND messages are used by the following tables and processes:

- **Neighbor Cache:** This is equivalent to an ARP Cache in IPv4. The Neighbor Cache maintains a list of information, primarily IPv6 addresses and corresponding Ethernet MAC addresses, to which traffic has been recently sent.
- **Destination Cache:** Like the Neighbor Cache, the Destination Cache maintains a list of destinations to which traffic has been recently sent, including those on other links or networks. An IPv6 host maintains a Default Router List, from which it selects a router for traffic to off-link destinations. The selected router for a destination is then cached in the Destination Cache, including the remote address and next-hop address (default gateway address).
- **Address resolution:** Address resolution uses Neighbor Solicitation and Neighbor Advertisement messages to resolve a Layer 3 address to a corresponding Layer 2 (Ethernet MAC) address. This is similar to ARP in IPv4.
- **Duplicate Address Detection (DAD):** A device uses DAD to determine whether an address it wants to use is already in use by another device. DAD uses Neighbor Solicitation with the device's own address. If another device is already using this address, it responds with a Neighbor Advertisement message.
- **Neighbor Unreachability Detection (NUD):** Devices actively track the reachability states for the neighbors to which they are sending packets. Devices use Neighbor Solicitation and Neighbor Advertisement messages to make this determination.

Review Questions

1. What type of message is used to send a device's own Ethernet MAC address to resolve a known IPv6 address?
   a. Router Solicitation
   b. Router Advertisement
   c. Neighbor Solicitation
   d. Neighbor Advertisement
   e. Redirect

2. What type of message is used to request information on obtaining addressing information dynamically?
   a. Router Solicitation
   b. Router Advertisement
   c. Neighbor Solicitation
   d. Neighbor Advertisement
   e. Redirect
3. What type of message is used to request an Ethernet MAC address for a known IPv6 address?
   a. Router Solicitation
   b. Router Advertisement
   c. Neighbor Solicitation
   d. Neighbor Advertisement
   e. Redirect
4. What type of message informs a host of a better next-hop router, closer to the destination of the packet?
   a. Router Solicitation
   b. Router Advertisement
   c. Neighbor Solicitation
   d. Neighbor Advertisement
   e. Redirect
5. What type of message is used to suggest to a device how to obtain addressing information dynamically?
   a. Router Solicitation
   b. Router Advertisement
   c. Neighbor Solicitation
   d. Neighbor Advertisement
   e. Redirect
6. What is the destination IPv6 address of a Router Solicitation message?
   a. Solicited-node multicast address
   b. All-IPv6 devices multicast address (ff02::1)
   c. All-IPv6 routers multicast address (ff02::2)
   d. Link-local address of the destination
   e. Global unicast address of the destination
7. By default, what is the destination IPv6 address of a Cisco IOS Router Advertisement message?
   a. Solicited-node multicast address
   b. All-IPv6 devices multicast address (ff02::1)
   c. All-IPv6 routers multicast address (ff02::2)
   d. Source IPv6 address of the Router Solicitation message

8. What is the destination IPv6 address of a Neighbor Solicitation message that can also be mapped to an Ethernet multicast address?
   a. Solicited-node multicast address
   b. All-IPv6 devices multicast address (ff02::1)
   c. All-IPv6 routers multicast address (ff02::2)
   d. Target unicast address
9. Which ICMPv6 option is used in a Neighbor Solicitation message and includes the Layer 2 address of the sender?
   a. Source Link-Layer Address
   b. Target Link-Layer Address
   c. Prefix Information
   d. Redirect Header
   e. MTU
10. Which ICMPv6 options are used by a Router Advertisement message? (Choose all that apply.)
    a. Source Link-Layer Address
    b. Target Link-Layer Address
    c. Prefix Information
    d. Redirect Header
    e. MTU
11. What is the state of an entry in the neighbor cache when a packet has recently been received?
    a. Stale
    b. On (1)
    c. Established
    d. Reachable
    e. There is no state, but the addresses are displayed in the neighbor cache.
12. What is the state of an entry in a router's neighbor cache if it has not received a packet from this address in 30 seconds?
    a. Stale
    b. Off (0)
    c. Delayed
    d. Reachable
    e. There is no state, but the entry is removed from the neighbor cache.
13. What ICMPv6 process ensures that an IPv6 address is unique before it is assigned to an interface?
    a. Unique Neighbor Detection
    b. Neighbor Unreachability Detection
    c. Duplicate Address Detection
    d. Gratuitous Neighbor Solicitation

14. What type of message does a device send to determine if its IPv6 address is already in use by another device on the link?
    a. Router Solicitation
    b. Router Advertisement
    d. Neighbor Solicitation
    e. Neighbor Advertisement
    f. Redirect
15. What is the difference between the destination MAC of an ARP Request and ICMPv6 Neighbor Solicitation message?

References

RFCs

RFC 3122, *Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification*, A. Conta, Transwitch Corp., www.ietf.org/rfc/rfc3122.txt, June 2001.

RFC 3775, *Mobility Support in IPv6*, D. Johnson, Nokia, www.ietf.org/rfc/rfc3775.txt, June 2004.

RFC 4286, *Multicast Router Discovery*, B. Haberman, JHU APL, www.ietf.org/rfc/rfc4286.txt, December 2005.

RFC 4429, *Optimistic Duplicate Address Detection (DAD) for IPv6,* N. Moore, Monash University CTIE, www.ietf.org/rfc/rfc4429.txt, April 2006.

RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, A. Conta, Transwitch, www.ietf.org/rfc/rfc4443.txt, March 2006.

RFC 4861, *Neighbor Discovery for IP Version 6 (IPv6)*, T. Narten, IBM, www.ietf.org/rfc/rfc4861.txt, September 2007.

RFC 7527, *Enhanced Duplicate Address Detection*, R. Asati, Cisco Systems, www.ietf.org/rfc/rfc7527.txt, April 2015.

This page intentionally left blank

Part V

Routing IPv6

This page intentionally left blank

Chapter 14

IPv6 Routing Table and Static Routes

The previous chapters examine IPv6 and related protocols, such as ICMPv6 and ICMPv6 Neighbor Discovery protocol. Those chapters also discuss the different IPv6 address types, including global unicast, link-local, and multicast addresses. This chapter and the next two examine the routing of IPv6 packets, including static routing, EIGRP for IPv6, and OSPFv3.

Routing IPv6 is very similar to routing IPv4. The protocols have simply been redesigned or extended to support IPv6. The objective of this chapter and the next two is to highlight the operational differences between routing IPv4 and IPv6, along with the differences in configuration and verification commands. These chapters assume some knowledge of routing IPv4.

This chapter examines the following topics associated with routing IPv6:

- Configuring a router as an IPv6 router
- Understanding the IPv6 routing table
- Configuring and verifying IPv6 static routes
- Cisco Express Forwarding (CEF) for IPv6

IPv6 has been implemented in all the major dynamic routing protocols used today, including the following:

- Routing Information Protocol next generation (RIPng), RFC 2080, *RIPng for IPv6*
- Enhanced Interior Gateway Routing Protocol (EIGRP) for IPv6, RFC 7868, *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP) for IPv6*
- Open Shortest Path First version 3 (OSPFv3), RFC 5340, *OSPF for IPv6*
- Intermediate System to Intermediate System for IPv6, RFC 5308, *Routing IPv6 with IS-IS*

- Border Gateway Protocol (BGP) for IPv6, RFC 4760, *Multiprotocol Extensions for BGP-4*

The next two chapters discuss two of the most common Interior Gateway Protocols (IGPs) for IPv6:

- Chapter 15, "EIGRP for IPv6"
- Chapter 16, "OSPFv3"

Note For more information on these and other routing protocols (such as IS-IS) for both IPv4 and IPv6, along with multicast routing, see the excellent Cisco Press book *IP Routing on Cisco IOS, IOS XE, and IOS XR: An Essential Guide to Understanding and Implementing IP Routing Protocols*, by Brad Edgework, Aaron Foss, and Ramiro Rios.

Configuring a Router as an IPv6 Router

Before configuring an IPv6 dynamic routing protocol and before a router can forward IPv6 packets received on one of its interfaces, the router must be configured as an *IPv6 router*.

Most IPv6 routers typically use a combination of dynamic routing and static routes. IPv6 routers also announce prefix, prefix length, default gateway, and other configuration information using ICMPv6 Router Advertisements. To enable a router as an IPv6 router, you need to ensure that it has an IPv6 address on an interface and use the following global configuration command:

```
Router(config)# ipv6 unicast-routing
```

An IPv6 router—a router configured with **ipv6 unicast-routing**—has the following characteristics:

- It is a member of the all-IPv6 routers multicast group, ff02::2, shown in Example 14-1.
- It can be enabled with an IPv6 dynamic routing protocol.
- It forwards IPv6 packets transiting the router.
- It sends ICMPv6 Router Advertisements on Ethernet (and FDDI) interfaces.

Router interfaces can be configured with an IPv6 address or can use the **ipv6 enable** interface command without being configured with the **ipv6 unicast-routing** command. This essentially makes the router into an IPv6 host and a member of the all-IPv6 devices multicast group, ff02::1. The **ipv6 unicast-routing** command and an interface with an IPv6 address are required to make the router an IPv6 router. An IPv6 router is also a member of the all-IPv6 routers multicast group, ff02::2.

Note Packets that transit an IPv6 router are packets that did not originate from the router itself. In other words, these packets were received on one of the router's interfaces. A router *not* enabled as an IPv6 router can be configured with IPv6 static routes, and it will forward those packets but only if it is the source of the packets.

The **ip routing** command for IPv4 is enabled by default on Cisco IOS. So why isn't the similar **ipv6 unicast-routing** command enabled by default on Cisco IOS? The difference is that the **ipv6 unicast-routing** command causes the router to send Router Advertisement messages on Ethernet interfaces. By default, most host operating systems receiving these RA messages configure global unicast addresses using SLAAC. For security reasons, network administrators need to be aware of the impact this will have on their networks.

Figure 14-1 shows the topology of the examples used in this chapter. At this point, the routers have been configured with global unicast and link-local unicast addresses. For simplicity, each router has the same link-local address configured on both of its interfaces. Static and dynamic routing have not been configured.

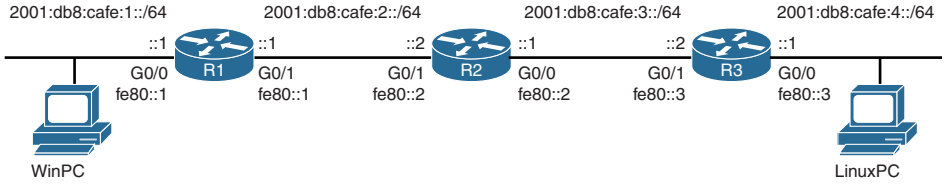


Figure 14-1 *Topology for an IPv6 Routing Table*

Note RFC 7404, Using Only Link-Local Addressing inside an IPv6 Network, discusses implementing routing protocols using only link-local addresses on infrastructure links.

Example 14-1 shows router R1 configured as an IPv6 router with the **ipv6 unicast-routing** command, and verified using the **show ipv6 interface gigabitethernet 0/0** command.

Example 14-1 *Verification of Router R1 as an IPv6 Router*

```
R1(config)# ipv6 unicast-routing
R1(config)# exit
R1# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::1
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1::1, subnet is 2001:DB8:CAFE:1::/64
```

```
 Joined group address(es):
    FF02::1
    FF02::2
    FF02::FB
    FF02::1:FF00:1
<output omitted for brevity>
```

Example 14-2 show routers R2 and R3 enabled as IPv6 routers.

Example 14-2 *Enabling IPv6 Routing with the* **ipv6 unicast-routing** *Command on R2 and R3*

```
R2(config)# ipv6 unicast-routing
------------------------------------------------------------
R3(config)# ipv6 unicast-routing
```

Understanding the IPv6 Routing Table

Cisco IOS maintains separate routing tables for IPv4 and IPv6. In many ways, an IPv6 routing table is easier to understand than an IPv4 routing table. An IPv4 routing table is constructed in a classful manner: It is organized by classful network addresses, with any subnets listed below a classful entry. With IPv6, classful networks and Variable Length Subnet Masking (VLSM) are no longer concerns—which means an IPv6 routing table is much more straightforward. Because classful and classless addressing are nonexistent in IPv6, IPv6 routes are simply displayed as individual prefixes.

The **show ipv6 route** command is used to display the contents of the IPv6 routing table. As shown in the following syntax, many options can be used with the **show ipv6 route** command:

```
Router# show ipv6 route [ipv6-address | ipv6-prefix/prefix-length [ longer-prefixes ] |
[ protocol ] | [ repair] | [ updated [ boot-up ] [ day month ] [ time ] ] | interface
type number | nd | nsf | table table-id | watch ]
```

The most common options in this syntax are discussed in this section.

Note All these syntax options are described in *Cisco IOS IPv6 Command Reference*, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book.html.

The **show ipv6 route** command provides output similar to that of the **show ip route** command in IPv4, except that the information is IPv6-specific.

When the *ipv6-address* or *ipv6-prefix/prefix-length* argument is specified, the longest match lookup is performed from the routing table, and only route information for that address or prefix is displayed, along with more detailed information about the route. When a routing protocol is specified, only routes for that protocol are displayed. Specifying the **connected**, **local**, **mobile**, or **static** keyword displays only those routes

with that specific code. If the **interface** keyword and *type* and *number* arguments are specified, only routes for the specified interface are displayed. You will see examples for most of these options throughout this chapter.

The IPv4 **show ip route** command includes a timestamp with each route—the last time the route was updated, in hours:minutes:seconds. By default, the timestamp is not displayed in the IPv6 routing table. The **updated** option must be included—**show ipv6 route updated**—to display the last time the route was updated in hours:minutes:seconds, day, month, and year.

The **show ipv6 route** command in Example 14-3 displays R1's IPv6 routing table. The partial output from the **show ipv6 route updated** command includes the last time the route was updated.

Example 14-3 *Displaying R1's IPv6 Routing Table*

```
R1# show ipv6 route
IPv6 Routing Table - default - 5 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, I1 - ISIS L1, I2 - ISIS L2
       IA - ISIS interarea, IS - ISIS summary, D - EIGRP, EX - EIGRP external
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, a - Application
C   2001:DB8:CAFE:1::/64 [0/0]
     via GigabitEthernet0/0, directly connected
L   2001:DB8:CAFE:1::1/128 [0/0]
     via GigabitEthernet0/0, receive
C   2001:DB8:CAFE:2::/64 [0/0]
     via GigabitEthernet0/1, directly connected
L   2001:DB8:CAFE:2::1/128 [0/0]
     via GigabitEthernet0/1, receive
L   FF00::/8 [0/0]
     via Null0, receive
R1#
R1# show ipv6 route updated
IPv6 Routing Table - default - 5 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity>
C   2001:DB8:CAFE:1::/64 [0/0]
     via GigabitEthernet0/0, directly connected
      Last updated 16:01:38 05 February 2017
L   2001:DB8:CAFE:1::1/128 [0/0]
     via GigabitEthernet0/0, receive
      Last updated 16:01:39 05 February 2017
<output omitted for brevity>
R1#
```

Immediately following the **show ipv6 route** command in Example 14-3, the first line of output displays the number of entries in the routing table:

```
IPv6 Routing Table - default - 5 entries
```

This is followed by codes indicating the protocol that derived the route. After the codes are entries or routes that populate the contents of the IPv6 routing table. Many of these codes are similar to the same codes in an IPv4 routing table. They indicate the source of the routing table entry, such as S for a static route or D for EIGRP.

Note You may be wondering why the code D is used for EIGRP instead an E. The predecessor to the Border Gateway Protocol (BGP) routing protocol was the Exterior Gateway Protocol (EGP). The routing table used the code E for EGP. So, why does the routing table use a D for EIGRP? EIGRP uses Diffusing Update Algorithm (DUAL) to calculate the shortest, loop-free path to destination networks. Therefore, the D for DUAL is used for EIGRP.

Codes: NDp and ND

There are two codes specific to the IPv6 routing table: NDp and ND. NDp indicates that the network prefix was learned using the Neighbor Discovery Protocol, and ND indicates that the default route was learned via Neighbor Discovery.

Digressing for a moment from our previous topology, Figure 14-2 shows two routers, HQ and BRANCH. The HQ router is configured with the **ipv6 unicast-routing** command and sends ICMPv6 Router Advertisement messages out its G0/0 interface.

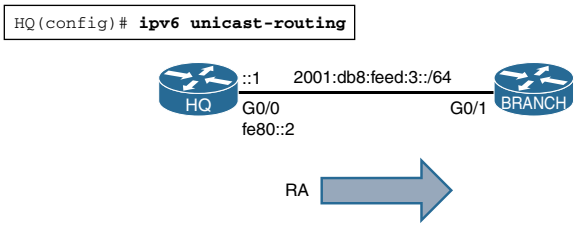


Figure 14-2 *Topology for NDp and ND*

The BRANCH router's G0/1 interface is configured with the following commands:

```
BRANCH(config)# interface gigabitethernet 0/1
BRANCH(config-if)# ipv6 address autoconfig default
```

The **ipv6 address autoconfig** command causes the BRANCH router to use the prefix in the RA message and EUI-64 to create a GUA address on its G0/1 interface. The **default** option creates an additional default route entry via HQ in the BRANCH's IPv6 routing table, as shown in Example 14-4.

Example 14-4 *Displaying BRANCH's IPv6 Routing Table*

```
BRANCH# show ipv6 route
IPv6 Routing Table - default - 6 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, I1 - ISIS L1, I2 - ISIS L2
       IA - ISIS interarea, IS - ISIS summary, D - EIGRP, EX - EIGRP external
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, a - Application
ND   ::/0 [2/0]
     via FE80::2, GigabitEthernet0/1
NDp 2001:DB8:FEED:3::/64 [2/0]
     via GigabitEthernet0/1, directly connected
L   2001:DB8:FEED:3:662:73FF:FE5E:F901/128 [0/0]
     via GigabitEthernet0/1, receive
BRANCH#
```

Example 14-4 shows the following entries in BRANCH's IPv6 routing table:

- **ND ::/0 [2/0]:** This is a default route via HQ, created by the **default** option in the **ipv6 address autoconfig** command. The second line of the output shows the next-hop address, fe80::2. This is HQ's link-local address on its G0/0 interface, which was the source IPv6 address of the RA message. The second line of the output also shows "via GigabitEthernet 0/1," which is the exit interface BRANCH uses to forward packets using this routing entry.
- **NDp 2001:DB8:FEED:3::/64 [2/0]:** This is similar to a connected network (discussed next) created by the **ipv6 address autoconfig** command. The router learns the prefix 2001:db8:feed:3::/64 from HQ's Router Advertisement message. The second line of the output, "via GigabitEthernet0/1, directly connected," indicates that the interface is directly connected to this network.
- **L 2001:DB8:FEED:3:662:73FF:FE5E:F901/128 [0/0]:** This is a local route, which is the global unicast address of BRANCH's G0/1 interface. The GUA address was generated using SLAAC, with the prefix learned from HQ's RA message and the Interface ID generated using EUI-64. The second line of the output, "via GigabitEthernet0/1, receive," indicates that the router receives and processes any packets for this address as a destination device and does not attempt to route the packets. Local routes are discussed shortly.

Note The code "NDr - Redirect" indicates that this entry was learned from an ICMPv6 Redirect message from another router.

Code: Connected

In the topology shown in Figure 14-1 and in R1's IPv6 routing table in Example 14-3, you can see five entries. Five entries might seem a little strange because R1 has only two interfaces, and there are no static or dynamic routes. Each interface has two entries with two different codes: an entry C for connected (or directly connected network) and an entry L for local. The fifth entry is a local route for the multicast address ff00::/8, which forwards packets to Null0.

The code field C indicates that this is a directly connected network, similar to an IPv4 route of the same code. When an IPv6 interface is configured with a global unicast or unique local unicast address and is in the "up" state, the IPv6 prefix and the prefix length of that interface are added to the routing table as a connected route. In Example 14-3, the first entry in R1's IPv6 routing table is the directly connected network of the GigabitEthernet0/0 interface:

```
C   2001:DB8:CAFE:1::/64 [0/0]
     via GigabitEthernet0/0, directly connected
```

This entry includes the prefix and prefix length 2001:db8:cafe:1::/64. The [0/0] indicates the administrative distance and metric of the route. A directly connected network can only have an administrative distance of 0 and a metric of 0. The second line of the routing table entry displays the exit interface and indicates that this entry is directly connected.

Note The administrative distance is a measure of the *trustworthiness*, or *preference*, of the source of the routing information. If a router learns about a network from more than one routing source, it chooses the source with the lower administrative distance. Administrative distance has only local significance and is not advertised in routing updates.

There are two connected routes in R1's routing table, one for each interface that was configured with a GUA IPv6 address. By using the **connected** option with the **show ipv6 route** command, you can filter the routing table output to display only the directly connected networks, as shown in Example 14-5.

Example 14-5 *Displaying R1's Connected Routes*

```
R1# show ipv6 route connected
IPv6 Routing Table - default - 5 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity>
C   2001:DB8:CAFE:1::/64 [0/0]
     via GigabitEthernet0/0, directly connected
C   2001:DB8:CAFE:2::/64 [0/0]
     via GigabitEthernet0/1, directly connected
R1#
```

Note An interface configured with only a link-local address is not displayed in the routing table because link-local addresses are not routable. Link-local addresses are discussed shortly.

Code: Local

Besides the two connected routes in R1's routing table, notice that there are three other networks with the code L, indicating a local route. Your first inclination might be to think that these are link-local networks, but they are not. A local route is the IPv6 unicast (GUA or ULA) address of that interface. These are /128 routes and are essentially host routes for the router's GUA or ULA address, along with an additional local route for a multicast prefix. As in a connected network, a local route has an administrative distance of 0 and a metric of 0.

R1's GigabitEthernet0/0 interface has been manually configured with the GUA address 2001:db8:cafe:1::1/64. The local route entry in its routing table is:

```
L   2001:DB8:CAFE:1::1/128 [0/0]
     via GigabitEthernet0/0, receive
```

Local routes allow the router to more efficiently process packets directed to the router itself rather than packets that need to be routed. Just like any destination device, the router "receives" and processes packets with a destination IPv6 address that matches an address of one of its local entries.

For example, if another device is attempting to ping (ICMPv6 Echo Request) one of the router's interface addresses, the router receives the packet and determines that the packet's destination address matches one of its local routes. Instead of attempting to forward the packet, the router processes the packet as the destination device. In this example, after processing the packet, the router responds with an ICMPv6 Echo Reply.

Note Unlike connected networks, local routes are not propagated using the **redistributed connected** command within a dynamic routing process. Because a connected network is redistributed, it is unnecessary to advertise a potentially large number of host routes.

Because local host routes are in the routing table, they are also included in the Cisco Express Forwarding (CEF) table. CEF for IPv6 (CEFv6) is discussed later in this chapter.

R1 has two interfaces, both with routable GUA addresses. This accounts for two of the three local routes in its IPv6 routing table. The **show ipv6 route local** command in Example 14-6 filters the output of the IPv6 routing table, displaying only the three local routes.

Example 14-6 *Displaying R1's Local Routes*

```
R1# show ipv6 route local
IPv6 Routing Table - default - 5 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity>
L   2001:DB8:CAFE:1::1/128 [0/0]
     via GigabitEthernet0/0, receive
L   2001:DB8:CAFE:2::1/128 [0/0]
     via GigabitEthernet0/1, receive
L   FF00::/8 [0/0]
     via Null0, receive
R1#
```

The third local route in R1's IPv6 routing table is a Null0 route to the multicast address ff00::/8:

```
L   FF00::/8 [0/0]
     via Null0, receive
```

By default, a router does not forward multicast packets. To forward multicast packets, the following global unicast command must be configured.

```
Router(config)# ipv6 multicast-routing
```

This command does not remove the Null0 route but causes the router to discard any multicast packets that are not explicitly listed in the IPv6 routing table.

What about the link-local addresses on IPv6 interfaces? Link-local addresses are not included in the IPv6 routing table because they are not routable off the link.

Note In earlier IOS versions, the IPv6 routing table included an additional local entry, fe80::/10, with exit interface Null0. This confirmed that the router discarded any packets destined for a link-local address other than its own. Although current IOS versions no longer include this entry, the router still does not forward any packets with an IPv6 destination (or a source) address that is a link-local address.

Configuring IPv6 Static Routes

Configuring IPv6 static routes is much like configuring static routes for IPv4. This section assumes that you are familiar with configuring IPv4 static routes.

The following is the basic syntax for an IPv6 static route used in this chapter:

```
Router(config)# ipv6 route ipv6-prefix/prefix-length {ipv6-address | interface-type
interface-number} [next-hop-address]
```

This is the complete syntax for configuring an IPv6 static route:

```
Router(config)# ipv6 route [vrf vrf-name] ipv6-prefix/prefix-length
{ipv6-address | interface-type interface-number [ipv6-address]}
[nexthop-vrf [vrf-name1 | default]] [administrative-distance]
[administrative-multicast-distance | unicast | multicast]
[next-hop-address] [tag tag] [name name]
```

Note Most of these **show ipv6 route** options are beyond the scope of this book and are not discussed. For further information, see *Cisco IOS IPv6 Command Reference*, at www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book/ipv6-i4.html#wp3033854992.

The following descriptions are from the *Cisco IOS IPv6 Command Reference* for the basic **ipv6 route** syntax used in this chapter:

- *ipv6-prefix*: The IPv6 network that is the destination of the static route. Can also be a host name when static host routes are configured.
- */prefix-length*: The length of the IPv6 prefix. A decimal value that indicates how many of the high-order contiguous bits of the address comprise the prefix (the network portion of the address). A slash must precede the decimal value.
- *ipv6-address*: The IPv6 address of the next hop that can be used to reach the specified network. The IPv6 address of the next hop need not be directly connected; recursion is done to find the IPv6 address of the directly connected next hop. When an interface type and interface number are specified, you can optionally specify the IPv6 address of the next hop to which packets are output. Note that you must specify an interface type and an interface number when using a link-local address as the next hop. (The link-local next hop must also be an adjacent router.) This argument must be in the form documented in RFC 4291, where the address is specified in hexadecimal, using 16-bit values between colons.
- *interface-type*: The interface type. For more information about supported interface types, use the ? online help function. You can use the *interface-type* argument to direct static routes out point-to-point interfaces (such as serial or tunnel interfaces) and broadcast interfaces (such as Ethernet interfaces). When using the *interface-type* argument with point-to-point interfaces, there is no need to specify the IPv6 address of the next hop. When using the *interface-type* argument with broadcast interfaces, you should always specify the IPv6 address of the next hop or ensure that the specified prefix is assigned to the link. A link-local address should be specified as the next hop for broadcast interfaces. (Although the next-hop address can be either a global unicast address [or unique local address] or a link-local address, it is recommended that you use a link-local address.) The next-hop address is used only to determine the proper link layer (MAC) address of the next-hop router. This prevents any need for a recursive lookup of the next-hop address in the routing table. Link-local addresses are usually associated with the egress interface.

- *interface-number*: The interface number. For more information about the numbering syntax for supported interface types, use the ? online help function.
- *next-hop-address* (optional): The address of the next hop that can be used to reach the specified network.

The following sections show how to configure IPv6 static routes for the routers in Figure 14-3 to enable reachability from all the networks.

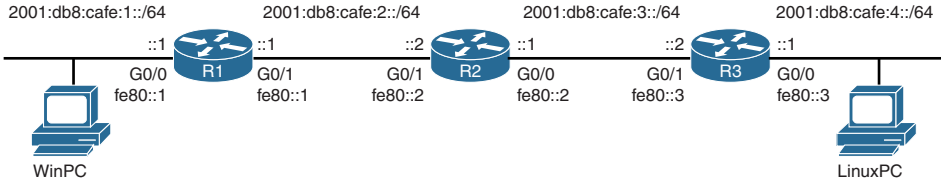


Figure 14-3 *Topology for an IPv6 Static Route*

Static Routes with a GUA Next-Hop Address

A common type of IPv6 static route is with a global unicast address (or unique local address) as the next-hop address. Example 14-7 shows the configuration of two static routes on R1 in order to reach the two remote networks in the topology. Both static routes use the same next-hop GUA address, R2's G0/1 address 2001:db8:cafe:2::2. The **show ipv6 route static** command in Example 14-7 verifies that both of these routes are in R1's IPv6 routing table.

Example 14-7 *Configuration and Verification of a Static Route with a GUA Next-Hop Address*

```
R1(config)# ipv6 route ?
X:X:X:X::X/<0-128>  IPv6 prefix
  static              Configure static route attributes
  vrf                 IPv6 Routing table

R1(config)# ipv6 route 2001:db8:cafe:3::/64 2001:db8:cafe:2::2
R1(config)# ipv6 route 2001:db8:cafe:4::/64 2001:db8:cafe:2::2
R1(config)# end
R1# show ipv6 route static
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity>
S   2001:DB8:CAFE:3::/64 [1/0]
     via 2001:DB8:CAFE:2::2
S   2001:DB8:CAFE:4::/64 [1/0]
     via 2001:DB8:CAFE:2::2
R1#
```

Note Routers R1, R2, and R3 have been previously enabled as IPv6 routers using the **ipv6 unicast-routing** global configuration command.

Static Routes with a Link-Local Next-Hop Address

This section shows how to configure the IPv6 static routes on router R2, this time using the link-local address of the adjacent router as the next-hop address. Since a next-hop address is typically an address on the same shared network segment, using a link-local address makes sense in IPv6. Remember that link-local addresses can only be reached from devices on the same link.

The syntax for configuring a static route with a link-local next-hop address requires the addition of an exit interface prior to the next-hop address:

```
Router(config)# ipv6 route ipv6-prefix/prefix-length interface-type interface-
number next-hop-address
```

Example 14-8 shows R2 configured with two IPv6 static routes, using the link-local addresses of R1 and R3 as the respective next-hop addresses for the routes. Notice the error when attempting to configure the static route using a link-local address without an exit interface. When using a link-local address as a next-hop address, the exit interface is required because the same link-local address can exist on any of the router's links. (A link-local address has to be unique only on the link.)

Example 14-8 *Configuration and Verification of a Static Route with a Link-Local Next-Hop Address*

```
R2(config)# ipv6 route 2001:db8:cafe:1::/64 fe80::1
R2(config)# % Interface has to be specified for a link-local nexthop
R2(config)# ipv6 route 2001:db8:cafe:1::/64 gigabitethernet0/1 fe80::1
R2(config)# ipv6 route 2001:db8:cafe:4::/64 gigabitethernet0/0 fe80::3
R2(config)# end
R2# show ipv6 route static
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity>
S   2001:DB8:CAFE:1::/64 [1/0]
     via FE80::1, GigabitEthernet0/1
S   2001:DB8:CAFE:4::/64 [1/0]
     via FE80::3, GigabitEthernet0/0
R2#
```

The static route to R1's 2001:db8:cafe:1::/64 network uses R2's G0/1 exit interface and R1's link-local address, fe80::1. The static route to R3's 2001:db8:cafe:4::/64 network uses

R2's G0/0 exit interface and R3's link-local address, fe80::3. The **show ipv6 route static** command in the same example verifies the two entries in R2's routing table.

Note A static route configured with an exit interface and a next-hop address is known as a *fully qualified static route*. A fully qualified static route can use a global unicast address, a unique local address, or a link-local address as the next-hop address.

Static Routes with Only an Exit Interface

Static routes on point-to-point serial links are sometimes configured using only an exit interface. This method was commonly used to avoid having the router perform a recursive lookup for the next-hop IP address. A recursive lookup requires the router to perform two routing table lookups. The first lookup is for the prefix that matches the destination IP address. The second lookup is for the next-hop address in the routing table entry. This type of configuration was common before CEF became the default behavior on most IOS routing platforms.

Note Broadcast networks such as Ethernet require IPv6 static routes to include a next-hop IPv6 address.

CEF (Cisco Express Forwarding) has been the default behavior since IOS 12.0. CEF provides optimized lookup for efficient packet forwarding by using two main data structures stored in the data plane: a FIB (Forwarding Information Base), which is a copy of the routing table, and an adjacency table that includes Layer 2 addressing information. The information combined in both of these tables work together so there is no recursive lookup needed for next-hop IP address lookups. In other words, a static route using a next-hop IP requires only a single lookup when CEF is enabled on the router. Although static routes that use only an exit interface on point-to-point networks are (or were) common, the use of the default CEF forwarding mechanism makes this practice unnecessary. CEF is discussed later in this chapter.

Note It is typically recommended that you always include a next-hop address with a static route. In some circumstances, a static route with only an exit interface and no next-hop address may lead to an unresolved route, and this is therefore discouraged unless absolutely necessary.

Example 14-9 shows RouterA configured with a static route using only a serial exit interface and no next-hop address. Notice that although the output in the routing table shows this entry as directly connected, the administrative distance for this route is 1. The administrative distance of 1 is also verified using the command **show ipv6 route 2001:db8:feed:3::/64**. Only a directly connected network with a code of C has an administrative distance of 0.

Example 14-9 *Configuration and Verification of a Static Route with an Exit Interface on a Serial Interface*

```
RouterA(config)# ipv6 route 2001:db8:feed:3::/64 serial0/0/0
RouterA(config)# end
RouterA#
RouterA# ping 2001:db8:feed:3::1
<output omitted for brevity>
!!!!!
RouterA# show ipv6 route static
IPv6 Routing Table - default - 4 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity>
S   2001:DB8:FEED:3::/64 [1/0]
     via Serial0/0/0, directly connected
RouterA# show ipv6 route 2001:db8:feed:3::/64
Routing entry for 2001:DB8:FEED:3::/64
  Known via "static", distance 1, metric 0
  Route count is 1/1, share count 0
  Routing paths:
    directly connected via Serial0/0/0
      Last updated 00:01:30 ago

RouterA#
```

Note There is some common misinformation regarding the administrative distance of 0. Only a directly connected network in IPv4 or a directly connected network or local interface in IPv6 can have an administrative distance of 0. A static route configured with an exit interface does not have an administrative distance of 0. Static routes and dynamically learned routes can never have an administrative distance of 0. Even though an IPv4 or IPv6 static route configured with an exit interface displays the phrase "directly connected" in the routing table, the administrative distance of that static route is still 1 by default.

Default Static Routes with Link-Local Next-Hop Addresses

Configuring an IPv6 default static route is similar to configuring a default static route for IPv4, with an all-0s prefix and a /0 prefix length. An IPv4 default route uses 0.0.0.0 0.0.0.0 to represent the all-0s prefix with a /0 prefix length, whereas IPv6 uses ::/0.

The following is the syntax for an IPv6 default static route:

```
Router(config)# ipv6 route ::/0 {ipv6-address | interface-type interface-number}
[next-hop-address]
```

Example 14-10 shows the configuration of a default static route on R3. The default route is configured using the exit interface G0/1 and R2's link-local address, fe80::2, as the next-hop address. The static route is verified in R3's IPv6 routing table using the **show ipv6 route static** command.

Example 14-10 *Configuration and Verification of a Default Static Route*

```
R3(config)# ipv6 route ::/0 gigabitethernet0/1 fe80::2
R3(config)# end
R3# show ipv6 route static
IPv6 Routing Table - default - 6 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity?>
S   ::/0 [1/0]
     via FE80::2, GigabitEthernet0/1
R3#
```

Verifying IPv6 Static Routes

Routers R1, R2, and R3 have all been configured with static routes that should provide complete reachability to all the networks for the topology in Figure 14-1. Previous examples have verified the static route entries in each of the router's routing tables with the **show ipv6 route static** command.

The **show ipv6 route summary** command displays the IPv6 routing table in a summarized format. Example 14-11 shows the output from the **show ipv6 route** and **show ipv6 route summary** commands for R1. Both of these commands are included so you can see how the information is summarized with **show ipv6 route summary**.

Example 14-11 **show ipv6 route** *and* **show ipv6 route summary** *Commands*

```
R1# show ipv6 route
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity>
C   2001:DB8:CAFE:1::/64 [0/0]
     via GigabitEthernet0/0, directly connected
L   2001:DB8:CAFE:1::1/128 [0/0]
     via GigabitEthernet0/0, receive
C   2001:DB8:CAFE:2::/64 [0/0]
     via GigabitEthernet0/1, directly connected
L   2001:DB8:CAFE:2::1/128 [0/0]
     via GigabitEthernet0/1, receive
```

```
S   2001:DB8:CAFE:3::/64 [1/0]
     via 2001:DB8:CAFE:2::2
S   2001:DB8:CAFE:4::/64 [1/0]
     via 2001:DB8:CAFE:2::2
L   FF00::/8 [0/0]
     via Null0, receive
R1# show ipv6 route summary
IPv6 routing table name is default(0) global scope - 7 entries
IPv6 routing table default maximum-paths is 16
Route Source    Networks    Overhead    Memory (bytes)
connected       2           224         264
local           3           336         396
application     0           0           0
ND              0           0           0
  Default: 0  Prefix: 0  Destination: 0  Redirect: 0
static          2           224         264
  Static: 2  Per-user static: 0
Total           7           784         924
  Number of prefixes:
    /8: 1, /64: 4, /128: 2
R1#
```

Example 14-12 shows the output for the **show running-config** command for each of the three routers. The output provides an overview of the previous static route configurations. The **| include ipv6 route** option displays only the lines in the running-config that contain the phrase "ipv6 route."

Example 14-12 *Verification Using* **show running-config**

```
R1# show running-config | include ipv6 route
ipv6 route 2001:DB8:CAFE:3::/64 2001:DB8:CAFE:2::2
ipv6 route 2001:DB8:CAFE:4::/64 2001:DB8:CAFE:2::2
R1#
------------------------------------------------------------
R2# show running-config | include ipv6 route
ipv6 route 2001:DB8:CAFE:1::/64 GigabitEthernet0/1 FE80::1
ipv6 route 2001:DB8:CAFE:4::/64 GigabitEthernet0/0 FE80::3
R2#
------------------------------------------------------------
R3# show running-config | include ipv6 route
ipv6 route ::/0 GigabitEthernet0/1 FE80::2
R3#
```

The **show ipv6 static** and **show ipv6 static detail** commands in Example 14-13 show information similar to that of the previous running-config example but with additional details. Both commands include the administrative distance and any information that matches one or more of the codes. The **detail** option includes the number of paths and the exit interface of each route.

Example 14-13 **show ipv6 static** *and* **show ipv6 static detail** *Commands*

```
R1# show ipv6 static
IPv6 Static routes Table - default
Codes: * - installed in RIB, u/m - Unicast/Multicast only
       U - Per-user Static route
       N - ND Static route
       M - MIP Static route
       P - DHCP-PD Static route
       R - RHI Static route
*   2001:DB8:CAFE:3::/64 via 2001:DB8:CAFE:2::2, distance 1
*   2001:DB8:CAFE:4::/64 via 2001:DB8:CAFE:2::2, distance 1
R1# show ipv6 static detail
IPv6 Static routes Table - default
Codes: * - installed in RIB, u/m - Unicast/Multicast only
       U - Per-user Static route
       N - ND Static route
       M - MIP Static route
       P - DHCP-PD Static route
       R - RHI Static route
*   2001:DB8:CAFE:3::/64 via 2001:DB8:CAFE:2::2, distance 1
     Resolves to 1 paths (max depth 1)
     via GigabitEthernet0/1
*   2001:DB8:CAFE:4::/64 via 2001:DB8:CAFE:2::2, distance 1
     Resolves to 1 paths (max depth 1)
     via GigabitEthernet0/1
R1#
```

The **ping** command is a useful tool to help verify reachability. In Example 14-14, two **ping** commands are used to verify reachability from both of R1's interfaces. The first command, **ping 2001:db8:cafe:4::1**, sends the pings from R1's G0/1 interface. This is the exit interface in R1's routing table for this destination network. The second **ping** command, **ping 2001:db8:cafe:4::1 source gigabitethernet0/0**, stipulates that the pings use the address of the G0/0 interface (2001:db8:cafe:1::1) as the source IPv6 address of the packets. A successful ping using the source address of G0/0 ensures that the destination (R3) can reach R1's 2001:db8:cafe:1::/64 network.

Example 14-14 *Verifying Reachability Using* **ping**

```
R1# ping 2001:db8:cafe:4::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:4::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/12 ms
R1# ping 2001:db8:cafe:4::1 source gigabitethernet0/0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:4::1, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:CAFE:1::1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
R1#
```

Note Instead of specifying the exit interface with **source gigabitethernet 0/0**, the actual source IPv6 address, **source 2001:db8:cafe:1::1**, can be used.

The last verification command to be used is the **traceroute** command. Example 14-15 shows a successful traceroute from R1 to R3's G0/0 interface address 2001:db8:cafe:4::1.

Example 14-15 *Verifying Reachability Using the* **traceroute** *Command*

```
R1# traceroute 2001:db8:cafe:4::1
Type escape sequence to abort.
Tracing the route to 2001:DB8:CAFE:4::1

  1 2001:DB8:CAFE:2::2 0 msec 0 msec 0 msec
  2 2001:DB8:CAFE:3::2 0 msec 0 msec 0 msec
R1#
```

Summarizing IPv6 Routes

Keeping routing tables small makes them easier to read and easier to analyze. But it also provides for faster and more efficient routing table lookups. The method for summarizing IPv6 routes is similar to the method for summarizing routes for IPv4.

In this section, the following five networks are summarized into a single network:

- 2001:db8:feed:1::/64
- 2001:db8:feed:2::/64
- 2001:db8:feed:3::/64

- 2001:db8:feed:4::/64
- 2001:db8:feed:5::/64

The following steps are used to summarize these five IPv6 networks into a single IPv6 prefix and prefix length:

Step 1. List the IPv6 networks and locate the first hextet where you see a difference in at least one of the networks. Convert the different hextet to binary. Figure 14-4 shows that all five networks have the first three hextets in common, 2001:db8:feed. The fourth hextet differs and is converted to binary.

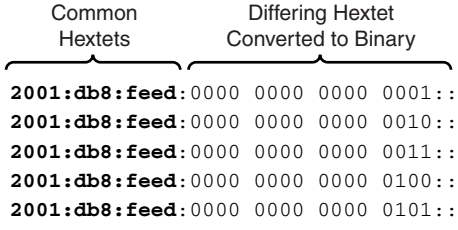


Figure 14-4 *Step 1: Converting the Different Hextet to Binary*

Step 2. Starting from the leftmost bit, locate the last matching bit for all the addresses. Count the number of matching bits to determine the prefix length for the summary route. Figure 14-5 shows that there are 61 matching bits, or a /61 prefix length, for the summary route.

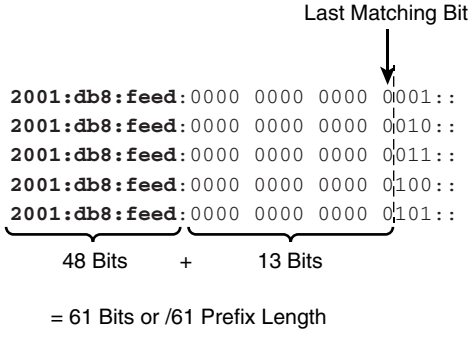


Figure 14-5 *Step 2: Counting the Matching Bits for the Prefix Length*

Step 3. To determine the summarized network address (prefix), copy the matching bits and append the rest of the address with zero bits. Convert the binary section back to hexadecimal. Figure 14-6 shows the matching bits with all zero bits added to the rest of the address. Converting the binary portion to hexadecimal results in the address 2001:db8:feed::. Appending the /61 prefix length from step 2 yields the summary address 2001:db8:feed::/61.

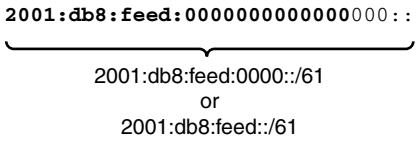


Figure 14-6 *Step 3: Determining the Prefix*

Note The summary prefix 2001:db8:feed::/61 not only includes the prefixes 2001:db8:feed:1::/64 through 2001:db8:feed:5::/64 but also includes the prefixes 2001:db8:feed::/64, 2001:db8:feed::6:/64, and 2001:db8:feed:7::/64.

IPv6 Summary Static Route

Figure 14-7 shows R1 connected to the five individual IPv6 networks summarized in the previous steps.

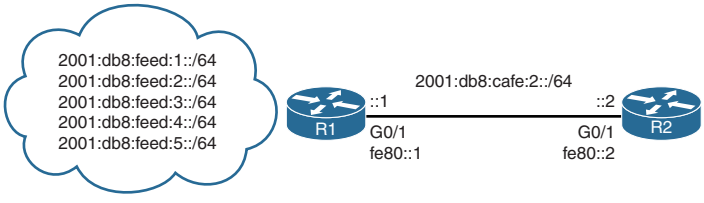


Figure 14-7 *R1 Networks*

Example 14-16 shows the configuration of a single IPv6 summary static route, using the prefix and prefix length 2001:db8:feed::/61. The **ping** command in Example 14-16 verifies reachability to one of the five prefixes.

Example 14-16 *Summary Static Route Configuration and Verification*

```
R2(config)# ipv6 route 2001:db8:feed::/61 gigabitethernet0/1 fe80::1
R2(config)# end
R2# ping 2001:db8:feed:4::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:FEED:4::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/8 ms
R2#
```

CEF for IPv6

CEF is a Layer 3 switching technology designed for Cisco routers and multilayer switches for IPv4 and is also available for IPv6 as CEFv6. CEF uses a switching method that optimizes route lookups to achieve very fast packet forwarding. CEF uses two tables to accomplish this:

- **Forwarding Information Base (FIB):** The FIB is similar to an IP routing table. It is an ordered list, with the most specific route first, for each IPv6 prefix/prefix length in the routing table or, in the case of IPv4, each IPv4 network/subnet. The FIB is a mirror image of the forwarding information contained in the IP routing table. When routing or topology changes occur in the network, the IP routing table is updated, and those changes are reflected in the FIB. The FIB maintains next-hop address information based on the information in the IP routing table.
- **Adjacency table:** The FIB contains Layer 3 network and next-hop information. To streamline packet forwarding, the FIB's corresponding Layer 2 information for every next-hop entry is maintained in a table called the adjacency table. The adjacency table is built from the IPv6 Neighbor Cache (or ARP Cache for IPv4) and contains the Layer 3–to–Layer 2 address mappings. The adjacency table contains the MAC addresses of next-hop nodes such as routers and hosts. Whenever the router updates its IPv6 Neighbor Cache (or ARP Cache for IPv4), the adjacency table is also updated.

Note CEF is beyond the scope of this book. For more information, see the book *Cisco Express Forwarding* by Nakia Stringfield, published by Cisco Press.

CEFv6 is available on Cisco IOS Release 12.2(13)T and later and 12.2(9)S and later. CEF for IPv4 is enabled by default on most platforms, but CEFv6 is disabled by default. Starting with IOS 15.0, CEFv6 is automatically enabled when IPv6 routing is enabled using the **ipv6 unicast-routing** command. If this command doesn't automatically enable CEFv6, use **ipv6 cef** in addition to the **ipv6 unicast-routing** command.

Note CEF for IPv4 must be enabled prior to enabling CEF for IPv6. CEF for IPv4 is enabled with the **ip cef** global configuration command.

Example 14-17 shows a sequence of commands to enable and verify CEFv6 on router R3. The initial **show running-config** command shows that CEF for IPv4 (**ip cef**) is enabled, but CEFv6 (**no ipv6 cef**) is disabled. The **show ipv6 cef** command verifies that CEFv6 is not running. After enabling R3 as an IPv6 router with the **ipv6 unicast-routing** command, the running-config shows that CEFv6 is now enabled (**ipv6 cef**). Using the **show ipv6 cef** command once again verifies that CEFv6 is now running on R3.

Example 14-17 *Summary Static Route Configuration and Verification*

```
R3# show running-config
<output omitted for brevity>
!
no ip domain lookup
ip cef
no ipv6 cef

R3# show ipv6 cef
%IPv6 CEF not running

R3# conf t
R3(config)# ipv6 unicast-routing
R3(config)# exit
R3# show running-config
!
no ip domain lookup
ip cef
ipv6 unicast-routing
ipv6 cef
multilink bundle-name authenticated
!

R3# show ipv6 cef
::/0
  nexthop FE80::2 GigabitEthernet0/1
::/127
  discard
2001:DB8:CAFE:3::/64
  attached to GigabitEthernet0/1
2001:DB8:CAFE:3::2/128
  receive for GigabitEthernet0/1
2001:DB8:CAFE:4::/64
  attached to GigabitEthernet0/0
2001:DB8:CAFE:4::1/128
  receive for GigabitEthernet0/0
FE80::/10
  receive for Null0
FF00::/8
  multicast
R3#
```

Summary

This chapter examines the following topics associated with routing IPv6:

- Configuring a router as an IPv6 router
- Understanding the IPv6 routing table
- Configuring and verifying IPv6 static routes
- CEF for IPv6

To be an IPv6 router, a Cisco router must be configured with the **ipv6-unicast routing** command. This command does the following:

- It enables the router as a member of the all-IPv6 routers multicast group ff02::2.
- It enables the router to be configured with an IPv6 dynamic routing protocol.
- It enables the router to forward IPv6 packets transiting the router.
- It enables the router to send ICMPv6 Router Advertisements on Ethernet (and FDDI) interfaces.

Cisco IOS maintains a separate routing table for IPv6 and IPv4. The contents of the IPv6 routing table are displayed using the **show ipv6 route** command. Many of the routing table codes are similar to the same codes in an IPv4 routing table, including the following:

- C (connected) indicates a directly connected network (prefix).
- L (local route) is the IPv6 unicast (GUA or ULA) address of an interface, along with an additional multicast prefix.

Two routing table codes are specific to IPv6:

- NDp (Neighbor Discovery prefix) indicates that the network prefix was learned using the Neighbor Discovery Protocol.
- ND (Neighbor Discovery) indicates that the default route was learned via Neighbor Discovery.

Configuring IPv6 static routes is similar to configuring static routes for IPv4. It involves the following basic syntax:

```
Router(config)# ipv6 route ipv6-prefix/prefix-length {ipv6-address | interface-type
interface-number} [next-hop-address]
```

An IPv6 static route can be configured using a next-hop address of either a global unicast address or unique local address. A static route can also be configured using a link-local next-hop address, but it must be a fully qualified route and include the exit interface. IPv6 static default routes are configured using ::/0 as the *ipv6-prefix/prefix-length*.

Route summarization keeps routing tables small and makes for more efficient routing table lookups. Summarizing multiple IPv6 networks to a single prefix and prefix length is much the same in IPv6 as it is in IPv4, and it involves the following steps:

Step 1: List the IPv6 networks and locate the first hextet where you see a difference in at least one of the networks. Convert the different hextet to binary.

Step 2: Starting from the leftmost bit, locate the last matching bit for all the addresses. Count the number of matching bits to determine the prefix length for the summary route.

Step 3: To determine the summarized network address (prefix), copy the matching bits and append the rest of the address with zero bits. Convert the binary section back to hex.

Cisco Express Forwarding (CEF) is a Layer 3 switching technology designed for Cisco routers and multilayer switches for IPv4 and is also available for IPv6 (CEFv6). CEF uses a switching method that optimizes route lookups to achieve very fast packet forwarding.

CEFv6 is automatically enabled when the **ipv6 unicast-routing** command is used. If this command doesn't automatically enable CEFv6, use **ipv6 cef** in addition to the **ipv6 unicast-routing** command. CEF for IPv4 (which is enabled by default) must be running before CEFv6 can be enabled.

Review Questions

1. Which of the following is enabled when a router is configured with the **ipv6 unicast-routing** command? (Choose all that apply.)
   a. IPv6 addresses can be configured on interfaces.
   b. IPv6 static routes can be configured.
   c. Dynamic routing protocols can be configured.
   d. Router Advertisement messages are sent out Ethernet interfaces.
   e. The router forwards packets transiting the router.
2. Which IPv6 routing table code indicates that the prefix was learned via Neighbor Discovery?
3. Which IPv6 routing table code displays the directly connected prefix of a router's interface?
4. Does the IPv6 routing table display link-local unicast addresses? Why or why not?
5. What does the following IPv6 routing table entry indicate?

```
L   FF00::/8 [0/0]
      via Null0, receive
```

6. Which of the following is required when configuring an IPv6 static route using a link-local address as the next-hop address?
   a. The link-local address of the exit interface.
   b. The interface type/number of the next-hop router.
   c. The interface type/number of the exit interface.
   d. The **link-local** keyword after the next-hop address.
7. What is the administrative distance of the following routing table entry?

```
S   2001:DB8:CAFE:2::/64 [1/0]
      via GigabitEthernet0/1, directly connected
```

   a. 0
   b. 1
   c. 64
   d. 255
8. What are the prefix and prefix length for an IPv6 default static route?
   a. ::/0
   b. 0.0.0.0/0
   c. 0/0
   d. 0::0/0
9. A router has four interfaces, each configured with an IPv6 GUA address. All four interfaces are in the "up" state. There are two IPv6 static routes configured on the router, both using the same exit interface and next-hop address. How many entries are in the IPv6 routing table?
10. Summarize the following four addresses into a single prefix and prefix length:

    2001:db8:face:11a0::/64

    2001:db8:face:11b0::/64

    2001:db8:face:11c0::/64

    2001:db8:face:11d0::/64
11. Which command(s) is used to automatically enable CEFv6 on most platforms?
    a. **ipv6-unicast routing**
    b. **ipv6-unicast routing** and **ipv6 cef**
    c. **ipv6 cef**
    d. No command is needed.

References

RFCs

RFC 2080, *RIPng for IPv6*, G. Malkin, Xylogics, www.ietf.org/rfc/rfc2080.txt, January 1997.

RFC 4291, *IP Version 6 Addressing Architecture*, R. Hinden, Nokia, www.ietf.org/rfc/rfc4291.txt, February 2006.

RFC 4760, *Multiprotocol Extensions for BGP-4*, T. Bates, Cisco Systems, www.ietf.org/rfc/rfc4760.txt, January 2007.

RFC 5308, *Routing IPv6 with IS-IS*, C. Hoppa, Cisco Systems, www.ietf.org/rfc/rfc5308.txt, October 2008.

RFC 7868, *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP) for IPv6*, D. Savage, Cisco Systems, www.ietf.org/rfc/rfc7868.txt, May 2016.

Websites

Cisco IOS IPv6 Command Reference, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book.html.

Implementing Static Routes for IPv6, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-stat-routes.pdf.

Books

IP Routing on Cisco IOS, IOS XE, and IOS XR: An Essential Guide to Understanding and Implementing IP Routing Protocols, by Brad Edgework, Aaron Foss, and Ramiro Rios, Cisco Press.

Cisco Express Forwarding, by Nakia Stringfield, Cisco Press.

This page intentionally left blank

Chapter 15

EIGRP for IPv6

Enhanced Interior Gateway Routing Protocol (EIGRP) is a distance-vector, classless routing protocol that was released as a proprietary protocol in 1992 with IOS 9.21. As its name suggests, EIGRP is an enhancement of Cisco's Interior Gateway Routing Protocol (IGRP). Cisco's main purpose in developing EIGRP was to create a classless version of IGRP. EIGRP also includes several features that are not commonly found in other distance-vector routing protocols, such as IGRP and Routing Information Protocol (RIP).

Cisco released the basic functionality of EIGRP as an open standard to the IETF in RFC 7868, *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)*. Other networking vendors can implement EIGRP in order to interoperate with both Cisco and non-Cisco routers running EIGRP. However, advanced features of EIGRP, such as EIGRP stub, needed for Dynamic Multipoint Virtual Private Network (DMVPN) deployment, have not been released to the IETF. As this is an informational RFC, Cisco will continue to maintain control of EIGRP.

Note The term *hybrid routing protocol* is sometimes used when referring to EIGRP. However, this term is misleading because EIGRP is not a hybrid between distance-vector and link-state routing protocols; it is solely a distance-vector routing protocol. Therefore, Cisco is no longer using this term to refer to EIGRP. However, EIGRP is sometimes correctly referred to as an advanced distance-vector routing protocol.

Unique to EIGRP is Reliable Transport Protocol (RTP), which provides reliable and unreliable delivery of EIGRP packets. In addition, EIGRP establishes relationships with directly connected EIGRP neighbors. Neighbor relationships are used to track the status of these neighbors. RTP and the tracking of neighbor adjacencies set the stage for the EIGRP workhorse, Diffusing Update Algorithm (DUAL).

As the computational engine that drives EIGRP, DUAL resides at the center of the routing protocol. DUAL guarantees loop-free paths and backup paths, and it provides fast convergence. DUAL selects a primary route to install in the routing table, known as the successor, and also maintains a list of guaranteed loop-free backup routes, known as feasible successors. Both successors and feasible successors are maintained in EIGRP's topology table.

EIGRP for IPv6 is the EIGRP IPv4 equivalent for routing IPv6 prefixes. EIGRP IPv4 runs over an IPv4 network layer, communicating with other EIGRP IPv4 neighbors, and advertises only IPv4 routes. EIGRP for IPv6 has the same functionality as EIGRP for IPv4 but uses IPv6 as the network layer transport, communicating with EIGRP for IPv6 neighbors and advertising IPv6 routes.

The operations and functionality of EIGRP for IPv6 are similar to those of EIGRP for IPv4. You will see how these protocols compare in the next section.

Note This chapter, which focuses on the configuration of EIGRP for IPv6, assumes that you have some knowledge of EIGRP. However, this chapter reviews many EIGRP concepts, so even if you are new to EIGRP, you will find this chapter helpful. If you are new to EIGRP or wish to learn more, see the recommendations provided at the end of the chapter.

This chapter discusses two methods to configure EIGRP for IPv6:

- Classic EIGRP for IPv6
- EIGRP named mode

Note Cisco refers to the IPv4 version of EIGRP as just EIGRP, or sometimes EIGRP for IPv4. The IPv6 version of EIGRP is EIGRP for IPv6. For brevity, this chapter sometimes refers to these protocols as EIGRPv4 and EIGRPv6.

Comparing EIGRPv4 and EIGRPv6

EIGRP for IPv6 is available in Cisco IOS Release 12.4(6)T and later. EIGRP for IPv4 and EIGRP for IPv6 are two separate routing protocols. However, the configuration of EIGRPv4 and EIGRPv6 is similar. Many of the same processes and operational functionality are the same in both protocols. EIGRP has simply been redesigned and extended to support IPv6.

Table 15-1 provides an overview of EIGRP and also a comparison between EIGRPv4 and EIGRPv6.

Table 15-1 *Comparing EIGRPv4 and EIGRPv6*

	EIGRP for IPv4	EIGRP for IPv6
Advertised routes	IPv4 prefixes	IPv6 prefixes
Distance vector?	Yes	Yes
Computational algorithm	DUAL	DUAL
Default metric:	Bandwidth and delay	Bandwidth and delay
Optional metric:	Reliability and load	Reliability and load
Classic EIGRP metric calculation	32-bit composite metric	32-bit composite metric
EIGRP named mode metric calculation	64-bit wide metric	64-bit wide metric
Transport protocol	RTP	RTP
Update messages	Partial and bounded updates	Partial and bounded updates
Neighbor discovery	Hello packets	Hello packets
Message source address:	IPv4 address	IPv6 link-local address
Message destination address:	224.0.0.10 (multicast)	ff02::a (multicast)
Authentication	Plain text and MD5	MD5 and SHA with named EIGRP
EIGRP Router ID	32-bit Router ID	32-bit Router ID
Automatic summarization	At classful boundaries	N/A

As you can see from Table 15-1, the functionality is the same for both protocols, but there are some differences. EIGRPv4 sends messages to the IPv4 multicast address 224.0.0.10, the EIGRP router's multicast group address. These messages use the source IPv4 address of the exit interface. EIGRPv6 sends messages to the IPv6 multicast address ff02::a, the all-EIGRP-routers multicast group with link-local scope. EIGRP for IPv6 messages are sourced using the link-local address of the exit interface.

Both EIGRPv4 and EIGRPv6 use a 32-bit value for the Router ID. The 32-bit Router ID is represented in dotted-decimal notation, and an interface's IPv4 address can be used. However, the Router ID is a 32-bit value and not an IPv4 address.

The process for determining the Router ID is the same for both EIGRPv4 and EIGRPv6. It is best practice to use the EIGRP **router-id** command to configure the Router ID. If this command isn't used, EIGRP uses the highest IPv4 address of a loopback interface. If there are no loopback interfaces configured, EIGRP uses the highest IPv4 address of any of its active interfaces. The process of determining the 32-bit Router ID is the same for EIGRPv6. If the router is *not* dual-stacked, EIGRPv6 requires the use of the EIGRP **router-id** command to configure a 32-bit Router ID.

Note *Dual-stacked* means that a router has interfaces configured for both IPv4 and IPv6.

The traditional way to configure EIGRP involves configuring various parameters under the interface and EIGRP configuration mode, also known as classic EIGRP. In order to configure classic EIGRP IPv4 and IPv6, you need to configure separate EIGRP instances. With EIGRP named mode, everything is configured at a single place under the EIGRP configuration for both IPv4 and IPv6.

Classic EIGRP on IOS uses a 32-bit metric derived from a composite metric formula using five K values. To accommodate interfaces with bandwidths above 1 gigabit and up to 4.2 terabits and to allow EIGRP to perform path selections, the composite metric formula is modified in EIGRP named mode on IOS for IPv4 and IPv6. The modified metric is known as the EIGRP wide metric. It is a 64-bit metric and includes a sixth K value for future use. The delay is modified from being measured in tens of microseconds to being measured in picoseconds. The end result is that classic EIGRP and EIGRP named mode show different metrics for the same route.

Note The EIGRP wide metric is beyond the scope of this book. For more information see the *Cisco IP Routing: EIGRP Configuration Guide*, at www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_eigrp/configuration/xe-3s/ire-xe-3s-book.html.

EIGRP maintains separate EIGRP neighbor tables and EIGRP topology tables for IPv4 and IPv6. This concept is illustrated in Figure 15-1 with two dual-stacked routers running EIGRP for both protocols. The routers also maintain separate IPv4 and IPv6 routing tables.

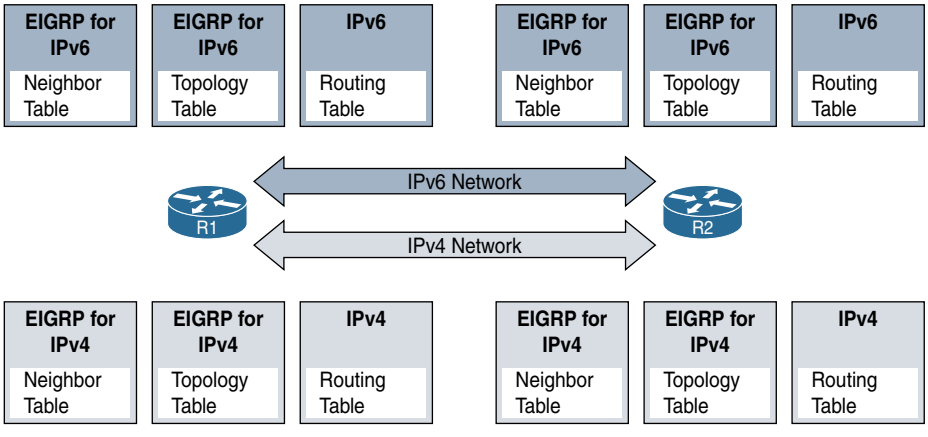


Figure 15-1 *Separate EIGRP Tables and Routing Tables for IPv4 and IPv6*

Classic EIGRP for IPv6

Configuration of classic (or traditional) EIGRP for IPv6 has its roots in the original method of configuring EIGRP for IPv4. Figure 15-2 shows the topology used in this

section to configure classic EIGRP for IPv6. Notice that each router has been configured with the same link-local address on each of its interfaces. This has been done to make it easier to recognize the source address of the EIGRP messages. You will see in Chapter 17, "Deploying IPv6 in the Network," that this is not typically best practice on LAN interfaces with end devices.

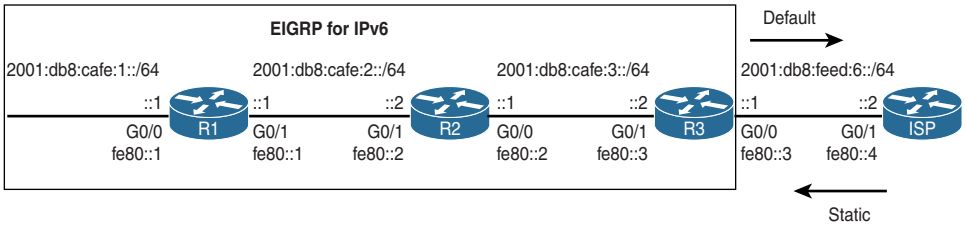


Figure 15-2 *Topology for Configuring Classic EIGRP for IPv6*

Note You could configure router R2's interfaces with only link-local addresses, no global unicast addresses. This is because R2 has no end user interfaces. RFC 7404, Using Only Link-Local Addressing inside an IPv6 Network, discusses implementing routing protocols using only link-local addresses on infrastructure links.

Configuring Classic EIGRP for IPv6

This section shows how to configure routers R1, R2, and R3 to share routing information within the EIGRPv6 routing domain 2001:db8:cafe::/48. You will configure R3 with a default route via the ISP router so that it propagates the default route to the other routers in the EIGRP domain.

The ISP router is configured with a static route to reach 2001:db8:cafe::/48, as shown in Example 15-1.

Example 15-1 *Configuring Static Routes on ISP*

```
ISP(config)# ipv6 route 2001:db8:cafe::/48 g0/1 fe80::3
```

Example 15-2 shows the configuration of classic EIGRPv6 on router R1. After configuring the first **ipv6 router eigrp 1** command, notice the error: **% IPv6 routing not enabled**. Before enabling any dynamic IPv6 routing protocol, including EIGRPv6, you must first enable IPv6 routing with the **ipv6 unicast-routing** command.

Example 15-2 *Configuring EIGRP for IPv6 on R1*

```
R1(config)# ipv6 router eigrp 1
% IPv6 routing not enabled
R1(config)# ipv6 unicast-routing
R1(config)# ipv6 router eigrp 1
R1(config-rtr)# eigrp router-id 1.1.1.1
```

```
R1(config-rtr)# passive-interface gigabitethernet0/0
R1(config-rtr)# exit
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 eigrp 1
R1(config-if)# exit
R1(config)# interface gigabitethernet 0/1
R1(config-if)# ipv6 eigrp 1
R1(config-if)#
```

The following is a brief description of the commands used in Example 15-2:

- R1(config)**# ipv6 router eigrp 1**

  This command creates an EIGRPv6 routing process using the autonomous system number 1. Just as with EIGRPv4, the autonomous system must be the same on all routers in the EIGRPv6 routing domain. The autonomous system number is only significant within the EIGRP routing domain and is not associated with any officially registered autonomous system number typically used by Border Gateway Protocol (BGP).

- R1(config-rtr)**# eigrp router-id 1.1.1.1**

  The **eigrp router-id** command configures the EIGRP Router ID. This command is required if there are no loopback or active physical interfaces with an IPv4 address.

- R1(config-rtr)**# passive-interface gigabitethernet0/0**

  R1's G0/0 interface is configured as a passive interface because it doesn't have any EIGRP neighbors on that interface. This command suppresses EIGRP hello messages and routing updates from being sent out an interface.

- R1(config-if)**# ipv6 eigrp 1**

  EIGRPv6 is enabled directly on the interface using the **ipv6 eigrp** *as-number* interface command. The autonomous system number must match the autonomous system number used to enable the EIGRP routing process with the **ipv6 router eigrp** command. When an interface is enabled for EIGRP, it attempts to form neighbor adjacencies with adjacent routers, and it includes the interface's prefix in its routing updates.

Note Older versions of IOS required the use of the **no shutdown** command in EIGRP router configuration mode. This command is no longer required.

Example 15-3 shows the classic EIGRP for IPv6 configuration on router R2. The same autonomous system number 1 is required for R2 to form an EIGRP adjacency with R1. Notice that immediately after EIGRPv6 is enabled on R2's G0/1 interface, it forms an EIGRP adjacency with its neighbor R1. EIGRP messages are sourced from the router's

link-local address on the shared link. As you can see in the "new adjacency" message, R2's G0/1 interface forms an adjacency with a neighbor R1 at fe80::1.

Example 15-3 *Configuring EIGRP for IPv6 on R2*

```
R2(config)# ipv6 unicast-routing
R2(config)# ipv6 router eigrp 1
R2(config-rtr)# eigrp router-id 2.2.2.2
R2(config-rtr)# exit
R2(config)# interface gigabitethernet 0/0
R2(config-if)# ipv6 eigrp 1
R2(config-if)# exit
R2(config)# interface gigabitethernet 0/1
R2(config-if)# ipv6 eigrp 1
R2(config-if)#
*Feb 12 21:14:24.242: %DUAL-5-NBRCHANGE: EIGRP-IPv6 1: Neighbor FE80::1
  (GigabitEthernet0/1) is up: new adjacency
R2(config-if)#
```

Example 15-4 shows the classic EIGRPv6 configuration on router R3, with commands similar to those used on R1 and R2. Only R3's G0/1 interface is within the EIGRPv6 routing domain and needs to be enabled for EIGRPv6. R3 is configured with an IPv6 static default route **ipv6 route ::/0 gigabitethernet0/0 fe80::4**. The static route uses R3's G0/0 exit interface and a next-hop address of the ISP's link-local address, fe80::4. R3's G0/1 interface is configured with the **ipv6 summary-address eigrp 1 ::/0** command. This summary route (::/0) is the equivalent of a default route being injected into the EIGRP routing domain.

Note EIGRP includes other methods for propagating default routing into the EIGRP domain.

Example 15-4 *Configuring EIGRP for IPv6 on R3*

```
R3(config)# ipv6 unicast-routing
R3(config)# ipv6 route ::/0 gigabitethernet0/0 fe80::4
R3(config)# ipv6 router eigrp 1
R3(config-rtr)# eigrp router-id 3.3.3.3
R3(config-rtr)# exit
R3(config)# interface gigabitethernet 0/1
R3(config-if)# ipv6 eigrp 1
*Feb 12 22:07:02.014: %DUAL-5-NBRCHANGE: EIGRP-IPv6 1: Neighbor FE80::2
  (GigabitEthernet0/1) is up: new adjacency
R3(config-if)# ipv6 summary-address eigrp 1 ::/0
*Feb 12 22:07:14.234: %DUAL-5-NBRCHANGE: EIGRP-IPv6 1: Neighbor FE80::2
  (GigabitEthernet0/1) is resync: summary configured
R3(config-if)#
```

Verifying Classic EIGRP for IPv6

This section discusses the following commands, which are used to verify EIGRP for IPv6:

- **show ipv6 eigrp neighbors**
- **show ipv6 eigrp topology**
- **show ipv6 route eigrp**
- **show ipv6 protocols**
- **show ipv6 eigrp traffic**
- **show ipv6 eigrp interfaces**
- **show ipv6 interface**
- **ping**
- **show running-config**

R2's EIGRP neighbor adjacencies are verified using the **show ipv6 eigrp neighbors** command, as shown in Example 15-5. Highlighted in the output is the address the neighbor used to form the adjacency, the IPv6 link-local address of the neighbor. This is the source address the neighbor router used in its EIGRPv6 hello messages to establish the adjacency. You can see why it is beneficial to manually configure the router's link-local addresses so that you can more easily recognize the source or destination router. Doing so makes it apparent that fe80::3 is the link-local address of R3, and fe80::1 is the link-local address of R1.

Example 15-5 **show ipv6 eigrp neighbors** *Command on R2*

```
R2# show ipv6 eigrp neighbors
EIGRP-IPv6 Neighbors for AS(1)
H   Address                 Interface    Hold Uptime   SRTT   RTO  Q  Seq
                                         (sec)         (ms)       Cnt Num
1   Link-local address:     Gi0/0          10 03:20:37    2   100  0  5
    FE80::3
0   Link-local address:     Gi0/1          12 04:08:26    3   100  0  4
    FE80::1
R2#
```

Example 15-6 shows the EIGRPv6 topology table displayed using the **show ipv6 eigrp topology** command. The information provided by this command is similar to the equivalent command in EIGRPv4. The only difference is that the next-hop address is an IPv6 link-local address, the source address of EIGRPv6 messages. Highlighted in Example 15-6 is the prefix 2001:db8:cafe:3::/64, with a feasible distance (FD) of 3072, a next-hop address of fe80::2, and a G0/1 exit interface.

Example 15-6 show ipv6 eigrp topology *Command on R1*

```
R1# show ipv6 eigrp topology
EIGRP-IPv6 Topology Table for AS(1)/ID(1.1.1.1)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 2001:DB8:CAFE:3::/64, 1 successors, FD is 3072
        via FE80::2 (3072/2816), GigabitEthernet0/1
P 2001:DB8:CAFE:2::/64, 1 successors, FD is 2816
        via Connected, GigabitEthernet0/1
P ::/0, 1 successors, FD is 3328
        via FE80::2 (3328/3072), GigabitEthernet0/1
P 2001:DB8:CAFE:1::/64, 1 successors, FD is 2816
        via Connected, GigabitEthernet0/0

R1#
```

The **show ipv6 route eigrp** command in Example 15-7 displays the EIGRPv6 routes in R1's IPv6 routing table. The first entry is the summary (default) route advertised by R3. The second entry is a route to 2001:db8:cafe:3::/64. The information for both entries matches their respective entries in the EIGRPv6 topology table in Example 15-6. For example, the information in the routing table and topology table for 2001:db8:cafe:3::/64 has a routing metric (feasible distance) of 3072, a next-hop address of fe80::2, and a G0/1 exit interface.

Example 15-7 show ipv6 route eigrp *Command on R1*

```
R1# show ipv6 route eigrp
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, I1 - ISIS L1, I2 - ISIS L2
       IA - ISIS interarea, IS - ISIS summary, D - EIGRP, EX - EIGRP external
<output omitted for brevity>
D   ::/0 [90/3328]
     via FE80::2, GigabitEthernet0/1
D   2001:DB8:CAFE:3::/64 [90/3072]
     via FE80::2, GigabitEthernet0/1
R1#
```

Example 15-8 displays output from the **show ipv6 protocols** command on R3. This command displays the parameters and current state of all active IPv6 routing protocol processes, including EIGRP for IPv6. The information is similar to the information provided by the equivalent command for EIGRP for IPv4. Highlighted in Example 15-8 is the ::/0 summary route, the administrative distance of 90 for internal routes, and the

administrative distance of 170 for external routes. The internal and external administrative distances are the same for both EIGRPv4 and EIGRPv6. The Router ID and other information associated with the EIGRP process are also shown.

Example 15-8 **show ipv6 protocols** *Command on R3*

```
R3# show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "application"
IPv6 Routing Protocol is "ND"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "eigrp 1"
EIGRP-IPv6 Protocol for AS(1)
  Metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  NSF-aware route hold timer is 240
  Router-ID: 3.3.3.3
  Topology : 0 (base)
    Active Timer: 3 min
    Distance: internal 90 external 170
    Maximum path: 16
    Maximum hopcount 100
    Maximum metric variance 1

  Interfaces:
    GigabitEthernet0/1
  Redistribution:
    None
  Address Summarization:
    ::/0 for Gi0/1
      Summarizing 3 components with metric 2816
R3#
```

The **show ipv6 eigrp traffic** command in Example 15-9 shows the number of EIGRPv6 packets sent and received on R1. These are the same types of EIGRP messages used in EIGRPv4: hello, update, query, reply, and acknowledgement.

Example 15-9 **show ipv6 eigrp traffic** *Command on R1*

```
R1# show ipv6 eigrp traffic
EIGRP-IPv6 Traffic Statistics for AS(1)
  Hellos sent/received: 3211/1043
  Updates sent/received: 5/4
  Queries sent/received: 0/0
  Replies sent/received: 0/0
  Acks sent/received: 4/3
```

```
  SIA-Queries sent/received: 0/0
  SIA-Replies sent/received: 0/0
  Hello Process ID: 221
  PDM Process ID: 188
  Socket Queue: 0/10000/1/0 (current/max/highest/drops)
  Input Queue: 0/2000/1/0 (current/max/highest/drops)

R1#
```

The **show ipv6 eigrp interfaces** command in Example 15-10 lists R1's interfaces enabled for EIGRP for IPv6.

Example 15-10 **show ipv6 eigrp interfaces** *Command on R1*

```
R1# show ipv6 eigrp interfaces
EIGRP-IPv6 Interfaces for AS(1)
                   Xmit Queue   PeerQ        Mean  Pacing Time  Multicast   Pending
Interface  Peers   Un/Reliable  Un/Reliable  SRTT  Un/Reliable  Flow Timer  Routes
Gi0/0      0       0/0          0/0          0     0/0          0           0
Gi0/1      1       0/0          0/0          2     0/0          50          0
R1#
```

In Example 15-11, R1's **show ipv6 interface gigabitethernet 0/0** command verifies that its G0/0 interface is now a member of EIGRPv6 multicast group ff02::a. EIGRPv6 uses ff02::a as the destination address for EIGRP update messages.

Example 15-11 **show ipv6 interface gigabitethernet 0/1** *Command on R1*

```
R1# show ipv6 interface gigabitethernet 0/1
GigabitEthernet0/1 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::1
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:2::1, subnet is 2001:DB8:CAFE:2::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::A
    FF02::FB
    FF02::1:FF00:1
```

```
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 30000)
  ND advertised reachable time is 0 (unspecified)
  ND advertised retransmit interval is 0 (unspecified)
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  ND advertised default router preference is Medium
  Hosts use stateless autoconfig for addresses.
R1#
```

In Example 15-12, the **ping** command is used to verify reachability. In this example, the pings to the interfaces on R2, R3, and ISP routers are all successful.

Example 15-12 **ping** *Command on R1*

```
R1# ping 2001:db8:cafe:2::2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:2::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R1# ping 2001:db8:cafe:3::2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:3::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/8 ms
R1# ping 2001:db8:feed:6::2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:FEED:6::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
R1#
```

Examining the running configs on each router provides a nice way to review the commands used in this chapter to configure classic EIGRPv6. Example 15-13 shows selected output from the running configs on each of the three routers. The commands used in EIGRPv6 or static routing are highlighted for each router.

Example 15-13 show running-config *Command on R1, R2, and R3*

```
R1# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface GigabitEthernet0/0
 no ip address
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:1::1/64
 ipv6 eigrp 1
!
interface GigabitEthernet0/1
 no ip address
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:2::1/64
 ipv6 eigrp 1
!
ipv6 router eigrp 1
 passive-interface GigabitEthernet0/0
 eigrp router-id 1.1.1.1
!
R1#
------------------------------------------------------------
R2# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface GigabitEthernet0/0
 no ip address
 ipv6 address FE80::2 link-local
 ipv6 address 2001:DB8:CAFE:3::1/64
 ipv6 eigrp 1
!
interface GigabitEthernet0/1
 no ip address
 ipv6 address FE80::2 link-local
 ipv6 address 2001:DB8:CAFE:2::2/64
 ipv6 eigrp 1
!
ipv6 router eigrp 1
 eigrp router-id 2.2.2.2
!
```

```
R2#
------------------------------------------------------------
R3# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface GigabitEthernet0/0
 no ip address
 ipv6 address FE80::3 link-local
 ipv6 address 2001:DB8:FEED:6::1/64
!
interface GigabitEthernet0/1
 no ip address
 ipv6 address FE80::3 link-local
 ipv6 address 2001:DB8:CAFE:3::2/64
 ipv6 eigrp 1
 ipv6 summary-address eigrp 1 ::/0
!
ipv6 route ::/0 GigabitEthernet0/0 FE80::4
ipv6 router eigrp 1
 eigrp router-id 3.3.3.3
!
R3#
```

EIGRP Named Mode for IPv6

The classic or traditional way to configure EIGRP requires various parameters to be configured under the interface and EIGRP configuration mode. Configuring EIGRPv4 and EIGRPv6 requires the configuration of a separate EIGRP instance (or process) for each protocol.

EIGRP named mode allows everything to be configured in one place—within EIGRP named configuration mode. EIGRP named mode uses address families to unify the configuration of both EIGRPv4 and EIGRPv6 under the same instance. Named mode is also required when using EIGRP for IPv6 routing with virtual routing and forwarding (VRF).

Note Later in this section, you will see a sample running config of a dual-stacked router configured using EIGRP named mode for both IPv4 and IPv6. It demonstrates how EIGRP is configured for both protocols under the same process.

Figure 15-3 shows the topology used in this section for configuring EIGRP named mode for IPv6. Similar to the previous topology, each router has been configured with the same link-local address on each of its interfaces.

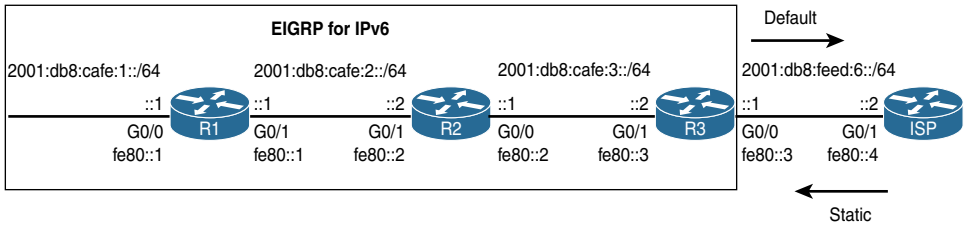


Figure 15-3 *Topology for Configuring EIGRP Named Mode for IPv6*

Note You could configure router R2's interfaces with only link-local addresses, no global unicast addresses. This is because R2 has no end user interfaces. RFC 7404, Using Only Link-Local Addressing inside an IPv6 Network, discusses implementing routing protocols using only link-local addresses on infrastructure links.

Configuring EIGRP Named Mode for IPv6

This section shows how to configure routers R1, R2, and R3 to share routing information within the 2001:db8:cafe::/48 routing domain—but this time using EIGRPv6 named mode. Once again, R3 is configured with a default route via the ISP router, and it propagates the default route to the other routers in the EIGRP domain.

The ISP router is configured with a static route to reach 2001:db8:cafe::/48, as shown in Example 15-14.

Example 15-14 *Configuring Static Routes on ISP*

```
ISP(config)# ipv6 route 2001:db8:cafe::/48 g0/1 fe80::3
```

Example 15-15 shows the EIGRPv6 named mode configuration on router R1. Before configuring EIGRPv6 named mode, you must first enable IPv6 routing by using the **ipv6 unicast-routing** command.

Example 15-15 *Configuring EIGRP Named Mode for IPv6 on R1*

```
R1(config)# ipv6 unicast-routing
R1(config)# router eigrp CAFE-DOMAIN
R1(config-router)# address-family ?
  ipv4  Address family IPv4
  ipv6  Address family IPv6

R1(config-router)# address-family ipv6 unicast autonomous-system 1
R1(config-router-af)# ?
Address Family configuration commands:
  af-interface          Enter Address Family interface configuration
  default               Set a command to its defaults
```

```
  eigrp                EIGRP Address Family specific commands
  exit-address-family  Exit Address Family configuration mode
  help                 Description of the interactive help system
  maximum-prefix       Maximum number of prefixes acceptable in aggregate
  metric               Modify metrics and parameters for address advertisement
  neighbor             Specify an IPv6 neighbor router
  no                   Negate a command or set its defaults
  shutdown             Shutdown address family
  timers               Adjust peering based timers
  topology             Topology configuration mode

R1(config-router-af)# eigrp router-id 1.1.1.1
R1(config-router-af)# af-interface gigabitethernet 0/0
R1(config-router-af-interface)# ?
Address Family Interfaces configuration commands:
  add-paths            Advertise add paths
  authentication       authentication subcommands
  bandwidth-percent    Set percentage of bandwidth percentage limit
  bfd                  Enable Bidirectional Forwarding Detection
  dampening-change     Percent interface metric must change to cause update
  dampening-interval   Time in seconds to check interface metrics
  default              Set a command to its defaults
  exit-af-interface    Exit from Address Family Interface configuration mode
  hello-interval       Configures hello interval
  hold-time            Configures hold time
  next-hop-self        Configures EIGRP next-hop-self
  no                   Negate a command or set its defaults
  passive-interface    Suppress address updates on an interface
  shutdown             Disable Address-Family on interface
  split-horizon        Perform split horizon
  summary-address      Perform address summarization

R1(config-router-af-interface)# passive-interface
R1(config-router-af-interface)# exit-af-interface
R1(config-router-af)# exit-address-family
R1(config-router)#
```

EIGRP named mode has four modes under which most of the configuration is completed:

- Named configuration mode: (config-router)#
- Address-family configuration mode: (config-router-af)#
- Address-family interface configuration mode: (config-router-af-interface)#
- Address-family topology configuration mode: (config-router-af-topology)#

Note The names of the different configuration modes are long and might sound confusing at first. Later in this section you will see condensed output for the configurations on routers R2 and R3, and these modes will become more clear.

The following commands are used in Example 15-15 to configure EIGRPv6 named mode on R1:

- R1(config)# **router eigrp CAFE-DOMAIN**

  This command configures the name of the EIGRP virtual instance CAFE-DOMAIN and enters EIGRPv6 named configuration mode: (config-router)#. Unlike with classic EIGRP, this command does not create or start an EIGRP process. It simply names the virtual instance. The virtual instance name does not have to match the virtual instance name on other routers in the domain. Notice that there is no mention of whether this instance is for IPv4 or IPv6. As you will see next, the **address-family** command is used to define one or more instances and the protocol.

- R1(config-router)# **address-family ?**

  By using the help feature (**?**), you can see that there are two address families (AFs) available: IPv4 and IPv6.

- R1(config-router)# **address-family ipv6 unicast autonomous-system 1**

  The **address-family** command creates an instance of EIGRP and enters address-family configuration mode for the specific protocol (EIGRPv4 or EIGRPv6). An address family refers to the Layer 3 protocol, IPv4 or IPv6.

  The **address-family ipv6 unicast autonomous-system 1** command configures an instance of EIGRP for the IPv6 address family, using the autonomous system number 1. The autonomous system must be the same on all routers in the domain. Using this command puts you in the address-family configuration mode: (config-router-af)#.

- R1(config-router-af)# **?**

  The help command (**?**) in address-family configuration mode displays the commands available in this mode, such as **af-interface**, **eigrp**, **exit-address-family**, and **shutdown**, which are examined later in this section.

- R1(config-router-af)# **eigrp router-id 1.1.1.1**

  The **eigrp router-id 1.1.1.1** AF configuration mode command on R1 configures the 32-bit EIGRP Router ID for the IPv6 address family. A different Router ID can be used for the IPv4 AF. This command is required for EIGRPv6 if there are no loopback or active interfaces with an IPv4 address. The EIGRPv6 process does not start unless there is an EIGRP Router ID.

- R1(config-router-af)# **af-interface gigabitethernet 0/0**

  The **af-interface** command in AF configuration mode puts you in the address-family interface configure mode for the G0/0 interface: (config-router-af-interface)#.

- R1(config-router-af-interface)# **?**

  The help command (**?**) in AF interface configure mode shows the commands available in this mode, including **passive-interface**, **shutdown**, and **summary-address**, which are discussed in this section. Other commands in this mode are used to configure authentication, the hello interval, and the hold-time interval.

- R1(config-router-af-interface)# **passive-interface**

  Because there are no other routers on the G0/0 interface, the **passive-interface** command is used in this mode to suppress EIGRP message on this interface.

- R1(config-router-af-interface)# **exit-af-interface**

  This **exit-af-interface** command (or the shorter form, **exit**) is used to exit address-family interface configuration mode and returns you to address-family configuration mode: (config-router-af)#.

- R1(config-router-af)# **exit-address-family**

  The **exit-address-family** command (or the shorter form, **exit**) exits the IPv6 address-family configuration mode and returns you to named configuration mode: (config-router)#. The **exit** command can be used again to return to global configuration mode: (config)#.

Important There are no commands used to configure EIGRPv6 named mode on an interface. EIGRPv6 named mode is automatically enabled on all active IPv6 interfaces. This will become evident when you configure EIGRPv6 named mode on router R2.

Example 15-16 shows the configuration of EIGRPv6 named mode on router R2. The condensed output, without the help command (**?**), makes it easier to understand the commands in EIGRP named mode. R2 uses the same autonomous system number 1 that is used on R1.

Example 15-16 *Configuring EIGRP Named Mode for IPv6 on R2*

```
R2(config)# ipv6 unicast-routing
R2(config)# router eigrp CAFE-DOMAIN
R2(config-router)# address-family ipv6 unicast autonomous-system 1
R2(config-router-af)# eigrp router-id 2.2.2.2
*Feb 13 20:39:12.646: %DUAL-5-NBRCHANGE: EIGRP-IPv6 1: Neighbor FE80::1 (GigabitEth-
  ernet0/1) is up: new adjacency
R2(config-router-af)# exit-address-family
R2(config-router)# exit
R2(config)#
```

Example 15-16 shows R2 forming an EIGRP neighbor adjacency with R1 immediately after entering the **eigrp router-id 2.2.2.2** command, which is required to start the EIGRPv6 process. Because EIGRPv6 named mode automatically enables all active IPv6 interfaces, R2's G0/1 interface forms an adjacency with R1 at fe80::1 without any manual configuration.

Example 15-17 shows the configuration of EIGRPv6 named mode on router R3. Similar to what you saw on R2, after the Router ID is configured, R3 automatically forms an adjacency on its G0/1 interface with R2.

Example 15-17 *Configuring EIGRP Named Mode for IPv6 on R3*

```
R3(config)# ipv6 unicast-routing
R3(config)# ipv6 route ::/0 gigabitethernet0/0 fe80::4
R3(config)# router eigrp CAFE-DOMAIN
R3(config-router)# address-family ipv6 unicast autonomous-system 1
R3(config-router-af)# eigrp router-id 3.3.3.3
*Feb 13 21:15:01.038: %DUAL-5-NBRCHANGE: EIGRP-IPv6 1: Neighbor FE80::2
  (GigabitEthernet0/1) is up: new adjacency
R3(config-router-af)# exit
R3(config-router)#
```

The default route isn't yet being propagated by R3 into the EIGRPv6 routing domain. Before this is done, examine the IPv6 routing table on R1 in Example 15-18.

Example 15-18 **show ipv6 route eigrp** *Command on R1*

```
R1# show ipv6 route eigrp
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, I1 - ISIS L1, I2 - ISIS L2
       IA - ISIS interarea, IS - ISIS summary, D - EIGRP, EX - EIGRP external
<output omitted for brevity>
D   2001:DB8:CAFE:3::/64 [90/15360]
     via FE80::2, GigabitEthernet0/1
D   2001:DB8:FEED:6::/64 [90/112640]
     via FE80::2, GigabitEthernet0/1
R1#
```

Notice in Example 15-18 that R1's IPv6 routing table has an entry for R3's prefix, 2001:db8:feed:6::/64. R3 is advertising this prefix because EIGRPv6 named mode is automatically enabled on all of R3's IPv6 interfaces, including its G0/0 interface. The **show ipv6 eigrp interfaces** command in Example 15-19 verifies that R3's G0/0 interface is enabled for EIGRPv6.

Example 15-19 **show ipv6 eigrp interfaces** *Command on R3*

```
R3# show ipv6 eigrp interfaces
EIGRP-IPv6 VR(CAFE-DOMAIN) Address-Family Interfaces for AS(1)
                   Xmit Queue   PeerQ        Mean   Pacing Time  Multicast    Pending
Interface  Peers   Un/Reliable  Un/Reliable  SRTT   Un/Reliable  Flow Timer   Routes
Gi0/0      0       0/0          0/0          0      0/0          0            0
Gi0/1      1       0/0          0/0          2      0/0          50           0
R3#
```

The **shutdown** address-family interface command in Example 15-20 disables EIGRPv6 named mode on a specific interface. This command does not shut down the physical interface, as shown with the **show ipv6 interface brief** command. The **show ipv6 eigrp interfaces** command verifies that the G0/0 interface is no longer enabled for EIGRP.

Note Most of the configuration examples in this section begin with the **router eigrp CAFE-DOMAIN** command and end with exiting to the global configuration command prompt. This should help familiarize you with the hierarchical structure of these commands.

Example 15-20 *Disabling EIGRP for IPv6 on an Interface*

```
R3(config)# router eigrp CAFE-DOMAIN
R3(config-router)# address-family ipv6 unicast autonomous-system 1
R3(config-router-af)# af-interface gigabitethernet 0/0
R3(config-router-af-interface)# shutdown
R3(config-router-af-interface)# exit-af-interface
R3(config-router-af)# end
R3# show ipv6 interface brief
GigabitEthernet0/0      [up/up]
    FE80::3
    2001:DB8:FEED:6::1
GigabitEthernet0/1      [up/up]
    FE80::3
    2001:DB8:CAFE:3::2
R3# show ipv6 eigrp interfaces
EIGRP-IPv6 VR(CAFE-DOMAIN) Address-Family Interfaces for AS(1)
                   Xmit Queue   PeerQ        Mean   Pacing Time  Multicast    Pending
Interface  Peers   Un/Reliable  Un/Reliable  SRTT   Un/Reliable  Flow Timer   Routes
Gi0/1      1       0/0          0/0          1      0/0          50           0
R3#
```

Because R3's G0/0 interface has now been disabled as an EIGRPv6 interface, R3 is no longer advertising its prefix 2001:db8:feed:6::/64. You can verify this by examining R1's IPv6 routing table in Example 15-21. Notice that R1's IPv6 routing table no longer includes this prefix.

Example 15-21 **show ipv6 route eigrp** *Command on R1*

```
R1# show ipv6 route eigrp
IPv6 Routing Table - default - 6 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, I1 - ISIS L1, I2 - ISIS L2
       IA - ISIS interarea, IS - ISIS summary, D - EIGRP, EX - EIGRP external
<output omitted for brevity>
D   2001:DB8:CAFE:3::/64 [90/15360]
     via FE80::2, GigabitEthernet0/1
R1#
```

Similar to with classic EIGRPv6, the **summary-address** address-family interface command can be used to advertise the summary route ::/0 into the EIGRPv6 domain. Example 15-22 shows the **summary-address ::/0** command on R3's G0/1 AF interface.

Example 15-22 *Advertising a ::/0 Summary Route Within the EIGRPv6 Domain*

```
R3(config)# router eigrp CAFE-DOMAIN
R3(config-router)# address-family ipv6 unicast autonomous-system 1
R3(config-router-af)# af-interface gigabitethernet 0/1
R3(config-router-af-interface)# summary-address ::/0
*Feb 13 22:12:48.185: %DUAL-5-NBRCHANGE: EIGRP-IPv6 1: Neighbor FE80::2
  (GigabitEthernet0/1) is resync: summary configured
R3(config-router-af-interface)# exit-af-interface
R3(config-router-af)# exit
R3(config-router)#
```

Example 15-23 displays R1's IPv6 routing table. Notice the summary (default) route learned via EIGRPv6.

Example 15-23 **show ipv6 route eigrp** *Command on R1*

```
R1# show ipv6 route eigrp
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, I1 - ISIS L1, I2 - ISIS L2
       IA - ISIS interarea, IS - ISIS summary, D - EIGRP, EX - EIGRP external
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, a - Application
```

```
D    ::/0 [90/20480]
     via FE80::2, GigabitEthernet0/1
D   2001:DB8:CAFE:3::/64 [90/15360]
     via FE80::2, GigabitEthernet0/1
R1#
```

Note Earlier in this chapter, you saw the **shutdown** address-family interface command used on R3's G0/0 to disable EIGRP on the interface and therefore keep R3 from advertising the 2001:db8:feed:6::/64 prefix into the EIGRPv6 routing domain. The **summary-address ::/0** address-family interface command on G0/1 has the same effect of preventing R3 from advertising this prefix. This is because the ::/0 summary route covers the 2001:db8:feed:6::/64 prefix in its summarization. The **summary-address** command has the same effect on the IPv6 routing tables as the **shutdown** command but doesn't disable EIGRPv6 on R3's interface. The **summary-address** command doesn't advertise the configured prefix, but it does peer with neighbors on that link. It is best practice to use the **shutdown** command on G0/1 to prevent R3 from sending EIGRP messages to ISP.

Verifying EIGRP Named Mode for IPv6

The verification commands used earlier for classic EIGRPv6 are the same commands that can be used to verify EIGRPv6 named mode. This section shows three of these commands used with EIGRPv6 named mode with output that varies slightly from classic EIGRPv6:

- **show ipv6 route eigrp**
- **show ipv6 protocols**
- **show running-config**

Example 15-24 displays output from the **show ipv6 route eigrp** command on R1. Compare this output with output from the same command used for classic EIGRPv6 in Example 15-7. Notice the differences in the highlighted metrics. These metrics vary due to EIGRPv6 named mode using a 64-bit-wide metric compared to the 32-bit traditional composite metric used by classic EIGRPv6.

Example 15-24 **show ipv6 route eigrp** *Command on R1 Using EIGRPv6 Named Mode*

```
R1# show ipv6 route eigrp
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, I1 - ISIS L1, I2 - ISIS L2
       IA - ISIS interarea, IS - ISIS summary, D - EIGRP, EX - EIGRP external
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
```

```
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, a - Application
D   ::/0 [90/20480]
     via FE80::2, GigabitEthernet0/1
D   2001:DB8:CAFE:3::/64 [90/15360]
     via FE80::2, GigabitEthernet0/1
R1#
```

Output from the **show ipv6 route eigrp** command on R3 is shown in Example 15-25. This example highlights some of the differences between this output and the output for the same command for classic EIGRPv6 (refer to Example 15-8). Highlights in Example 15-25 show that EIGRP for IPv6 is using the "Address-Family Protocol for AS(1)." The additional K6 metric weight is shown, and you can see that EIGRPv6 named mode uses a 64-bit metric.

Example 15-25 **show ipv6 protocols** *Command on R3 Using EIGRPv6 Named Mode*

```
R3# show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "application"
IPv6 Routing Protocol is "ND"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "eigrp 1"
EIGRP-IPv6 VR(CAFE-DOMAIN) Address-Family Protocol for AS(1)
  Metric weight K1=1, K2=0, K3=1, K4=0, K5=0 K6=0
  Metric rib-scale 128
  Metric version 64bit
  NSF-aware route hold timer is 240
  Router-ID: 3.3.3.3
  Topology : 0 (base)
    Active Timer: 3 min
    Distance: internal 90 external 170
    Maximum path: 16
    Maximum hopcount 100
    Maximum metric variance 1
    Total Prefix Count: 4
    Total Redist Count: 0

  Interfaces:
    GigabitEthernet0/1
  Redistribution:
    None
  Address Summarization:
    ::/0 for Gi0/1
      Summarizing 3 components with metric 1310720
R3#
```

Example 15-26 shows selected output from the running configs for routers R1, R2, and R3. The commands used to configure EIGRPv6 named mode on the three routers and a static route on R3 are highlighted. To see the differences between EIGRPv6 named mode and classic EIGPRv6, compare this output to the running configs on the same routers in Example 15-13.

Example 15-26 **show running-config** *Command on R1, R2, and R3*

```
R1# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface GigabitEthernet0/0
 no ip address
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:1::1/64
!
interface GigabitEthernet0/1
 no ip address
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:2::1/64
!
router eigrp CAFE-DOMAIN
 !
 address-family ipv6 unicast autonomous-system 1
  !
  af-interface GigabitEthernet0/0
   passive-interface
  exit-af-interface
  !
  topology base
  exit-af-topology
  eigrp router-id 1.1.1.1
 exit-address-family
!
R1#
------------------------------------------------------------
R2# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
```

```
interface GigabitEthernet0/0
 no ip address
 ipv6 address FE80::2 link-local
 ipv6 address 2001:DB8:CAFE:3::1/64
!
interface GigabitEthernet0/1
 no ip address
 ipv6 address FE80::2 link-local
 ipv6 address 2001:DB8:CAFE:2::2/64
!
router eigrp CAFE-DOMAIN
 !
 address-family ipv6 unicast autonomous-system 1
  !
  topology base
  exit-af-topology
  eigrp router-id 2.2.2.2
 exit-address-family
!
R2#
------------------------------------------------------------
R3# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface GigabitEthernet0/0
 no ip address
 ipv6 address FE80::3 link-local
 ipv6 address 2001:DB8:FEED:6::1/64
!
interface GigabitEthernet0/1
 no ip address
 ipv6 address FE80::3 link-local
 ipv6 address 2001:DB8:CAFE:3::2/64
!
router eigrp CAFE-DOMAIN
 !
 address-family ipv6 unicast autonomous-system 1
  !
  af-interface GigabitEthernet0/1
   summary-address ::/0
  exit-af-interface
  !
```

```
  af-interface GigabitEthernet0/0
   shutdown
  exit-af-interface
  !
  topology base
  exit-af-topology
  eigrp router-id 3.3.3.3
 exit-address-family
!
ipv6 route ::/0 GigabitEthernet0/0 FE80::4
!
R3#
```

Comparing EIGRP Named Mode for IPv4 and IPv6

With classic EIGRP for IPv4 and IPv6 you need to configure each protocol separately. Here is how it looks in classic EIGRPv4:

```
R1(config)# router eigrp 1
R1(config-router)# <EIGRPv4 commands>
```

And here is what you use in classic EIGRPv6:

```
R1(config)# ipv6 router eigrp 2
R1(config-rtr)# <EIGRPv6 commands>
```

EIGRP named mode provides the benefit of allowing you to configure both the IPv4 and IPv6 address families in one place. To see the advantage of this, it is helpful to show the running config of a router configured with both the IPv4 and IPv6 address families.

Example 15-27 shows the EIGRP named mode configuration on R1 for the IPv4 address family. The autonomous system number does not have to be the same as the autonomous system number used for the IPv6 AF. However, it must match the autonomous system number of other EIGRPv4 routers in the same domain.

The **eigrp router-id** command is not required for EIGRPv4 because the Router ID can be obtained from either the highest IPv4 address of a loopback interface or a physical interface. Also, the Router ID for the IPv4 AF doesn't have to match the Router ID for the IPv6 AF.

Similar to in the original method of configuring EIGRP for IPv4, the **network** command is used in AF-configuration mode to specify which interfaces are enabled for EIGRPv4. Unlike with EIGRPv6 named mode, the interfaces for EIGRPv4 named mode are not automatically enabled.

Example 15-27 *Configuring EIGRP Named Mode for IPv4 on R1*

```
R1(config)# router eigrp CAFE-DOMAIN
R1(config-router)# address-family ipv4 unicast autonomous-system 4
R1(config-router-af)# eigrp router-id 1.1.1.1
R1(config-router-af)# network 192.168.1.0 0.0.0.255
R1(config-router-af)# network 192.168.2.0 0.0.0.255
R1(config-router-af)# exit-address-family
R1(config-router)# exit
R1(config)#
```

The **show running-config | section router eigrp** command in Example 15-28 shows selected output for R1's EIGRP virtual instance CAFE-DOMAIN. The address family sections for IPv4 and IPv6 are highlighted.

Example 15-28 *Displaying R1's EIGRP Named Mode Configuration for IPv6 and IPv4 AF*

```
R1# show running-config | section router eigrp
router eigrp CAFE-DOMAIN
 !
 address-family ipv6 unicast autonomous-system 1
  !
  af-interface GigabitEthernet0/0
   passive-interface
  exit-af-interface
  !
  topology base
  exit-af-topology
  eigrp router-id 1.1.1.1
 exit-address-family
 !
 address-family ipv4 unicast autonomous-system 4
  !
  topology base
  exit-af-topology
  network 192.168.1.0
  network 192.168.2.0
  eigrp router-id 1.1.1.1
 exit-address-family
R1#
```

Note See the references section at the end of this chapter for more information on EIGRPv4 and EIGRPv6, both classic and named mode.

Summary

This chapter discusses the similarities and differences between EIGRPv4 and EIGRPv6. It also covers configuration and verification for both classic EIGRPv6 and EIGRPv6 named mode.

Many of the processes and operational functionalities are the same in both EIGRPv4 and EIGRPv6. EIGRP has simply been redesigned and extended to support IPv6. The following are true of both routing protocols:

- They are both distance-vector routing protocols.
- They both use DUAL as the computational algorithm.
- They both use bandwidth and delay for the default metric.
- They both use RTP as the transport protocol.
- They both use a 32-bit EIGRP Router ID.

However, there are also some differences, including the following:

- EIGRPv4 advertises IPv4 prefixes, whereas EIGRPv6 advertises IPv6 prefixes.
- EIGRPv4 messages are sent from the IPv4 address of the interface, whereas EIGRPv6 messages are sent from the IPv6 link-local address of the interface.
- EIGRPv4 sends updates using the 224.0.0.10 multicast address, whereas EIGRPv6 sends updates using the fe80::a multicast address.

The configuration of classic EIGRPv6 has its roots in the original EIGRPv4 protocol. The following commands are included in the syntax used by the classic EIGRPv6 commands in this chapter:

- Router(config)# **ipv6 router eigrp** *autonomous-system-number*

  This command creates an EIGRPv6 routing process, using an autonomous system number common to the routing domain.

- Router(config-rtr)# **eigrp router-id** *router-id*

  This command, which configures the EIGRP Router ID, is required if there are no loopback or active physical interfaces with an IPv4 address.

- Router(config-rtr)# **passive-interface** *interface-type interface-number*

  This command suppresses EIGRP hello messages and routing updates from being sent out an interface.

- Router(config-if)# **ipv6 eigrp** *autonomous-system-number*

  This command enables EIGRPv6 directly on the interface.

- Router(config-if)# **ipv6 summary-address eigrp** *autonomous-system-number prefix/prefix-length*

  This command advertises a summary route into the EIGRP routing domain.

EIGRP named mode allows you to configure everything in one place—in EIGRP named configuration mode. EIGRP named mode uses address families to unify the configuration of both EIGRPv4 and EIGRPv6 under the same instance.

EIGRP named mode has four modes under which most of the configuration is completed:

- Named configuration mode: (config-router)#
- Address-family configuration mode: (config-router-af)#
- Address-family interface configuration mode: (config-router-af-interface)#
- Address-family topology configuration mode: (config-router-af-topology)#

The EIGRP named mode commands in this chapter use the following syntax:

- Router(config)# **router eigrp** *virtual-instance-name*

  This command configures the virtual instance name.

- Router(config-router)# **address-family** *address-family* **unicast autonomous-system** *autonomous-system-number*

  This command creates an instance of EIGRP and enters address-family configuration mode for the specific protocol (IPv4 or IPv6).

- Router(config-router-af)# **eigrp router-id** *router-id*

  This command configures the 32-bit EIGRP Router ID.

- Router(config-router-af)# **af-interface** *interface-type interface-number*

  This command puts you in the address-family interface configure mode for the specific interface.

- Router(config-router-af-interface)# **passive-interface**

  This command suppresses EIGRP messages on an interface.

- Router(config-router-af-interface)# **shutdown**

  This command disables EIGRPv6 on a specific interface, without shutting down the physical interface.

There are no commands used to configure EIGRPv6 named mode on an interface. EIGRPv6 named mode is automatically enabled on all active IPv6 interfaces.

The commands used to verify EIGRPv6 include the following:

- **show ipv6 eigrp neighbors**
- **show ipv6 eigrp topology**

- **show ipv6 route eigrp**
- **show ipv6 protocols**
- **show ipv6 eigrp traffic**
- **show ipv6 eigrp interfaces**
- **show ipv6 interface**
- **ping**
- **show running-config**

Review Questions

1. Which EIGRP routing protocol uses a 32-bit Router ID?
   a. EIGRPv4
   b. EIGRPv6
   c. Both EIGRPv4 and EIGRPv6
2. Which EIGRP routing protocol uses a 64-bit-wide metric?
   a. Classic EIGRPv4
   b. Both EIGRPv4 named mode and EIGRPv6 named mode
   c. Both Classic EIGRPv4 and EIGRPv6 named mode
3. Which EIGRP routing protocol uses DUAL as the computational algorithm?
   a. EIGRPv4
   b. EIGRPv6
   c. Both EIGRPv4 and EIGRPv6
4. Which EIGRP routing protocol sends EIGRP messages using its link-local address?
   a. EIGRPv4
   b. EIGRPv6
   c. Both EIGRPv4 and EIGRPv6
5. Which EIGRP routing protocol sends EIGRP messages to the 224.0.0.10 multicast address?
   a. EIGRPv4
   b. EIGRPv6
   c. Both EIGRPv4 and EIGRPv6
6. How is an interface enabled for classic EIGRPv6?
7. What command creates an EIGRPv6 routing process using classic EIGRPv6?
8. Which EIGRP routing protocol requires you to configure each protocol separately?
   a. Classic EIGRPv4 only
   b. Classic EIGRPv6 only
   c. Both classic EIGRPv4 and classic EIGRPv6

9. Match the router prompt to the proper configuration mode for EIGRP named mode.
   Router(config-router)#
   Router(config-router-af)#
   Router(config-router-af-interface)#
   Router(config-router-af-topology)#

   a. Address-family interface configuration mode
   b. Address-family topology configuration mode
   c. Address-family configuration mode
   d. Named configuration mode

10. What command is used in EIGRP named mode to configure EIGRPv4 or EIGRPv6?
11. How is an IPv6 interface enabled for EIGRPv6 named mode?

References

RFC

RFC 7868, *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)*, D. Savage, Cisco Systems, tools.ietf.org/html/rfc7868, May 2016.

Websites

IPv6 Design and Deployment LiveLessons, http://www.ciscopress.com/store/ipv6-design-and-deployment-livelessons-9780134655512.

Cisco IOS IPv6 Command Reference, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book.html.

IP Routing: EIGRP Configuration Guide, www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_eigrp/configuration/xe-3s/ire-xe-3s-book.html.

Configure EIGRP Named Mode, www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/200156-Configure-EIGRP-Named-Mode.html.

Books

Implementing Cisco IP Routing (ROUTE) Foundation Learning Guide (CCNP ROUTE 300-101), by Diane Teare, Cisco Press.

IP Routing on Cisco IOS, IOS XE, and IOS XR: An Essential Guide to Understanding and Implementing IP Routing Protocols, by Brad Edgework, Aaron Foss, and Ramiro Rios, Cisco Press.

Routing TCP/IP, by Jeff Doyle, Cisco Press.

This page intentionally left blank

Chapter 16

OSPFv3

Open Shortest Path First (OSPF) is a link-state routing protocol that was developed as a replacement for the distance-vector routing protocol Routing Information Protocol (RIP). RIP was an acceptable routing protocol in the early days of networking, but its reliance on a limited hop count as the only metric for choosing the best route quickly became unacceptable in larger networks that needed a more robust routing solution. OSPF is a classless routing protocol that uses the concept of areas for scalability.

In 1989, the specifications for OSPFv1 were published in RFC 1131, *The OSPF Specification*. In 1991, John Moy introduced OSPFv2 in RFC 1247, *OSPF Version 2*. OSPFv2 offered significant technical improvements over OSPFv1. Both OSPFv1 and OSPFv2 are link-state routing protocols for advertising IPv4 prefixes. In 1998, the OSPFv2 specification was updated in RFC 2328, *OSPF Version 2*, which is the current RFC for OSPFv2. RFC 2328 defines the OSPF metric as an arbitrary value called *cost*. Cisco IOS uses bandwidth as the OSPF cost metric.

In 1999, OSPFv3 for IPv6 was published in RFC 2740, *OSPF for IPv6*, written by John Moy, Rob Coltun, and Dennis Ferguson. OSPFv3 was later updated in RFC 5340, *OSPF for IPv6*.

In 2010, OSPFv3 included support for address families in RFC 5838, *Support of Address Families in OSPFv3*. The original implementation of OSPFv3 supported only IPv6. With the introduction of OSPFv3 support for address families (AFs), OSPFv3 now can support both IPv4 and IPv6 address families within a single process.

Note The development of OSPF is an interesting topic covered in John Moy's book *OSPF: Anatomy of an Internet Routing Protocol*.

Note This chapter, which focuses on the configuration of OSPFv3, assumes that you have some knowledge of OSPF. However, this chapter reviews many OSPF concepts, so even if you are new to OSPF, you will find this chapter helpful. If you are new to OSPF or wish to learn more, see the recommendations provided in the references section at the end of the chapter.

This chapter discusses two methods to configure OSPFv3:

- Traditional OSPFv3
- OSPFv3 with address families

Comparing OSPFv2 and OSPFv3

OSPFv2 and OSPFv3 share the same core functionality and operations. However, there are some significant changes to OSPFv3. OSPFv3 is not only a new protocol implementation of OSPF for IPv6 but also a major rewrite of the internals of the protocol. But as you will see in this section, the same link-state concepts and hierarchical design apply to both OSPFv2 and OSPFv3.

Note In this chapter, OSPF refers to OSPF2, OSPFv3, and OSPFv3 with address families.

Table 16-1 provides an overview of OSPF and also a comparison between OSPFv2, traditional OSPFv3, and OSPFv3 with address families (AF).

Table 16-1 *Comparing OSPFv2, Traditional OSPFv3, and OSPFv3 with AF*

	OSPFv2	Traditional OSPFv3	OSPFv3 with AF
OSPF version	OSPFv2	OSPFv3	OSPFv3
Advertised routes	IPv4 prefixes	IPv6 prefixes	IPv4 and IPv6 prefixes
Link state?	Yes	Yes	Yes
Metric	Cost Cisco: bandwidth	Cost Cisco: bandwidth	Cost Cisco: bandwidth
Multi-area support?	Yes	Yes	Yes
Router ID	32-bit	32-bit	32-bit
DR and BDR?	Yes	Yes	Yes
Layer 3 encapsulation	IPv4	IPv6	IPv6 for both IPv4 and IPv6 AF
Source address	IPv4 address	IPv6 link-local address	IPv6 link-local address
Destination address, AllSPFRouters	224.0.0.5	ff02::5	ff02::5
Destination address, AllDRouters	224.0.0.6	ff02::6	ff02::6

	OSPFv2	Traditional OSPFv3	OSPFv3 with AF
IPv6 unicast routing	N/A	Required	Required, even if only IPv4 AF is configured
Authentication	Plain text and MD5	IPsec	IPsec, HMAC SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512

As you can see in Table 16-1, OSPFv2 and both types of OSPFv3 have the same basic operations and functionality. Both protocols share the same basic concepts and processes, including LSA distribution, DR/BDR on multi-access networks, and the OSPF metric.

Other similarities include the aging of the link-state database and flushing LSAs from the routing domain through the premature aging process. Cisco routers running OSPFv2 or OSPFv3 periodically flood their link states every 30 minutes and flush LSAs after 60 minutes.

The RFCs for both OSPFv2 and OSPFv3 define the metric as the cost of sending packets out the interface. In the Cisco implementation, the metric is the cumulative bandwidths of the outgoing interfaces to the destination network (10^8/bandwidth in b/s). The reference bandwidth (10^8) in both OSPFv2 and OSPFv3 can be modified using the **auto-cost reference-bandwidth** *ref-bw* command in router configuration mode. In both cases, the *ref-bw* (reference bandwidth) parameter is in megabits per second. Because **auto-cost reference-bandwidth** *ref-bw* is entered in router configuration mode, it only influences the OSPF metric where it was configured. For example, if this command was entered for OSPFv3, it does not affect the OSPFv2 routing metrics.

Note The **auto-cost reference-bandwidth** *ref-bw* command should be configured on all routers to avoid potential routing issues.

The concept of multiple areas in OSPFv3 is the same as in OSPFv2: to minimize link-state flooding and provide better stability within the OSPF domain. In addition, stub, totally stubby, and NSSA (not-so-stubby area) area types are designed to minimize link-state database and routing table sizes for the areas' internal routers. These same area types used in OSPFv2 are also available in OSPFv3.

Both OSPFv2 and OSPFv3 use a 32-bit value for the Router ID, represented in dotted-decimal notation. If the router has not been configured with an IPv4 address, the OSPF **router-id** command must be used to configure the Router ID. The process of determining the 32-bit Router ID is the same in both protocols. It is best practice to use the **router-id** command to configure the OSPF Router ID. If this command isn't used, OSPF uses the highest IPv4 address of a loopback interface. If there are no loopback interfaces, OSPF uses the highest IPv4 address of any of its active interfaces.

OSPFv3 uses the same five basic packet types as OSPFv2:

- Hello
- Database Description (DBD)
- Link-State Request (LSR)
- Link-State Update (LSU)
- Link-State Acknowledgment (LSAck)

However, there are some differences. The main difference between OSPF2, traditional OSPFv3, and OSPFv3 with address families is related to their support for advertising IP prefixes. OSPFv2 advertises only IPv4 prefixes, traditional OSPFv3 advertises only IPv6 prefixes, and OSPFv3 with address families advertises both IPv4 and IPv6 prefixes.

The AllSPFRouters (all SPF routers) multicast address in OSPFv3 is ff02::5, equivalent to 224.0.0.5 in OSPFv2. The AllDRouters (all DR routers) multicast address in OSPFv3 is ff02::6, equivalent to 224.0.0.6 in OSPFv2. Both OSPFv3 multicast addresses have link-local scope.

OSPFv2 messages are sourced from the IPv4 address of the exit interface. With OSPFv3, messages are sourced using the IPv6 link-local address of the exit interface.

With OSPFv3, a routing process does not need to be explicitly created, as it does with OSPFv2. Enabling OSPFv3 on an interface automatically creates the OSPFv3 routing process.

With IPv6, you can configure multiple address prefixes on an interface. With OSPFv3, all address prefixes on an interface are included by default. You cannot select some address prefixes to be imported into OSPFv3; either all address prefixes on an interface are imported or no address prefixes on an interface are imported.

You may already be familiar with OSPFv2 Link-State Updates and the different types of Link-State Advertisements (LSAs). OSPFv3 added two new types of LSAs: Link-LSA (Type 8) and Intra-Area-Prefix-LSA (Type 9). Some of the existing LSA types were also renamed. Table 16-2 shows the types of LSAs for OSPFv2 and OSPFv3. (LSA types are beyond the scope of this book. For further information, refer to RFC 5340, *OSPF for IPv6*, and references listed at the end of the chapter.)

Table 16-2 *Comparing LSAs for OSPFv2 and OSPFv3*

OSPFv2 LSAs		OSPFv3 LSAs	
Type	**Name**	**LS Type Code**	**Name**
1	Router LSA	0x2001	Router LSA
2	Network LSA	0x2002	Network LSA
3	Network Summary LSA	0x2003	Inter-Area Prefix LSA
4	ASBR Summary LSA	0x2004	Inter-Area Router LSA

OSPFv2 LSAs		OSPFv3 LSAs	
Type	**Name**	**LS Type Code**	**Name**
5	AS-External LSA	0x4005	AS-External LSA
6	Group Membership LSA	0x2006	Group Membership LSA (This LSA was defined for Multicast extensions to OSPF [MOSPF], which has since been deprecated.)
7	NSSA-External LSA	0x2007	Type-7 LSA
		0x0008	Link LSA
		0x2009	Intra-Area Prefix LSA

Note OSPFv3 LS type codes are commonly referred to by their last digit (1 through 9). The first hexadecimal digit of LS type codes 0x4005 and 0x0008 differ from the others because of their flooding scope. See RFC 5340, A.4.2.1 for more information.

Traditional OSPFv3

OSPFv2 supports only IPv4 prefixes, whereas traditional OSPFv3 supports only IPv6 prefixes. When implementing both protocols in a dual-stack environment, OSPFv2 and traditional OSPFv3 maintain separate neighbor tables and link-state databases (LSDBs) for their respective protocols, as shown in Figure 16-1. Later in this chapter, you will see that this differs from OSPFv3 with address families, which uses the same neighbor table and LSDB for routing both IPv4 and IPv6.

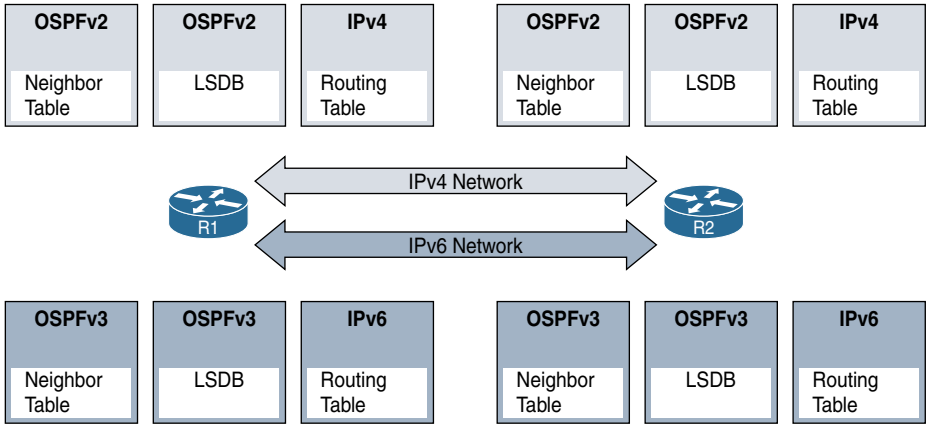


Figure 16-1 *Separate Neighbor Tables and LSDBs for OSPFv2 and OSPFv3*

Configuring traditional OSPFv3 is similar to configuring OSPFv2. Figure 16-2 shows the topology used in this section. Each router is configured with the same link-local address on each of its interfaces. This has been done here, as in Chapter 15, "EIGRP for IPv6," to make it easier to recognize the source address of the OSPFv3 messages. The multi-area OSPFv3 routing domain consists of two areas: the backbone area, area 0, and a totally stubby area, area 51.

Router R1 is an OSPF autonomous system boundary router (ASBR) with an external interface connected to the ISP router. Router R1 has two interfaces in area 0.

Router R2 is an area border router (ABR) with interfaces in area 0 and in area 51. Area 51 is a totally stubby area, which means routers within this area do not receive any prefixes from other areas but receive a default route from the ABR. Router R3 is an internal router in the totally stubby area, area 51.

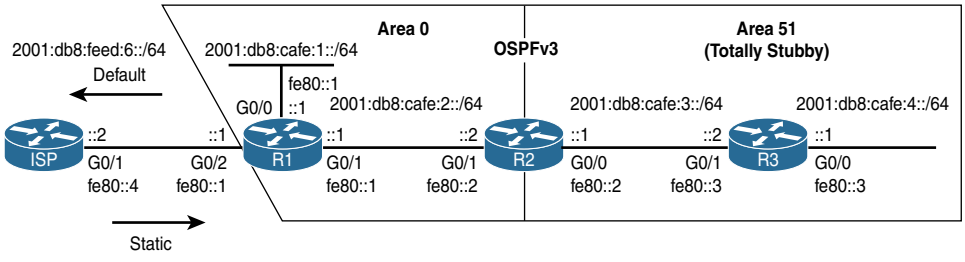


Figure 16-2 *Topology for Traditional OSPFv3*

Note You could configure router R2's interfaces with only link-local addresses, no global unicast addresses. This is because R2 has no end user interfaces. RFC 7404, Using Only Link-Local Addressing inside an IPv6 Network, discusses implementing routing protocols using only link-local addresses on infrastructure links.

Configuring Traditional OSPFv3

This section shows how to configure routers R1, R2, and R3 to share routing information within the multi-area OSPFv3 routing domain, 2001:db8:cafe::/48. The ASBR, R1, is configured with a default route via the ISP router, and to propagate the default route to the other routers in the OSPFv3 domain.

The ISP router is configured with a static route to reach 2001:db8:cafe::/48, as shown in Example 16-1.

Example 16-1 *Configuring a Static Route on ISP*

```
ISP(config)# ipv6 route 2001:db8:cafe::/48 g0/1 fe80::1
```

ASBR and Advertising a Default Route

Example 16-2 shows the configuration of traditional OSPFv3 on router R1. R1 is the ASBR that connects its OSFPv3 domain to ISP. A default static route is configured on R1 and advertised into the OSPFv3 domain.

Example 16-2 shows the **ipv6 unicast-routing** command configured before any dynamic IPv6 routing protocol can be enabled. R1 is also configured with an IPv6 default static route via ISP.

Example 16-2 *Configuring OSPFv3 on R1*

```
R1(config)# ipv6 unicast-routing
R1(config)# ipv6 route ::/0 gigabitethernet0/2 fe80::4
R1(config)# ipv6 router ospf 1
*Feb 16 21:53:33.902: %OSPFv3-4-NORTRID: Process OSPFv3-1-IPv6 could not pick a
  router-id, please configure manually
R1(config-rtr)# router-id 1.1.1.1
R1(config-rtr)# passive-interface gigabitethernet 0/0
R1(config-rtr)# default-information originate
R1(config-rtr)# exit
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 ospf 1 area 0
R1(config-if)# exit
R1(config)# interface gigabitethernet 0/1
R1(config-if)# ipv6 ospf 1 area 0
R1(config-if)# exit
R1(config)#
```

The following is a brief description of the commands used to configure traditional OSPFv3 on R1 in Example 16-2:

- R1(config)# **ipv6 router ospf 1**

  This **ipv6 router ospf** *process-id* command enables the OSPFv3 routing process using process ID 1. The process ID is locally significant and does not have to match other routers in the OSPFv3 routing domain. However, it is typically best practice to use the same process ID on all routers in the domain.

  You may receive the following message when enabling the OSPFv3 process on a router without any IPv4 loopback or interface addresses:

  ```
  *Feb 16 21:53:33.902: %OSPFv3-4-NORTRID: Process OSPFv3-1-IPv6 could not pick
  a router-id, please configure manually
  ```

  Without an IPv4 address on any of the interfaces, this is a reminder that you must manually configure the OSPF Router ID.

- R1(config-rtr)# **router-id 1.1.1.1**

The **router-id** *router-id* command configures the OSPFv3 Router ID. This command is required if there are no loopback or active physical interfaces with an IPv4 address.

- R1(config-rtr)# **passive-interface gigabitethernet 0/0**

  The **passive-interface** *interface-type interface-number* interface command suppresses OSPFv3 hellos and other OSPF messages from being sent out an interface. R1's G0/0 interface is configured as a passive interface because it doesn't have any OSPFv3 neighbors on that interface.

- R1(config-rtr)# **default-information originate**

  R1 has a default static route via ISP. To distribute the default route on R1 to other routers in the OSPFv3 domain, use the command **default-information originate**, which is explained in more detail later in this section.

- R1(config-if)# **ipv6 ospf 1 area 0**

  Similar to other interior gateway protocols (IGPs), you must enable OSPFv3 on interfaces to advertise the prefix of the interface and establish neighbor adjacencies with other routers. The **ipv6 ospf 1 area 0** command is used in interface mode to enable OSPFv3 directly on the interface.

  OSPFv3 is enabled directly on the interface using the **ipv6 ospf** *process-id* **area** *area-id* interface command. The *process-id* parameter identifies the specific routing process and must be the same as the process ID used to create the routing process in the **ipv6 router ospf** *process-id* command. The *area-id* parameter is the area that is to be associated with the OSPFv3 interface.

  Both of R1's OSPFv3 interfaces are in area 0, the backbone area. Notice that its G0/2 interface is not enabled for OSPFv3 because ISP is not an OSPFv3 neighbor, and it's not advertising this prefix into the OSPFv3 routing domain.

Area Border Router with Totally Stubby Area

Example 16-3 shows the OSPFv3 configuration of the ABR, R2. Although this is not required, the same *process-id*, 1, is used that was configured on R1. As shown in the example, after you enable OSPFv3 on R2's G0/1 interface, R2 forms a full adjacency with its neighbor R1.

R2 is an ABR with its G0/1 interface in area 0 and its G0/0 interface in area 51. Because area 51 is a totally stubby area, the ABR doesn't advertise prefixes from other areas, into area 51. The ABR, router R2, injects only a default route into the totally stubby area. To configure a totally stubby area, you use the command **area** *area-id* **stub no-summary** under the OSPFv3 router process. The **no-summary** option tells the router that this area does not receive summary (inter-area) routes or external routes, which makes the area totally stubby instead of a stub area.

Note To configure an area as a stub area, use the **area** *area-id* **stub** command in OSPFv3 router configuration mode. A stub area includes prefixes from other areas (inter-area routes) but blocks any external routes. The ABR injects a default route into the stub area.

Router R2 has two interfaces in two different areas. The *area-id* in the **ipv6 ospf** interface command in Example 16-3 defines the area for each interface. R2's G0/1 interface is in the backbone area, area 0. Its G0/0 interface is in area 51, which is configured as a totally stubby area.

Example 16-3 *Configuring OSPFv3 on R2*

```
R2(config)# ipv6 unicast-routing
R2(config)# ipv6 router ospf 1
*Feb 16 22:09:33.992: %OSPFv3-4-NORTRID: Process OSPFv3-1-IPv6 could not pick a
  router-id, please configure manually
R2(config-rtr)# router-id 2.2.2.2
R2(config-rtr)# area 51 stub no-summary
R2(config-rtr)# exit
R2(config)# interface gigabitethernet 0/1
R2(config-if)# ipv6 ospf 1 area 0
*Feb 16 22:10:55.144: %OSPFv3-5-ADJCHG: Process 1, Nbr 1.1.1.1 on GigabitEthernet0/1
  from LOADING to FULL, Loading Done
R2(config-if)# exit
R2(config)# interface gigabitethernet 0/0
R2(config-if)# ipv6 ospf 1 area 51
R2(config-if)# exit
R2(config)#
```

Internal Router: Totally Stubby Area

R3 is an internal router within the totally stubby area, with both its G0/0 and G0/1 interfaces in area 51. An OSPF internal router is a router with all of its interfaces in the same area. An internal router in a totally stubby area is configured the same as an internal stub router, using the **area** *area-id* **stub** router mode command. The **no-summary** parameter is omitted. Only the ABR needs to be configured with the **no-summary** parameter in its **area** *area-id* **stub** interface command. Example 16-4 shows the configuration of router R3, an internal router within totally stubby area 51.

Example 16-4 *Configuring OSPFv3 on R3*

```
R3(config)# ipv6 unicast-routing
R3(config)# ipv6 router ospf 1
*Feb 16 23:05:42.483: %OSPFv3-4-NORTRID: Process OSPFv3-1-IPv6 could not pick a
  router-id, please configure manually
R3(config-rtr)# router-id 3.3.3.3
R3(config-rtr)# area 51 stub
R3(config-rtr)# exit
R3(config)# interface gigabitethernet 0/0
R3(config-if)# ipv6 ospf 1 area 51
R3(config-if)# exit
```

```
R3(config)# interface gigabitethernet 0/1
R3(config-if)# ipv6 ospf 1 area 51
*Feb 16 23:14:25.955: %OSPFv3-5-ADJCHG: Process 1, Nbr 2.2.2.2 on GigabitEthernet0/1
  from LOADING to FULL, Loading Done
R3(config-if)# exit
R3(config)#
```

Advertising a Default Route

An ASBR connects the OSPF domain to an external networks, including another routing domain, and sometimes to an Internet service provider. The ASBR provides the way out of the OSPF domain. Example 16-5 shows the configuration of the IPv6 static default route and OSPFv3 **default-information originate** command configured earlier on the ASBR, R1.

Example 16-5 *Advertising a Default Route Using OSPFv2*

```
R1(config)# ipv6 route ::/0 gigabitethernet0/2 fe80::4
R1(config)# ipv6 router ospf 1
R1(config-rtr)# router-id 1.1.1.1
R1(config-rtr)# passive-interface gigabitethernet 0/0
R1(config-rtr)# default-information originate
```

The complete syntax for the **default-information originate** command is as follows:

```
Router(config-rtr)# default-information originate [always | metric metric-value |
metric-type type-value | route-map map-name]
```

This command generates a default external route into the OSPFv3 routing domain. The parameters are as follows:

- **always (optional):** Always advertises the default route, regardless of whether there is a default route.
- **metric** *metric-value* **(optional):** Used for generating the default route. If you omit a value and do not specify a value using the default metric router configuration command, the default metric value is 1. The default metric value range is from 0 to 16777214.
- **metric-type** *type-value* **(optional):** External link type associated with the default route advertised into the OSPF for IPv6 routing domain. It can be one of the following values:
  - **1:** Type 1 external route
  - **2:** Type 2 external route

  The default is a type 2 external route. The cost of a type 2 route is always the external cost, regardless of the interior cost, the cost within the OSPF domain, to reach

that route. A type 1 cost is the sum of the external cost plus the internal cost used to reach that route.

- **route-map** *map-name* **(optional):** The routing process will generate the default route if the route map is satisfied.

Verifying Traditional OSPFv3

This section discusses the following commands, which are used to verify traditional OSPFv3:

- **show ipv6 route ospf**
- **show ipv6 ospf database**
- **show ipv6 protocols**
- **show ipv6 interface**
- **show ipv6 ospf neighbor**
- **show ipv6 ospf interface**
- **ping**
- **show running-config**

To verify that the OSPFv3 domain is fully converged, examine the IPv6 routing table on R1. The **show ipv6 route ospf** command in Example 16-6 shows the prefixes learned via OSPFv3 and the default route that is distributed by R1 into the OSPFv3 domain. Notice in the routing table entries that the administrative distance of 110 for an OSPFv3 route is the same as for OSPFv2.

Example 16-6 show ipv6 route ospf *Command on R2*

```
R2# show ipv6 route ospf
IPv6 Routing Table - default - 8 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity>
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, a - Application
OE2 ::/0 [110/1], tag 1
     via FE80::1, GigabitEthernet0/1
O   2001:DB8:CAFE:1::/64 [110/2]
     via FE80::1, GigabitEthernet0/1
O   2001:DB8:CAFE:4::/64 [110/2]
     via FE80::3, GigabitEthernet0/0
R2#
```

The **show ipv6 ospf database** command in Example 16-7 displays IPv6 LSA information from R2's OSPF LSDB. Highlighted in the example are the IPv6 LSAs displayed in

Table 16-2 and the Router ID associated with each LSA. For more information about OSPF LSAs and the **show ipv6 ospf database** command, see the references listed at the end of this chapter.

Example 16-7 **show ipv6 ospf database** *Command on R2*

```
R2# show ipv6 ospf database

            OSPFv3 Router with ID (2.2.2.2) (Process ID 1)

                Router Link States (Area 0)

ADV Router       Age         Seq#        Fragment ID  Link count  Bits
 1.1.1.1         315         0x80000002  0            1           E
 2.2.2.2         314         0x80000001  0            1           B

                Net Link States (Area 0)

ADV Router       Age         Seq#        Link ID    Rtr count
 1.1.1.1         315         0x80000001  4          2

                Inter Area Prefix Link States (Area 0)

ADV Router       Age         Seq#        Prefix
 2.2.2.2         305         0x80000002  2001:DB8:CAFE:3::/64
 2.2.2.2         219         0x80000001  2001:DB8:CAFE:4::/64
              Link (Type-8) Link States (Area 0)

ADV Router       Age         Seq#        Link ID    Interface
 1.1.1.1         422         0x80000001  4          Gi0/1
 2.2.2.2         315         0x80000001  4          Gi0/1

               Intra Area Prefix Link States (Area 0)

ADV Router       Age         Seq#        Link ID    Ref-lstype  Ref-LSID
 1.1.1.1         315         0x80000003  0          0x2001      0
 1.1.1.1         315         0x80000001  4096       0x2002      4

               Router Link States (Area 51)

ADV Router       Age         Seq#        Fragment ID  Link count  Bits
 2.2.2.2         224         0x80000003  0            1           B
 3.3.3.3         225         0x80000002  0            1           None

                Net Link States (Area 51)

ADV Router       Age         Seq#        Link ID    Rtr count
 2.2.2.2         224         0x80000001  3          2
```

```
                Inter Area Prefix Link States (Area 51)

ADV Router       Age          Seq#        Prefix
 2.2.2.2         315          0x80000001  ::/0

                Link (Type-8) Link States (Area 51)

ADV Router       Age          Seq#        Link ID    Interface
 2.2.2.2         346          0x80000001  3          Gi0/0
 3.3.3.3         225          0x80000001  4          Gi0/0

                Intra Area Prefix Link States (Area 51)

ADV Router       Age          Seq#        Link ID    Ref-lstype  Ref-LSID
 2.2.2.2         224          0x80000001  3072       0x2002      3
 3.3.3.3         220          0x80000003  0          0x2001      0

                Type-5 AS External Link States

ADV Router       Age          Seq#        Prefix
 1.1.1.1         437          0x80000001  ::/0
R2#
```

Examine the OSPFv3 routes in R3's routing table in Example 16-8 and notice that there is only one OSPFv3 route: a default route. This is not the default route propagated by the ASBR, router R1. This is the default route injected into the totally stubby area by the ABR, R2. An internal router within a totally stubby area has in its routing table only routes with prefixes within its own area and the default route injected by the ABR.

Example 16-8 **show ipv6 route ospf** *Command on R3*

```
R3# show ipv6 route ospf
IPv6 Routing Table - default - 6 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<output omitted for brevity>
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, a - Application
OI  ::/0 [110/2]
     via FE80::2, GigabitEthernet0/1
R3#
```

Example 16-9 shows the output from the **show ipv6 protocols** command on R2, the ABR. The output for the OSPFv3 routing process includes the following:

- OSPF process ID 1
- Router ID 2.2.2.2

- Indication that the router is an area border router with one normal area (area 0) and one stub area (area 51, a totally stubby area)
- Indication that interface G0/1 is in area 0 and interface G0/0 is in area 51

Example 16-9 **show ipv6 protocols** *Command on R2*

```
R2# show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "application"
IPv6 Routing Protocol is "ND"
IPv6 Routing Protocol is "ospf 1"
  Router ID 2.2.2.2
  Area border router
  Number of areas: 1 normal, 1 stub, 0 nssa
  Interfaces (Area 0):
    GigabitEthernet0/1
  Interfaces (Area 51):
    GigabitEthernet0/0
  Redistribution:
    None
R2#
```

Example 16-10 shows output from the **show ipv6 interface gigabitethernet 0/0** command on R3. This command verifies that R3's G0/0 interface is a member of ff02::5, the all SPF routers multicast group, and ff02::6, the all DR routers multicast group. R1 processes any OSPFv3 message with either of these destination addresses.

Example 16-10 **show ipv6 interface gigabitethernet 0/0** *Command on R3*

```
R3# show ipv6 interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::3
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:4::1, subnet is 2001:DB8:CAFE:4::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::5
    FF02::6
    FF02::FB
    FF02::1:FF00:1
    FF02::1:FF00:3
<output omitted for brevity>
R3#
```

In Example 16-11, the **show ipv6 ospf neighbor** command lists the OSPFv3 neighbor adjacencies for R2. Notice that the neighbor ID of each neighbor is the other neighbor's 32-bit OSPFv3 Router ID.

Example 16-11 **show ipv6 ospf neighbor** *Command on R2*

```
R2# show ipv6 ospf neighbor

            OSPFv3 Router with ID (2.2.2.2) (Process ID 1)

Neighbor ID     Pri   State           Dead Time   Interface ID    Interface
1.1.1.1           1   FULL/DR         00:00:33    4               GigabitEthernet0/1
3.3.3.3           1   FULL/BDR        00:00:30    4               GigabitEthernet0/0
R2#
```

Example 16-12 shows output from R2's **show ipv6 ospf interface gigabitethernet 0/0** command, which is similar to output from the equivalent command in OSPFv2. The difference is that OSPFv3 sources packets from the interface's link-local address, which is fe80::2 in the case of R2. This also applies to the local addresses of the DR and BDR. Manually configuring link-local addresses makes it easier to recognize the source. In OSPFv2, these packets are sourced from the interface's IPv4 address.

Example 16-12 **show ipv6 ospf interface gigabitethernet 0/0** *Command on R2*

```
R2# show ipv6 ospf interface gigabitethernet 0/0
GigabitEthernet0/0 is up, line protocol is up
  Link Local Address FE80::2, Interface ID 3
  Area 51, Process ID 1, Instance ID 0, Router ID 2.2.2.2
  Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 2.2.2.2, local address FE80::2
  Backup Designated router (ID) 3.3.3.3, local address FE80::3
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:00
  Graceful restart helper support enabled
  Index 1/1/2, flood queue length 0
  Next 0x0(0)/0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 4
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 3.3.3.3  (Backup Designated Router)
  Suppress hello for 0 neighbor(s)
R2#
```

Example 16-13 shows the **ping** command used to verify reachability. The pings from R3 to the interfaces on R2, R1, and ISP routers are all successful.

Example 16-13 *Using* **ping** *to Verify Reachability*

```
R3# ping 2001:db8:cafe:2::2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:2::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/8 ms
R3# ping 2001:db8:cafe:1::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:1::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R3# ping 2001:db8:feed:6::2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:FEED:6::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
R3#
```

Examining the running configs on each router provides a helpful review of the commands used in this chapter to configure traditional OSPFv3. Example 16-14 shows selected output from the running configs of each of the three routers. The commands used in this section are highlighted for each router.

Example 16-14 **show running-config** *Command on R1, R2, and R3*

```
R1# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface GigabitEthernet0/0
 no ip address
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:1::1/64
 ipv6 ospf 1 area 0
!
interface GigabitEthernet0/1
 no ip address
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:2::1/64
 ipv6 ospf 1 area 0
```

```
!
interface GigabitEthernet0/2
 no ip address
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:FEED:6::1/64
!
ipv6 route ::/0 GigabitEthernet0/2 FE80::4
ipv6 router ospf 1
 router-id 1.1.1.1
 default-information originate
 passive-interface GigabitEthernet0/0
!
R1#
------------------------------------------------------------
R2# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface GigabitEthernet0/0
 no ip address
 ipv6 address FE80::2 link-local
 ipv6 address 2001:DB8:CAFE:3::1/64
 ipv6 ospf 1 area 51
!
interface GigabitEthernet0/1
 no ip address
 ipv6 address FE80::2 link-local
 ipv6 address 2001:DB8:CAFE:2::2/64
 ipv6 ospf 1 area 0
!
ipv6 router ospf 1
 router-id 2.2.2.2
 area 51 stub no-summary
!
R2#
------------------------------------------------------------
R3# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
```

```
!
interface GigabitEthernet0/0
 no ip address
 ipv6 address FE80::3 link-local
 ipv6 address 2001:DB8:CAFE:4::1/64
 ipv6 ospf 1 area 51
!
interface GigabitEthernet0/1
 no ip address
 ipv6 address FE80::3 link-local
 ipv6 address 2001:DB8:CAFE:3::2/64
 ipv6 ospf 1 area 51
!
ipv6 router ospf 1
 router-id 3.3.3.3
 area 51 stub
!
R3#
```

OSPFv3 with Address Families

OSPFv2 supports only IPv4, whereas the original implementation of OSPFv3 (traditional OSFPv3) supports only IPv6. With the introduction of OSPFv3 support for address families (AF), OSPFv3 can now support both IPv4 and IPv6 address families. Conceptually, OSPFv3 address families are similar to address families used by Routing Information Protocol (RIP), Intermediate System–to–Intermediate System (IS-IS), Enhanced Interior Gateway Routing Protocol (EIGRP), and Border Gateway Protocol (BGP).

A significant difference from traditional OSPFv3 is that OSPFv3 with AF uses the same neighbor table and LSDB for routing both IPv4 and IPv6, as shown in Figure 16-3.

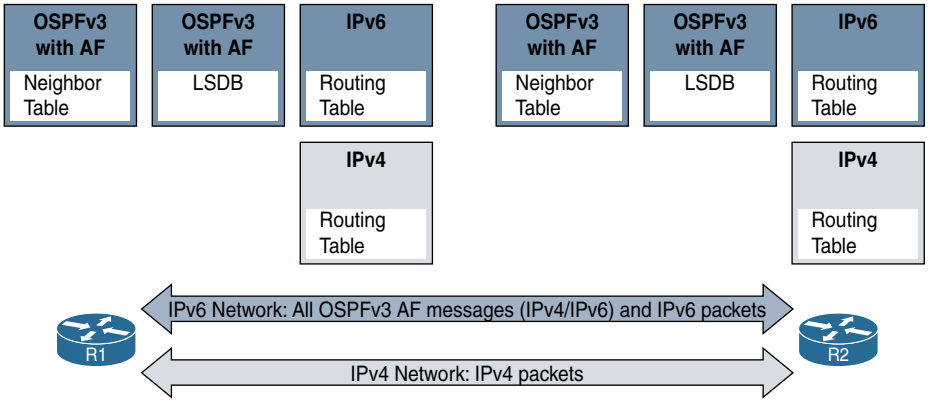


Figure 16-3 *OSPFv3 with AF Uses the Same Neighbor Table and LSDB for IPv4 and IPv6*

OSPFv3 with AF uses IPv6 for transporting OSPF messages for both the IPv4 and IPv6 address families. Therefore, OSPFv3 with AF requires router interfaces to be configured with an IPv6 link-local address and the **ipv6-unicast routing** command even when routing for the IPv4 AF only.

Similar to EIGRP named mode in Chapter 15, OSPFv3 with AF configures both IPv4 and IPv6 address families in one place.

Figure 16-4 shows the topology used in this section to configure OSPFv3 with AF. This topology includes IPv4 and IPv6 prefixes.

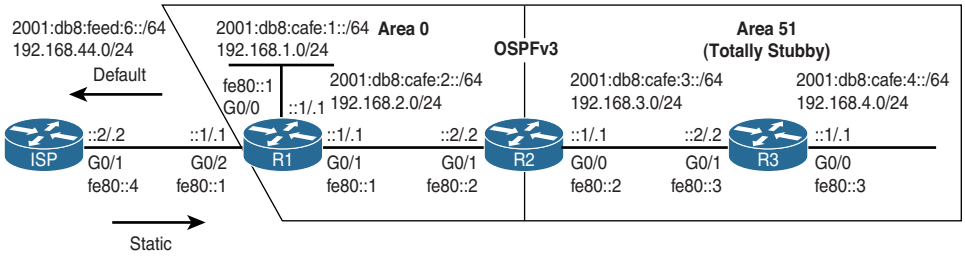


Figure 16-4 *Topology for OSPFv3 with AF*

Note IPv6 addressing on R2's interfaces could be configured with link-local addresses only, no global unicast addresses. This is because R2 has no end user interfaces. RFC 7404, Using Only Link-Local Addressing inside an IPv6 Network, discusses implementing routing protocols using only link-local addresses on infrastructure links.

Configuring OSPFv3 with AF

This section shows how to configure OSPFv3 address families for both IPv4 and IPv6. Configuring both address families will help you better understand the shared neighbor table, LSDB, and the commands for verifying OSPFv3 that are discussed in the next section.

In this section, routers R1, R2, and R3 are configured to share routing information within the routing domain using OSPFv3 with AF. The ISP router is configured with an IPv4 and an IPv6 static route to reach the OSPFv3 domain, as shown in Example 16-15.

Example 16-15 *Configuring Static Routes on ISP*

```
ISP(config)# ip route 192.168.4.0 255.255.248.0 192.168.44.1
ISP(config)# ipv6 route 2001:db8:cafe::/48 g0/1 fe80::1
```

ASBR and Advertising a Default Route

Example 16-16 shows the OSPFv3 with AF configuration on router R1. R1 is an ASBR, which connects its OSPFv3 domain to the upstream ISP router. The example begins with

enabling R1 as an IPv6 router, using the **ipv6 unicast-routing** command. This command is required to configure OSPFv3 for both the IPv6 and IPv4 address families. R1 is also configured with an IPv6 and an IPv4 default static route via ISP.

Example 16-16 *Configuring OSPFv3 IPv6 and IPv4 AFs on R1*

```
R1(config)# ipv6 unicast-routing
R1(config)# ipv6 route ::/0 gigabitethernet0/2 192.168.44.2
R1(config)# ip route 0.0.0.0 0.0.0.0 gigabitethernet0/2 fe80::4
R1(config)# router ospfv3 1
R1(config-router)# address-family ?
  ipv4  Address family
  ipv6  Address family

R1(config-router)# address-family ipv6 unicast
R1(config-router-af)# ?
Router Address Family configuration commands:
  area                    OSPF area parameters
  auto-cost               Calculate OSPF interface cost according to bandwidth
  compatible              Compatibility list
  default                 Set a command to its defaults
  default-information     Distribution of default information
  default-metric          Set metric of redistributed routes
  discard-route           Enable or disable discard-route installation
  distance                Administrative distance
  distribute-list         Filter networks in routing updates
  event-log               Event Logging
  exit-address-family     Exit from Address Family configuration mode
  graceful-restart        Graceful-restart options
  help                    Description of the interactive help system
  interface-id            Source of the interface ID
  limit                   Limit a specific OSPF feature
  log-adjacency-changes   Log changes in adjacency state
  max-lsa                 Maximum number of non self-generated LSAs to accept
  max-metric              Set maximum metric
  maximum-paths           Forward packets over multiple paths
  no                      Negate a command or set its defaults
  passive-interface       Suppress routing updates on an interface
  prefix-suppression      Enable prefix suppression
  queue-depth             Hello/Router process queue depth
  redistribute            Redistribute IPv6 prefixes from another routing
                          protocol
  router-id               router-id for this OSPF process
  shutdown                Shutdown the router process
  snmp                    Modify snmp parameters
  summary-prefix          Configure IPv6 summary prefix
```

```
  table-map              Map external entry attributes into routing table
  timers                 Adjust routing timers
R1(config-router-af)# router-id 1.1.1.6
R1(config-router-af)# passive-interface gigabitethernet 0/0
R1(config-router-af)# default-information originate
R1(config-router-af)# exit-address-family
R1(config-router)# address-family ipv4 unicast
R1(config-router-af)# router-id 1.1.1.4
R1(config-router-af)# passive-interface gigabitethernet 0/0
R1(config-router-af)# default-information originate
R1(config-router-af)# exit-address-family
R1(config-router)# exit
R1(config)#
```

OSPFv3 with AF has two configuration modes:

- **Router configuration mode (config-router):** Used to determine which address family to configure
- **Address-family configuration mode (config-router-af):** Used to configure OSPFv3 for a specific AF, either IPv4 or IPv6

The following commands are used in Example 16-16 to configure OSPFv3 with AF on R1:

- R1(config)**# router ospfv3 1**

  The **router ospfv3** *process-id* command enters you into router configuration mode. The process ID is locally significant and does not have to match the process ID on other routers.

- R1(config-router)**# address-family ?**

  By using the help feature (**?**), you can see that there are two address families available: IPv4 and IPv6.

- R1(config-router)**# address-family ipv6 unicast**

  This command enters you into IPv6 address-family configuration mode for OSPFv3. If there are no loopback or physical interfaces with an IPv4 address, you receive the following IOS message, indicating that a Router ID needs to be configured manually:

  ```
  *Feb 17 17:01:22.887: %OSPFv3-4-NORTRID: Process OSPFv3-1-IPv6 could not pick
  a router-id, please configure manually
  ```

- R1(config-router-af)**# ?**

  The help command (**?**) in address-family configuration mode displays the commands available in this mode, such as **area**, **default-information**, **exit-address-family**, **passive-interface**, and **router-id**, which are examined later in this section.

- R1(config-router-af)**# router-id 1.1.1.6**

  The **router-id 1.1.1.6** address-family configuration mode command on R1 configures the 32-bit Router ID for the IPv6 address family. The same or a different Router ID can be used for the IPv4 AF.

Note In order to help differentiate between IPv6 and IPv4 address families, Router IDs in the IPv6 AF end in a 6, whereas Router IDs in the IPv4 AF end in a 4.

- R1(config-router-af)**# passive-interface gigabitethernet 0/0**

  The **passive-interface** *interface-type interface-number* command is used in this mode to suppress OSPF hello messages and other OSPF messages on the interface. R1's G0/0 interface doesn't have any neighbors, so this command is used to make the interface passive.

- R1(config-router-af)**# default-information originate**

  The **default-information originate** command advertises the default static route into the routing domain. The same options are available as discussed earlier, for traditional OSPFv3.

- R1(config-router-af)**# exit-address-family**

  The **exit-address-family** command (or the shorter form, **exit)** exits the IPv6 address-family configuration mode and returns you to router configuration mode (config-router).

Example 16-16 shows similar OSPFv3 commands used to configure the IPv4 address family on R1. Although it is not required, a different value is used for Router ID in the IPv4 AF than is used for the IPv6 AF.

OSPFv3 is enabled directly on the interfaces for both IPv4 and IPv6 AFs using the **ospfv3** *process-id* [**ipv4** | **ipv6**] **area** *area-id* interface command, as shown in Example 16-17.

Example 16-17 *Enabling OSPFv3 with AF Directly on the Interfaces*

```
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ospfv3 1 ?
  cost                  Route cost of this interface
  database-filter       Filter OSPF LSA during synchronization and flooding
  dead-interval         Interval after which a neighbor is declared dead
  demand-circuit        OSPF demand circuit
  flood-reduction       OSPF Flood Reduction
  hello-interval        Time between HELLO packets
```

```
  ipv4                  Specify parameters for IPv4
  ipv6                  Specify parameters for IPv6
  mtu-ignore            Ignores the MTU in DBD packets
  neighbor              OSPF neighbor
  network               Network type
  prefix-suppression    OSPF prefix suppression
  priority              Router priority
  retransmit-interval   Time between retransmitting lost link state
                        advertisements
  shutdown              Shut down the interface in OSPFv3
  transmit-delay        Link state transmit delay

R1(config-if)# ospfv3 1 ipv6 area 0
R1(config-if)# ospfv3 1 ipv4 area 0
R1(config-if)# exit
R1(config)# interface gigabitethernet 0/1
R1(config-if)# ospfv3 1 ipv6 area 0
R1(config-if)# ospfv3 1 ipv4 area 0
R1(config-if)# exit
R1(config)#
```

ABR with Totally Stubby Area

Example 16-18 shows the OSPFv3 with AF configuration on R2. The example shows the complete configuration without any help commands (**?**), which makes the configuration a little easier to visualize.

R2 is an ABR with G0/1 interface in area 0 and G0/0 interface in area 51. Area 51 is a totally stubby area and R2 injects only a single, default route into area 51. R2 does not advertise any prefixes from other areas or external prefixes into the totally stubby area. For both address families, the **area 51 stub no-summary** command is used in router configuration mode, to make area 51 a totally stubby area. (Omitting the **no-summary** option makes the area a stub area.)

Example 16-18 *Configuring OSPFv3 IPv6 and IPv4 AFs on R2*

```
R2(config)# ipv6 unicast-routing
R2(config)# router ospfv3 1
R2(config-router)# address-family ipv6 unicast
R2(config-router-af)# router-id 2.2.2.6
R2(config-router-af)# area 51 stub no-summary
R2(config-router-af)# exit-address-family
R2(config-router)# address-family ipv4 unicast
```

```
R2(config-router-af)# router-id 2.2.2.4
R2(config-router-af)# area 51 stub no-summary
R2(config-router-af)# exit-address-family
R2(config-router)# exit
R2(config)# interface gigabitethernet 0/1
R2(config-if)# ospfv3 1 ipv6 area 0
R2(config-if)# ospfv3 1 ipv4 area 0
R2(config-if)# exit
R2(config)# interface gigabitethernet 0/0
R2(config-if)# ospfv3 1 ipv6 area 51
R2(config-if)# ospfv3 1 ipv4 area 51
R2(config-if)# exit
R2(config)#
```

Internal Router: Totally Stubby Area

R3 is an internal router within the totally stubby area, with both of its interfaces in area 51. Example 16-19 shows the OSPFv3 with AF configuration for both the IPv4 and IPv6 address families on router R3. The **area 51 stub** command, excluding the **no-summary** parameter, is used for both AFs because R3 is an internal router within a totally stubby area. With no neighbors on its G0/0 interface, G0/0 is configured as a passive interface.

Example 16-19 *Configuring OSPFv3 IPv6 and IPv4 AFs on R3*

```
R3(config)# ipv6 unicast-routing
R3(config)# router ospfv3 1
R3(config-router)# address-family ipv6 unicast
R3(config-router-af)# router-id 3.3.3.6
R3(config-router-af)# area 51 stub
R3(config-router-af)# passive-interface gigabitethernet 0/0
R3(config-router-af)# exit-address-family
R3(config-router)# address-family ipv4 unicast
R3(config-router-af)# router-id 3.3.3.4
R3(config-router-af)# area 51 stub
R3(config-router-af)# passive-interface gigabitethernet 0/0
R3(config-router-af)# exit-address-family
R3(config-router)# exit
R3(config)# interface gigabitethernet 0/0
R3(config-if)# ospfv3 1 ipv6 area 51
R3(config-if)# ospfv3 1 ipv4 area 51
R3(config-if)# exit
R3(config)# interface gigabitethernet 0/1
R3(config-if)# ospfv3 1 ipv6 area 51
R3(config-if)# ospfv3 1 ipv4 area 51
R3(config-if)# exit
R3(config)#
```

Verifying OSPFv3 with AF

This section shows how to verify and display OSPFv3 information for both the IPv4 and IPv6 address families. The "**show** commands" used with OSPFv3 with AF can be somewhat confusing at first. This is because the **show** commands used with traditional OSPFv3 work for OSPFv3 with AF but display only IPv6 AF information. Showing information for both the IPv4 and IPv6 address families requires a different set of **show** commands. To understand why this is the case, let's take a look at how OSPF is configured.

Different OSPF configuration methods advertise prefixes for IPv4 or IPv6 or both:

- **OSPFv2:** Advertises only IPv4 prefixes
- **Traditional OSPFv3:** Advertises only IPv6 prefixes
- **OSPFv3 with AF:** Advertises both IPv4 prefixes and IPv6 prefixes

Each of the three OSPF configuration methods requires a different set of OSPF verification commands. If OSPFv3 with AF was used to configure IPv4 prefixes, this requires a different set of verification commands than if OSPFv2 was used.

Before exploring commands specific to verifying OSPF, let's examine the routing tables. IOS maintains separate routing tables for IPv4 and IPv6, as shown in Figure 16-3. Therefore, you need different commands to show each table.

In Example 16-20, the **show ip route ospf** command is used, but there isn't any output. You might be expecting the IPv4 prefixes learned via OSPF to be displayed. Although the **ip** parameter refers to IPv4, the **ospf** parameter refers to OSPFv2—not OSPFv3. This command, with the **ospf** parameter, is used to display IPv4 routes learned via OSPFv2. OSPFv3 had been used to configure the IPv4 address family.

The **show ip route ospfv3** command is used to display IPv4 prefixes learned from OSPFv3. The **ip** parameter refers to IPv4, and the **ospfv3** parameter refers to routes learned via OSPFv3.

Example 16-20 *Displaying OSPFv3 Routing Table Entries on R2*

```
R2# show ip route ospf
<no output>
R2# show ip route ospfv3
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
<Output omitted for brevity>

Gateway of last resort is 192.168.2.1 to network 0.0.0.0

O*E2  0.0.0.0/0 [110/1] via 192.168.2.1, 00:00:37, GigabitEthernet0/1
O     192.168.1.0/24 [110/2] via 192.168.2.1, 02:16:34, GigabitEthernet0/1
O     192.168.4.0/24 [110/2] via 192.168.3.2, 02:13:26, GigabitEthernet0/0
```

```
R2#
R2# show ipv6 route ospfv3
Translating "ospfv3"
                      ^
% Invalid input detected at '^' marker.

R2# show ipv6 route ospf
IPv6 Routing Table - default - 8 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
<Output omitted for brevity>
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, a - Application
OE2 ::/0 [110/1], tag 1
     via FE80::1, GigabitEthernet0/1
O   2001:DB8:CAFE:1::/64 [110/2]
     via FE80::1, GigabitEthernet0/1
O   2001:DB8:CAFE:4::/64 [110/2]
     via FE80::3, GigabitEthernet0/0
R2#
```

Continuing with the commands in Example 16-20, you might expect that the **show ipv6 route ospfv3** command would be used to show the IPv6 prefixes learned via OSPFv3. However, the actual command is the same as for traditional OSPFv3, the **show ipv6 route ospf** command.

Example 16-21 demonstrates the differences in OSPF verification commands for the three OSPF configuration methods. The first command is **show ip ospf neighbor**, which results in no output. This is because **ip ospf** refers to OSPF for IPv4 (OSPFv2), not OSPFv3.

The next command is **show ipv6 ospf neighbor**. This command provides information but only for OSPF for IPv6, not IPv4. (Notice that all of the Router IDs end in a 6.) The **ipv6 ospf** parameter refers only to OSPF for IPv6, either traditional OSPFv3 or OSPFv3 with AF.

The last command, **show ospfv3 neighbors**, displays OSPFv3 information for both IPv4 and IPv6. This is the command you use to see OSPF information for both IPv4 and IPv6. Notice that OSPF information for both address families is displayed. As illustrated in Figure 16-3, OSPFv3 with AF uses the same neighbor table for both address families.

Example 16-21 *Displaying OSPFv3 Neighbor Table Information on R2*

```
R2# show ip ospf neighbor
<no output>
R2# show ipv6 ospf neighbor
            OSPFv3 Router with ID (2.2.2.6) (Process ID 1)
Neighbor ID     Pri   State           Dead Time   Interface ID    Interface
1.1.1.6           1   FULL/BDR        00:00:37    4               GigabitEthernet0/1
3.3.3.6           1   FULL/DR         00:00:35    4               GigabitEthernet0/0
R2#
R2# show ospfv3 neighbor

          OSPFv3 1 address-family ipv4 (router-id 2.2.2.4)
Neighbor ID     Pri   State           Dead Time   Interface ID    Interface
1.1.1.4           1   FULL/BDR        00:00:37    4               GigabitEthernet0/1
3.3.3.4           1   FULL/DR         00:00:36    4               GigabitEthernet0/0
          OSPFv3 1 address-family ipv6 (router-id 2.2.2.6)
Neighbor ID     Pri   State           Dead Time   Interface ID    Interface
1.1.1.6           1   FULL/BDR        00:00:36    4               GigabitEthernet0/1
3.3.3.6           1   FULL/DR         00:00:34    4               GigabitEthernet0/0
R2#
```

The three groups of **show** command parameters are summarized as follows:

- **ip ospf:** Refers to IPv4 information learned via OSPFv2
- **ipv6 ospf:** Refers to IPv6 information learned via traditional OSPFv3 and OSPFv3 with AF
- **ospfv3:** Refers to IPv4 and IPv6 information learned via OSPFv3 with AF

This should help you understand why each configuration method uses different parameters and why you see (or don't see) the information you do.

Note These are the same parameters used to enable an interface for OSPFv2, traditional OSPFv3, and OSPFv3 with AF.

Example 16-22 shows another example that uses these three parameters to display a summary of the OSPF link-state database. The first command, **show ip ospf database**, results in no output. Again, this is because the **ip ospf** parameter is used for OSPFv2 only, not OSPFv3.

Next, the **show ipv6 ospf database** command provides information, but the **ipv6 ospf** parameter displays only the IPv6 LSAs in the LSDB and none for the IPv4 address family.

The last command in Example 16-22, **show ospfv3 database**, is the command necessary for displaying the OSPFv3 LSDB summary information for both IPv4 and IPv6. Notice the Router IDs for both the IPv4 and IPv6 address families. As illustrated in Figure 16-3, OSPFv3 with AFs uses the same LSDB for both address families.

Example 16-22 *Displaying OSPFv3 LSDB Summary Information on R3*

```
R3# show ip ospf database
<no output>

R3# show ipv6 ospf database

            OSPFv3 Router with ID (3.3.3.6) (Process ID 1)

                Router Link States (Area 51)

ADV Router       Age         Seq#        Fragment ID  Link count  Bits
 2.2.2.6         1876        0x8000014D  0            1           B
 3.3.3.6         862         0x8000014E  0            1           None

                Net Link States (Area 51)

ADV Router       Age         Seq#        Link ID    Rtr count
 3.3.3.6         862         0x8000014C  4          2

                Inter Area Prefix Link States (Area 51)

ADV Router       Age         Seq#        Prefix
 2.2.2.6         1876        0x8000014B  ::/0

                Link (Type-8) Link States (Area 51)

ADV Router       Age         Seq#        Link ID    Interface
 2.2.2.6         1876        0x8000014C  3          Gi0/1
 3.3.3.6         862         0x8000014D  4          Gi0/1
 3.3.3.6         862         0x8000014D  3          Gi0/0

                Intra Area Prefix Link States (Area 51)

ADV Router       Age         Seq#        Link ID    Ref-lstype  Ref-LSID
 3.3.3.6         862         0x80000150  0          0x2001      0
 3.3.3.6         862         0x8000014C  4096       0x2002      4
R3#
R3# show ospfv3 database

          OSPFv3 1 address-family ipv4 (router-id 3.3.3.4)

                Router Link States (Area 51)

ADV Router       Age         Seq#        Fragment ID  Link count  Bits
 2.2.2.4         428         0x8000014E  0            1           B
```

```
3.3.3.4         1700        0x8000014D  0           1           None

               Net Link States (Area 51)

ADV Router      Age         Seq#        Link ID    Rtr count
3.3.3.4         1700        0x8000014B  4          2

               Inter Area Prefix Link States (Area 51)

ADV Router      Age         Seq#        Prefix
2.2.2.4         428         0x8000014C  0.0.0.0/0

               Link (Type-8) Link States (Area 51)

ADV Router      Age         Seq#        Link ID    Interface
2.2.2.4         428         0x8000014D  3          Gi0/1
3.3.3.4         1700        0x8000014B  4          Gi0/1
3.3.3.4         1700        0x8000014C  3          Gi0/0

               Intra Area Prefix Link States (Area 51)

ADV Router      Age         Seq#        Link ID    Ref-lstype  Ref-LSID
3.3.3.4         1700        0x8000014F  0          0x2001      0
3.3.3.4         1700        0x8000014B  4096       0x2002      4

          OSPFv3 1 address-family ipv6 (router-id 3.3.3.6)

               Router Link States (Area 51)

ADV Router      Age         Seq#        Fragment ID  Link count  Bits
2.2.2.6         373         0x8000014E  0            1           B
3.3.3.6         1384        0x8000014E  0            1           None

               Net Link States (Area 51)

ADV Router      Age         Seq#        Link ID    Rtr count
3.3.3.6         1384        0x8000014C  4          2

               Inter Area Prefix Link States (Area 51)

ADV Router      Age         Seq#        Prefix
2.2.2.6         373         0x8000014C  ::/0
```

```
                Link (Type-8) Link States (Area 51)

ADV Router      Age         Seq#        Link ID    Interface
 2.2.2.6        373         0x8000014D  3          Gi0/1
 3.3.3.6        1384        0x8000014D  4          Gi0/1
 3.3.3.6        1384        0x8000014D  3          Gi0/0

               Intra Area Prefix Link States (Area 51)

ADV Router      Age         Seq#        Link ID    Ref-lstype  Ref-LSID
 3.3.3.6        1384        0x80000150  0          0x2001      0
 3.3.3.6        1384        0x8000014C  4096       0x2002      4
R3#
```

The final verification involves examining the running configs on each router. Example 16-23 shows selected output from the running configs on routers R1, R2, and R3. Some of the commands are highlighted to help you visualize the configurations.

Example 16-23 **show running-config** *for R1, R2, and R3*

```
R1# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface GigabitEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:1::1/64
 ospfv3 1 ipv4 area 0
 ospfv3 1 ipv6 area 0
!
interface GigabitEthernet0/1
 ip address 192.168.2.1 255.255.255.0
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:2::1/64
 ospfv3 1 ipv4 area 0
 ospfv3 1 ipv6 area 0
```

```
!
interface GigabitEthernet0/2
 ip address 192.168.44.1 255.255.255.0
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:FEED:6::1/64
!
router ospfv3 1
 !
 address-family ipv4 unicast
  passive-interface GigabitEthernet0/0
  default-information originate
  router-id 1.1.1.4
 exit-address-family
 !
 address-family ipv6 unicast
  passive-interface GigabitEthernet0/0
  default-information originate
  router-id 1.1.1.6
 exit-address-family
!
ip route 0.0.0.0 0.0.0.0 GigabitEthernet0/2 192.168.44.2
!
ipv6 route ::/0 GigabitEthernet0/2 FE80::4
!
R1#
-------------------------------------------------------
R2# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface GigabitEthernet0/0
 ip address 192.168.3.1 255.255.255.0
 ipv6 address FE80::2 link-local
 ipv6 address 2001:DB8:CAFE:3::1/64
 ospfv3 1 ipv4 area 51
 ospfv3 1 ipv6 area 51
!
interface GigabitEthernet0/1
 ip address 192.168.2.2 255.255.255.0
 ipv6 address FE80::2 link-local
 ipv6 address 2001:DB8:CAFE:2::2/64
```

```
 ospfv3 1 ipv4 area 0
 ospfv3 1 ipv6 area 0
!
router ospfv3 1
 !
 address-family ipv4 unicast
  router-id 2.2.2.4
  area 51 stub no-summary
 exit-address-family
 !
 address-family ipv6 unicast
  router-id 2.2.2.6
  area 51 stub no-summary
 exit-address-family
!
R2#
------------------------------------------------------------
R3# show running-config
!
ip cef
ipv6 unicast-routing
ipv6 cef
!
interface GigabitEthernet0/0
 ip address 192.168.4.1 255.255.255.0
 ipv6 address FE80::3 link-local
 ipv6 address 2001:DB8:CAFE:4::1/64
 ospfv3 1 ipv4 area 51
 ospfv3 1 ipv6 area 51
!
interface GigabitEthernet0/1
 ip address 192.168.3.2 255.255.255.0
 ipv6 address FE80::3 link-local
 ipv6 address 2001:DB8:CAFE:3::2/64
 ospfv3 1 ipv4 area 51
 ospfv3 1 ipv6 area 51
!
router ospfv3 1
 !
 address-family ipv4 unicast
  passive-interface GigabitEthernet0/0
  router-id 3.3.3.4
  area 51 stub
 exit-address-family !
```

```
 address-family ipv6 unicast
  passive-interface GigabitEthernet0/0
  router-id 3.3.3.6
  area 51 stub
 exit-address-family
!
R3#
```

Configuring OSPFv3 for an IPv4 Island

A router that needs to advertise only IPv4 prefixes is used here in order to illustrate that OSPF messages are sent over IPv6 for both address families. Router RZ in Figure 16-5 is an IPv4-only "island," which means it needs to route only IPv4 prefixes. Any end-user devices attached to any of its networks are IPv4-only as well. RZ only needs to learn about remote IPv4 prefixes and only needs to advertise its own IPv4 prefixes.

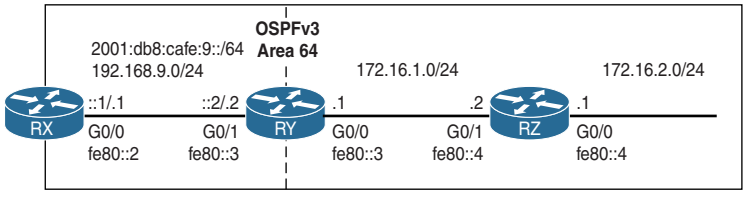


Figure 16-5 *OSPFv3 Topology with an IPv4 Island*

OSPFv3 with AF uses IPv6 for sending all OSPF messages, whether they are for the IPv4 address family or the IPv6 address family. Although RZ only needs to route for IPv4, and its end devices are IPv4-only, its interfaces need to have both IPv4 addresses and IPv6 link-local addresses.

Example 16-24 shows the configuration for router RZ. Notice that RZ must be enabled as an IPv6 router before you can configure OSPFv3 with AF. Using OSPFv3 router configuration mode, only the IPv4 address family is configured on RZ.

In Example 16-24, in the first attempt to configure OSPFv3 on the G0/1 interface, the **ospfv3 1 ipv4 area 64** command is rejected because IPv6 is not enabled on the interface. An IPv6 link-local address is required to be configured on the interface prior to using this command. The same commands are configured on RZ's G0/0 interface.

Example 16-24 *Configuring OSPFv3 with the IPv4 Address Family on RZ*

```
RZ(config)# router ospfv3 1
%OSPFv3: IPv6 routing not enabled
RZ(config)# ipv6 unicast-routing
RZ(config)# router ospfv3 1
RZ(config-router)# address-family ipv4 unicast
RZ(config-router-af)# router-id 4.4.4.4
RZ(config-router-af)# exit-address-family
RZ(config-router)# exit
RZ(config)# interface gigabitethernet 0/1
RZ(config-if)# ip address 172.16.1.2 255.255.255.0
RZ(config-if)# ospfv3 1 ipv4 area 64
% OSPFv3: IPV6 is not enabled on this interface
RZ(config-if)# ipv6 address fe80::4 link-local
RZ(config-if)# ospfv3 1 ipv4 area 64
RZ(config-if)# no shutdown
RZ(config-if)# exit
RZ(config)# interface gigabitethernet 0/0
RZ(config-if)# ip address 172.16.2.1 255.255.255.0
RZ(config-if)# ipv6 address fe80::4 link-local
RZ(config-if)# ospfv3 1 ipv4 area 64
RZ(config-if)# no shutdown
RZ(config-if)#
```

Note The G0/0 interface on router RY has a similar configuration to RZ's G0/0 interface. RY's G0/0 interface also requires an IPv4 address and an IPv6 link-local address.

If RY's G0/0 interface is already configured, RZ forms a full OSPF adjacency on its G0/1 interface with RY. All OSPFv3 messages, including hello messages and LSAs, are encapsulated in IPv6. For example, OSPF hello messages from RZ are sent to ff02::5, the AllSPFRouters multicast address, and sourced from its IPv6 link-local address, fe80::4.

All other packets between RZ and RY are sent using IPv4, such as any packets sent from devices on RZ's LAN and forwarded by RZ. Example 16-25 shows RZ's IPv4 routing table, using the **show ip route ospfv3** command.

Example 16-25 **show ip route ospfv3** *on RZ*

```
RZ# show ip route ospfv3
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF <output omitted for
           brevity>

O     192.168.9.0/24 [110/2] via 172.16.1.1, 03:04:09, GigabitEthernet0/1
RZ#
```

Summary

This chapter discusses the similarities and differences between OSPFv2, traditional OSPFv3, and OSPFv3 with AF. It shows configuration and verification for both traditional OSPFv3 and OSPFv3 with AF.

Many of the same processes and operational functionality and the same link-state concepts and hierarchical design apply to both OSPFv2 and OSPFv3. The following apply to OSPFv2, traditional OSPFv3, and OSPFv3 with AF:

- They are link-state routing protocols.
- They use cost as metric, with Cisco IOS interpreting cost as bandwidth.
- They use a 32-bit OSPF Router ID.
- They include support for multiple areas.

However, there are also some differences, including the following:

- OSPFv2 advertises IPv4 prefixes, whereas traditional OSPFv3 advertises IPv6 prefixes, and OSPFv3 with AF advertises both IPv4 and IPv6 prefixes.
- OSPFv2 messages are sent from the IPv4 address of the interface, whereas OSPFv3 messages are sent from the IPv6 link-local address of the interface.
- OSPFv2 sends updates using the 224.0.0.5 all SPF routers and 224.0.0.6 all DR routers multicast addresses, whereas OSPFv3 sends updates using the ff002::5 all SPF routers and ff02::6 all DR routers multicast addresses.

Configuring traditional OSPFv3 is similar to configuring OSPFv2. The following commands are the traditional OSPFv2 commands used in this chapter:

- R1(config)# **ipv6 router ospf** *process-id*

  This command creates and enables the OSPFv3 routing process. The process ID is locally significant and does not have to match other routers in the OSPFv3 routing domain.

- Router(config-rtr)# **router-id** *router-id*

  This command configures the OSPFv3 Router ID. This command is required if there are no loopback or active physical interfaces with an IPv4 address.

- Router(config-rtr)# **passive-interface** *interface-type interface-number*

  This command suppresses OSPF hello messages and other OSPF messages from being sent out an interface.

- Router(config-rtr)# **default-information originate**

  This command distributes the default route on to other routers in the OSPFv3 domain.

- Router(config-rtr)# **area** *area-id* **stub** [**no-summary**]

  This command defines the area as a stub area. The **no-summary** parameter on the ABR makes the area a totally stubby area.

- Router(config-if)# **ipv6 ospf** *process-id* **area** *area-id*

  This interface command enables OSPFv3 directly on the interface. *process-id* identifies the specific routing process and must be the same as the process ID used to create the routing process in the **ipv6 router ospf** *process-id* command.

OSPFv2 supports only IPv4, while the original implementation of OSPFv3 (traditional OSFPv3) supports only IPv6. With the introduction of OSPFv3 with AF, OSPFv3 supports both IPv4 and IPv6 address families.

Different OSPF configuration methods advertise prefixes for IPv4 or IPv6 or both:

- **OSPFv2:** Advertises only IPv4 prefixes
- **Traditional OSPFv3:** Advertises only IPv6 prefixes
- **OSPFv3 with AF:** Advertises both IPv4 prefixes and IPv6 prefixes

OSPFv3 with AF has two configuration modes:

- **Router configuration mode (config-router):** Used to determine which address family to configure
- **Address-family configuration mode (config-router-af):** Used to configure OSPFv3 for a specific AF, either IPv4 or IPv6

The following commands show the syntax for OSPFv3 with AF used in this chapter:

- Router(config)# **router ospfv3** *process-id*

  This command enters router configuration mode for OSPFv3.

- Router(config-router)# **address-family** [**IPv4** | **IPv6**] **unicast**

  This command enters address-family configuration mode for the specific protocol (IPv4 or IPv6).

- Router(config-router-af)# **router-id** *router-id*

  This command configures the 32-bit OSPFv3 Router ID.

- Router(config-router-af)# **passive-interface** *interface-type interface-number*

  This command suppresses OSPF hello messages on an interface.

- Router(config-router-af)# **default-information originate**

  This command advertises the default static route into the routing domain.

- Router(config-router-af)# **area** *area-id* **stub** [**no-summary**]

This command defines the area as a stub area. The **no-summary** parameter defines the area as a totally stubby area.

- Router(config-if)# **ospfv3** *process-id* [**ipv4** | **ipv6**] **area** *area-id*

  This interface command enables OSPFv3 directly on the interface.

The three groups of "**show** command" parameters are summarized as follows:

- **ip ospf:** Refers to IPv4 information learned via OSPFv2
- **ipv6 ospf:** Refers to IPv6 information learned via traditional OSPFv3 and OSPFv3 with AF
- **ospfv3:** Refers to IPv4 and IPv6 information learned via OSPFv3 with AF

The following commands are used to verify traditional OSPFv3 configuration:

- **show ipv6 route ospf**
- **show ipv6 ospf database**
- **show ipv6 protocols**
- **show ipv6 interface**
- **show ipv6 ospf neighbor**
- **show ipv6 ospf interface**
- **ping**
- **show running-config**

The following commands are used to verify the OSPFv3 with AF configuration:

- **show ip route ospfv3**
- **show ipv6 route ospf**
- **show ospfv3 neighbor**
- **show ospfv3 database**
- **show running-config**

Review Questions

1. Which OSPF routing protocol advertises IPv4 prefixes? (Choose all that apply.)
   a. OSPFv2
   b. Traditional OSPFv3
   c. OSPFv3 with AF

2. Which OSPF routing protocol advertises IPv6 prefixes? (Choose all that apply.)
   a. OSPFv2
   b. Traditional OSPFv3
   c. OSPFv3 with AF
3. Which OSPF routing protocol uses IPv6 multicast addresses when sending OSPF messages for IPv4? (Choose all that apply.)
   a. OSPFv2
   b. Traditional OSPFv3
   c. OSPFv3 with AF
4. Which OSPF routing protocol requires the router to be configured with the **ipv6 unicast-routing** command? (Choose all that apply.)
   a. OSPFv2
   b. Traditional OSPFv3
   c. OSPFv3 with AF
5. Which OSPF routing protocol uses IPv4 multicast addresses when sending OSPF messages for IPv4? (Choose all that apply.)
   a. OSPFv2
   b. Traditional OSPFv3
   c. OSPFv3 with AF
6. Which OSPF routing protocol displays IPv6 neighbor table information using the **show ipv6 ospf neighbor** command? (Choose all that apply.)
   a. OSPFv2
   b. Traditional OSPFv3
   c. OSPFv3 with AF
7. Which command displays IPv6 routes learned via traditional OSPFv3?
   a. **show ip route ospf**
   b. **show ip route ospfv3**
   c. **show ipv6 route ospfv3**
   d. **show ipv6 route ospf**
8. Which command displays IPv6 routes learned via OSPFv3 with address families?
   a. **show ip route ospf**
   b. **show ip route ospfv3**
   c. **show ipv6 route ospfv3**
   d. **show ipv6 route ospf**
9. Which command displays IPv4 routes learned via OSPFv2?
   a. **show ip route ospf**
   b. **show ip route ospfv3**
   c. **show ipv6 route ospfv3**
   d. **show ipv6 route ospf**

10. Which command displays IPv4 routes learned via OSPFv3 with AF?
    a. **show ip route ospf**
    b. **show ip route ospfv3**
    c. **show ipv6 route ospfv3**
    d. **show ipv6 route ospf**

References

RFCs

RFC 2328, *OSPF Version 2*, J. Moy, Ascend Communications, www.ietf.org/rfc/rfc2328, April 1998.

RFC 5340, *OSPF for IPv6*, R. Coltun, Acoustra Productions, tools.ietf.org/html/rfc5340, July 2008.

RFC 5838, *Support of Address Families in OSPFv3*, A. Lindem, Ericsson, tools.ietf.org/html/rfc5838, April 2010.

Websites

IPv6 Design and Deployment LiveLessons, http://www.ciscopress.com/store/ipv6-design-and-deployment-livelessons-9780134655512

Cisco IOS IPv6 Command Reference, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book.html.

OSPFv3 Support for Address Families, www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6554/ps6599/ps6629/whitepaper_c11-668030.html.

Books

IP Routing on Cisco IOS, IOS XE, and IOS XR: An Essential Guide to Understanding and Implementing IP Routing Protocols, by Brad Edgework, Aaron Foss, and Ramiro Rios, Cisco Press.

Routing TCP/IP, by Jeff Doyle, Cisco Press.

Implementing Cisco IP Routing (ROUTE) Foundation Learning Guide (CCNP ROUTE 300-101), by Diane Teare, Cisco Press.

OSPF: Anatomy of an Internet Routing Protocol, by John Moy, Addison-Wesley.

This page intentionally left blank

Part VI

Implementing IPv6

This page intentionally left blank

Chapter 17

Deploying IPv6 in the Network

Deploying IPv6 is an extensive topic and, depending on your network and your own needs, it can mean many different things. The goal of this chapter is to introduce you to some topics you may need to consider when deploying IPv6 in the network and then direct you to the appropriate resources.

This chapter examines the following topics that are common to many deployments:

- IPv6 address plan considerations
- Configuring IPv6 VLANs
- Choosing a first-hop redundancy protocol for IPv6
- Dual stack
- DNS for IPv6
- ACLs
- Translation protocols and tunneling

An excellent next-step resource following this book and a guide to deploying IPv6 is Tim Martin's Cisco Press video series, *IPv6 Design and Deployment LiveLessons*. This video series provides a thorough and comprehensive understanding of deploying IPv6 in the network. The following is an outline of the lessons from his video series, which provides an excellent overview of topics to consider when implementing IPv6:

- **Design Considerations:** Discusses how infrastructure providers are implementing IPv6, national strategies for IPv6 deployment, and understanding the impact of the "bring-your-own-device" era and the impact of the "Internet of Things."
- **IPv6 Protocol:** Provides an overview of IPv6 fundamentals.
- **General Design Principles:** Examines many design principles, including preparing an IPv6 project plan and preparing an IPv6 address plan.

- **Host Role in IPv6:** Discusses how IPv6 is implemented in host operating systems, including Windows, Mac OS X, Linux, Chromebook, mobile operating systems, and consumer devices.
- **IPv6 Network Services:** Examines deploying DHCPv6 and DNS, implementing IPv6 multicast, and understanding IPv6 zero configuration.
- **Campus Design Considerations:** Discusses implementing and securing first-hop redundancy protocols (FHRPs), securing EIGRP for IPv6 and OSPFv3, and implementing IPv6 at the wireless access layer.
- **Data Center:** Examines IPv6 in the data center, including storage access networks (SANs), Neighbor Cache scaling, verification of server readiness, application migration, and cloud services.
- **Wide Area Networking:** Discusses IPv6 in the WAN, including remote branch design, DMVPN considerations, MACsec, MPLS, segment routing, and remote access VPNs.
- **Translation Techniques:** Discusses NAT64, server load balancing, proxy services, Web Cache Control Protocol, and network prefix translation.
- **The Internet Edge:** Examines design scenarios, BGP multihoming, Location Identification Separation Protocol, and mitigating risk with security.

Note Tim Martin, CCIE #2020, along with Jim Bailey, CCIE #5275, are the two technical editors for this book. Tim and Jim both are both senior IPv6 engineers for Cisco Systems and distinguished speakers on IPv6 at Cisco Live events all over the world. For more information about both of them, see the "About the Technical Reviewers" section at the beginning of this book.

IPv6 Address Plan Considerations

IPv6 provides significantly larger address space than IPv4, which enables you to create an address plan that meets your needs in managing your network. Many IPv4 address plans were designed primarily to make efficient use of IPv4 address space, with practical or logical addressing being more of an afterthought.

Most organizations applying for IPv6 address space are assigned at least a /48 prefix. This is a vast amount of address space, providing 65,536 /64 subnets. This amount of address space allows addresses to be grouped effectively and logically for more efficient network management, easier implementation of security policies, and better scalability. This may seem like an inordinately large number of subnets, but you will see later in this section that it allows you to organize Subnet IDs in a hierarchical manner.

There are several methodologies for creating an address plan, and the one you choose depends on the needs of your organization and what fits best with your network policies. The following are some points to consider when implementing an address plan:

- **Keep it simple:** As Veronika McKillop and Wim Verrydt point out during their IPv6 address plan presentations at Cisco Live conferences, "Keep your address plan simple. You don't want to spend weeks explaining it!" At times this can be more difficult than it sounds. As you are working on an address plan, it is easy to make something that starts out simple more complicated than it needs to be. What may seem obvious at the time of designing the plan might be less obvious in the middle of the night when someone is troubleshooting a problem on the network.

- **Use an existing IP address schema or create a new one:** In some cases you might want to use your current IPv4 address schema for IPv6. This involves translating existing IPv4 subnets and VLAN numbers into IPv6 Subnet IDs. Many times this is the easiest and fastest way to create an IPv6 address schema, and it could be a short-term solution with a redesign of the schema to happen at a later time. (Of course, putting things off to do later might very well mean that it never happens at all.) It is best to avoid letting your IPv4 address plan influence your IPv6 address plan. Implementing IPv6 is an opportunity to think differently and give new consideration to creating an IPv6 address plan.

  When designing an IPv6 address schema from scratch, you can allocate addresses according to manageability and network policies, while considering growth and scalability. A redesigned schema has the potential to simplify network operations and service offerings while providing greater flexibility and easier troubleshooting.

- **Embed information to help operations:** To help make network operations and troubleshooting easier, you can embed in the Subnet ID information such as location, VLAN, or function. This can even be carried over into global unicast and link-local uncast addresses. Be careful not to over-engineer this. Remember the first point: Keep it simple.

- **Think about the address plan structure and subnetting:** Subnetting IPv6 can be as simple or as complicated as you wish to make it. In Chapter 5, "Global Unicast Address," you saw the basics of IPv6 subnetting. Regardless of how involved your subnetting schema is, in most cases it is recommended that you subnet on a nibble (hexadecimal digit) boundary. It is good practice to design a hierarchical address schema—which is also important for scalability. With a 16-bit Subnet ID giving you 65,536 subnets, hierarchy and aggregation are important for IGP routing protocol scalability. Although VLSM is not considered part of IPv6, the concept of subnetting a subnet still exists. For example, 2001:db8:cafe::/48 can be subnetted into 16 /52 subnets:

  2001:db8:cafe:0000::/52

  2001:db8:cafe:1000::/52

  2001:db8:cafe:2000::/52

2001:db8:cafe:3000::/52

2001:db8:cafe:4000::/52

2001:db8:cafe:5000::/52

2001:db8:cafe:6000::/52

2001:db8:cafe:7000::/52

2001:db8:cafe:8000::/52

2001:db8:cafe:9000::/52

2001:db8:cafe:a000::/52

2001:db8:cafe:b000::/52

2001:db8:cafe:c000::/52

2001:db8:cafe:d000::/52

2001:db8:cafe:e000::/52

2001:db8:cafe:f000::/52

Without affecting the other 15 /52 subnets, the 2001:db8:cafe:f000::/52 subnet (the last one in the preceding list) can be isolated and subnetted further into another 16/56 subnets:

2001:db8:cafe:f000::/56

2001:db8:cafe:f100::/56

2001:db8:cafe:f200::/56

2001:db8:cafe:f300::/56

2001:db8:cafe:f400::/56

2001:db8:cafe:f500::/56

2001:db8:cafe:f600::/56

2001:db8:cafe:f700::/56

2001:db8:cafe:f800::/56

2001:db8:cafe:f900::/56

2001:db8:cafe:fa00::/56

2001:db8:cafe:fb00::/56

2001:db8:cafe:fc00::/56

2001:db8:cafe:fd00::/56

2001:db8:cafe:fe00::/56

2001:db8:cafe:ff00::/56

Of course, any of these /56 subnets can be subnetted further. The point is that you can subnet a subnet in IPv6, just as you can in IPv4.

Note For a thorough examination of IPv6 address planning and subnetting, see the book *IPv6 Address Planning: Designing an Address Plan for the Future*, by Tom Coffeen, published by O'Reilly Media.

- **Consider prefix size and aggregation:** A /64 prefix is recommended for most network segments and should be used on traditional LAN interfaces. It is important to consider the possible implications before using a prefix that is not /64. A prefix other than a /64 will break certain operations involving Neighbor Discovery such as SLAAC, Secure Neighbor Discovery (SEND), privacy extensions, parts of mobile IPv6, Protocol Independent Multicast (PIM), and Site Multihoming by IPv6 Intermediation.

  You can use a /127 prefix for inter-router links, much as you'd use the /31 prefix with IPv4. The motivation for /127 is security though whereas the /31 for IPv4 is for address conservation. Chapter 5 discusses subnetting using a /127 prefix on point-to-point links to help mitigate security threats.

  Prefix aggregation is an important part of any address plan to reduce the size of routing tables and make dynamic routing more efficient. Proper prefix aggregation ensures both scalability and stability. As stated in RFC 7381, *Enterprise IPv6 Deployment*:

  > *Plan to aggregate at every layer of network hierarchy. There is no need for variable length subnet mask (VLSM) [RFC1817] in IPv6, and addressing plans based on conservation of addresses are shortsighted. Use of prefixes longer than /64 on network segments will break common IPv6 functions such as SLAAC [RFC4862]. Where multiple VLANs or other Layer 2 domains converge, allow some room for expansion. Renumbering due to outgrowing the network plan is a nuisance, so allow room within it. Generally, plan to grow to about twice the current size that can be accommodated; where rapid growth is planned, allow for twice that growth.*

- **Plan for network growth and flexibility:** As stated in RFC 7381, it is essential to plan for expansion. Building a reserve of internal IPv6 address space allows for future growth, mergers, new locations, restructuring, and unforeseen requirements and new technologies. Obtaining the right amount of address space from your provider is essential for flexibility and expansion.

Encoding Information in the Subnet ID

When you plan an IPv6 address schema, you can really see the benefits of a separate Subnet ID in an IPv6 global unicast or unique local address. The Subnet ID can be used to provide a hierarchical structure and embed information such as location and VLAN.

Figure 17-1 shows an example of a /48 global routing prefix, which gives you a 16-bit Subnet ID for your /64 prefixes. At first a 16-bit Subnet ID that gives you 65,536 subnets may seem like overkill. But if you look at the individual hexadecimal values, you can see that this field can be use to represent network information. An advantage of such a large IPv6 address space is that it allows for better ways to organize and manage IP networks. It's not necessarily about the number of subnets but how you can use this space to more effectively manage your network.

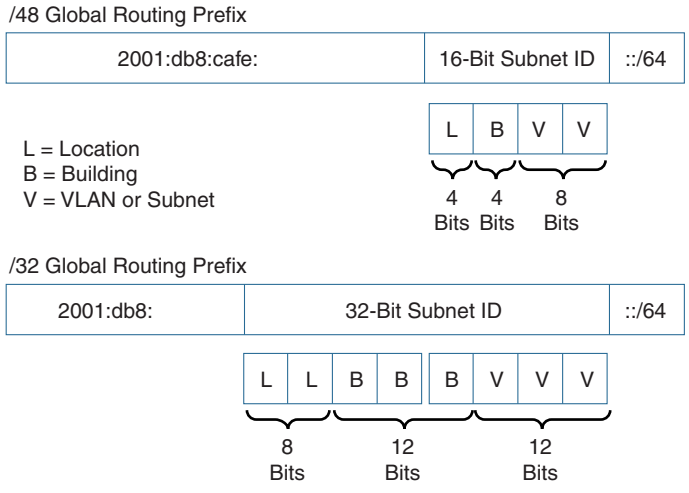


Figure 17-1 *Encoding Information in the Subnet ID*

Consider the following, shown in the top half of the example in Figure 17-1:

- **Location (4 bits):** This represents a location or site. It might be a corporate office, branch office, main campus, and so on. Allocating 4 bits allows for 16 different locations.
- **Building (4 bits):** This represents a building within a location. Allocating 4 bits allows for 16 buildings per location.
- **VLAN or Subnet (8 bits):** This represents the VLAN ID in either decimal or hexadecimal. Allocating 8 bits for the VLAN ID in hexadecimal gives you 255 VLANs per building or, if using decimal values for VLANs, 99 VLANs per building. (VLAN 0 is reserved for priority tagged frames.)

Depending on the size and needs of the organization, there are various ways to utilize the digits within the Subnet ID. For example, a single site might divide the Subnet ID using 4 bits for the location and 12 bits for the VLAN.

You can see why an organization may request a prefix shorter than a /48, such as a /32. A smaller prefix gives you more hexadecimal digits in the Subnet ID and greater flexibility in organizing your subnets/VLANs. A /32 prefix gives you a 32-bit subnet ID.

(This is the same number of IPv6 subnets as public IPv4 addresses!) A 32-bit Subnet ID is represented as 8 hexadecimal values of 4 bits each. This can make it easier to allocate the right number of subnets for each location, building, function, or whatever method is used for the IPv6 address schema. Using the location, building, and VLAN format, a possible 32-bit Subnet ID is LLBB BVVV, shown in the bottom half of Figure 17-1.

Note The address plans discussed later in this chapter provide different options for encoding information within the Subnet ID.

VLAN-Mapped Subnet ID

Although it is not required, many network administrators map VLAN IDs to IP addresses. IPv6 makes this practice even easier and usually accommodates a larger number of VLANs.

The easiest way to do this with the IPv6 Subnet ID is to simply use the decimal values of VLANs. Using the format shown in Figure 17-1, the rightmost hexadecimal digits in the Subnet ID would be used to accommodate VLANs 1 through 99 (VLAN 0 is reserved). The following example uses a location value of 1, a building number of 7, and the last two digits (shown in bold) representing the decimal value for the VLAN:

VLAN 1: 2001:db8:cafe:17**01**::/64

VLAN 2: 2001:db8:cafe:17**02**::/64

VLAN 10: 2001:db8:cafe:17**10**::/64

VLAN 25: 2001:db8:cafe:17**25**::/64

VLAN 50: 2001:db8:cafe:17**50**::/64

VLAN 99: 2001:db8:cafe:17**99**::/64

The advantage of using the decimal VLAN representation in the Subnet ID is that it is easier to recognize. The disadvantage is that it you are limited to fewer digits: 0–9 for decimal compared to 0–f for hexadecimal. Converting the VLANs to hexadecimal gives you the flexibility of using all 8 bits, which adds up to 255 possibilities (VLAN 0 is reserved). The following example shows the same VLAN numbers as above, now represented in hexadecimal and including VLANs above 99:

VLAN 1: 2001:db8:cafe:17**01**::/64

VLAN 2: 2001:db8:cafe:17**02**::/64

VLAN 10: 2001:db8:cafe:17**0a**::/64

VLAN 25: 2001:db8:cafe:17**19**::/64

VLAN 50: 2001:db8:cafe:1732::/64

VLAN 99: 2001:db8:cafe:1763::/64

VLAN 125: 2001:db8:cafe:177d::/64

VLAN 197: 2001:db8:cafe:17c5::/64

VLAN 255: 2001:db8:cafe:17ff::/64

Note Allocating 3 hexadecimal values (12 bits) for VLANs within the Subnet ID provides 4096 VLANs (represented in hexadecimal). This is the maximum number of VLANs because the IEEE 802.1Q tag allocates 12 bits for the VLAN ID. The 802.1Q header is a 32-bit field between the source MAC address and the EtherType/Length field in the original Ethernet frame. The first 16 bits of the 802.1Q header is the Tag Protocol Identifier (TPID), the next 3 bits is the Priority Code Point (PCP), the next bit is the Drop Eligible Indicator, and the last 12 bits is the VLAN identifier (VID).

IPv6 Address Plans

There is no need to reinvent the wheel when creating an IPv6 address plan for a network. A number of IPv6 address planning guides are available from a wide range of sources. Some provide a general outline while others go into significant detail, including examples. The following are just some of the IPv6 address plans available:

- *IPv6 Design and Deployment LiveLessons*, Lesson 3.4: "Building an Address Plan," by Tim Martin, www.ciscopress.com/store/ipv6-design-and-deployment-livelessons-9780134655512
- *Preparing an IPv6 Address Plan*, IPv6 Forum, www.ipv6forum.com/dl/presentations/IPv6-addressing-plan-howto.pdf
- *IPv6 Address Planning: Designing an Address Plan for the Future*, by Tom Coffeen, O'Reilly Media
- *IPv6 Addressing Guide, Cisco Smart Business Architecture Guides,* Cisco Systems, www.cisco.com/c/dam/en/us/td/docs/solutions/Enterprise/Borderless_Networks/Smart_Business_Architecture/February2012/SBA_Ent_BN_IPv6AddressingGuide-February2012.pdf
- *IPv6 Addressing White Paper*, Cisco Systems, www.cisco.com/c/dam/en_us/solutions/industries/docs/gov/IPv6_WP.pdf
- *IPv6 Addressing Plan Basics*, Infoblox, https://www.infoblox.com/wp-content/uploads/2016/04/infoblox-whitepaper-ipv6-addressing-plan-basics_1.pdf

- *IPv6 Address Planning: Guidelines for IPv6 Address Allocation*, Internet Society (ISOC), www.internetsociety.org/deploy360/resources/ipv6-address-planning-guidelines-for-ipv6-address-allocation/

Note If you have an opportunity to attend Cisco Live, I recommend the session *How to Write an IPv6 Addressing Plan* (BRKRST-2667), presenter Wim Verrydt.

IPv6 VLANs

Configuring IPv6 VLANs on a Layer 3 switch is much like configuring IPv4 VLAN information. There are differences, however. A VLAN interface typically acts as the default gateway on the VLAN. With IPv4 VLANs, this equates to configuring an IPv4 address on the VLAN, which is used by end devices as the default gateway.

An IPv6 VLAN interface has the same function but serves the additional purpose of sending ICMPv6 Router Advertisement messages on the LAN just like a traditional router's IPv6 interface. End devices need the RA messages for the default gateway address and dynamic address determination.

Figure 17-2 shows WinPC connected to multilayer switch Switch1 in VLAN 5. The topology is part of 2001:db8:cafe::/48 and subnetted using the 16-bit Subnet ID format LBVV, as shown in Figure 17-1. Both VLANs 5 and 10 are in location 1, building 7. VLAN 5 has the Subnet ID 1705, and VLAN 10 has the Subnet ID 170a (decimal 10 is equivalent to hexadecimal 0xa).

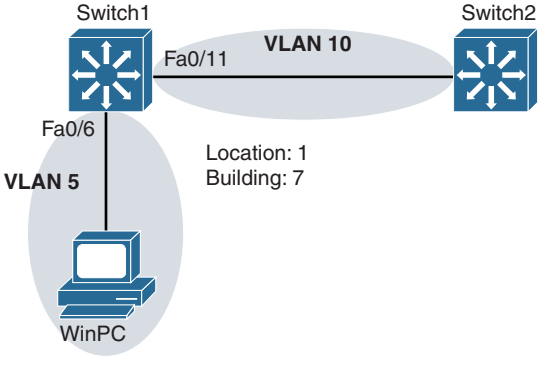


Figure 17-2 *Topology for Configuring IPv6 VLANs*

Example 17-1 shows the configuration of interface VLAN 5 on Switch1. The GUA address uses the Subnet ID 1705, along with the Interface ID ::1, both of which are appended to the 2001:db8:cafe::/48 prefix.

Example 17-1 *Configuring IPv6 Addresses on VLAN 5*

```
Switch1(config)# interface vlan 5
Switch1(config-if)# ipv6 address 2001:db8:cafe:1705::1/64
Switch1(config-if)# ipv6 address fe80::5:1 link-local
Switch1(config-if)# ipv6 nd ?
  advertisement-interval  Send an advertisement interval option in RA's
  autoconfig              Automatic Configuration
  cache                   Cache entry
  dad                     Duplicate Address Detection
  managed-config-flag     Hosts should use DHCP for address config
  na                      Neighbor Advertisement control
  ns-interval             Set advertised NS retransmission interval
  nud                     Neighbor Unreachability Detection
  other-config-flag       Hosts should use DHCP for non-address config
  prefix                  Configure IPv6 Routing Prefix Advertisement
  ra                      Router Advertisement control
  reachable-time          Set advertised reachability time
  router-preference       Set default router preference value

Switch1(config-if)# exit
Switch1(config)# interface fastethernet 0/6
Switch1(config-if)# switchport access vlan 5
Switch1(config-if)# switchport mode access
Switch1(config-if)# end
Switch1# show ipv6 interface brief | section Vlan5
Vlan5                      [up/up]
    FE80::5:1
    2001:DB8:CAFE:1705::1
Switch1#
```

Continuing with Example 17-1, Switch1's link-local address is configured using the VLAN ID as part of the Interface ID. Previous chapters show the same link-local address configured on all of the router's interfaces. This was done for simplicity and is fine for inter-router links. Using the VLAN ID as part of the link-local address is more consistent with a logical address structure and can make troubleshooting easier. Each VLAN interface then has a link-local address associated with that VLAN. For example, if you convert the decimal VLAN IDs to hexadecimal, these would be the values for the link-local addresses of other VLANs:

- fe80::1:1 for VLAN 1
- fe80::2:1 for VLAN 2

- fe80::a:1 for VLAN 10
- fe80::32:1 for VLAN 50

This is just one example of embedding the VLAN ID in the link-local address. Other methods involve using decimal values or using the VLAN ID as the Interface ID, omitting the :1. For example, if you were to combine both of these techniques, the link-local address for VLAN 10 would be fe80::10 and the link-local address for VLAN 50 would be fe80::50.

The **ipv6 nd ?** command in Example 17-1 displays the same options shown on router interfaces in previous chapters. As described shortly, the VLAN can send RA messages with the same configurable options as a router interface.

The last set of commands in Example 17-1 shows interface FastEthernet 0/6 assigned as an access port to VLAN 5 and the IPv6 addresses verified for VLAN 5 using the **show ipv6 interface brief** command.

Example 17-2 shows how to enable Switch1 to send Router Advertisements. Similar to a router, the **ipv6 unicast-routing** command is required to enable Switch1 as an IPv6 router. The **show ipv6 interface vlan 5** command verifies that VLAN 5 is a member of both the all IPv6 devices ff02::1 and all IPv6 routers ff02::2 multicast groups. The output verifies that interfaces that are members of VLAN 5 send RA messages every 200 seconds and with the A flag set to 1, enabling SLAAC (the M and O flags are set to 0). These are the same defaults as on a traditional IPv6 router.

Example 17-2 *Enabling Switch1 as an IPv6 Router*

```
Switch1(config)# ipv6 unicast-routing
Switch1(config)# exit
Switch1# show ipv6 interface vlan 5
Vlan5 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::5:1
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1705::1, subnet is 2001:DB8:CAFE:1705::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF00:1
    FF02::1:FF05:1
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  Output features: Check hwidb
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 30000)
  ND advertised reachable time is 0 (unspecified)
```

```
  ND advertised retransmit interval is 0 (unspecified)
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  ND advertised default router preference is Medium
  Hosts use stateless autoconfig for addresses.
Switch1#
```

The **debug ipv6 nd** command on Switch1 in Example 17-3 verifies that Switch1 is sending Router Advertisements for VLAN 5. Notice that the link-local address fe80::5:1 is the source IPv6 address of the RA, with 2001:db8:cafe:1705::/64 advertised as the prefix.

Example 17-3 **debug ipv6 nd** *Command on Switch1*

```
Switch1# debug ipv6 nd
ICMP Neighbor Discovery events debugging is on
Switch1#
*Mar  1 01:12:09.930: ICMPv6-ND: Request to send RA for FE80::5:1
*Mar  1 01:12:09.930: ICMPv6-ND: Setup RA from FE80::5:1 to FF02::1 on Vlan5
*Mar  1 01:12:09.930: ICMPv6-ND:  MTU = 1500
*Mar  1 01:12:09.930: ICMPv6-ND:     prefix = 2001:DB8:CAFE:1705::/64 onlink
  autoconfig
*Mar  1 01:12:09.930: ICMPv6-ND:             2592000/604800 (valid/preferred)
Switch1# undebug all
```

The **ipconfig** command in Example 17-4 shows WinPC's IPv6 addressing information. With the flags in the RA message suggesting SLAAC, WinPC creates its IPv6 GUA addresses using the prefix in the RA and uses the source IPv6 address of the RA message as its default gateway.

Example 17-4 **ipconfig** *Command on WinPC*

```
WinPC> ipconfig
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv6 Address  . . . . . . . . . . . : 2001:db8:cafe:1705:ddbd:6053:612e:77ab
   Temporary IPv6 Address  . . . . . . : 2001:db8:cafe:1705:4c4c:7708:4b83:7b85
   Link-local IPv6 Address . . . . . : fe80::ddbd:6053:612e:77ab%11
   Autoconfiguration IPv4 Address  . : 169.254.119.171
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : fe80::5:1%11
```

You can also configure ports on Layer 3 switches as routed ports. A routed port is similar to physical interface on a router. The **no switchport** interface command makes the interface Layer-3-capable.

Example 17-5 shows the FastEthernet 0/11 interface configured as a routed port. This interface isn't actually associated with any VLAN because it is a routed port. For purposes of our address schema, we are using VLAN 10 for the VLAN portion of the Subnet ID. Switch1's Fa0/11 interface is configured using 170a as the Subnet ID in its GUA address 2001:db8:cafe:170a::1/64. (0xa is the hexadecimal equivalent of decimal 10.) The link-local address uses the same format used for the VLAN 5 interface, with the VLAN number part of the Interface ID. The **show ipv6 interface brief** command is used to verify both addresses.

Example 17-5 *Configuring a Routed Port*

```
Switch1(config)# interface fastethernet 0/11
Switch1(config-if)# no switchport
Switch1(config-if)# ipv6 address 2001:db8:cafe:170a::1/64
Switch1(config-if)# ipv6 address fe80::a:1 link-local
Switch1(config-if)# end
Switch1# show ipv6 interface brief
Vlan1                      [up/up]
    unassigned
Vlan5                      [up/up]
    FE80::5:1
    2001:DB8:CAFE:1705::1
FastEthernet0/1            [down/down]
    unassigned
...
FastEthernet0/11           [up/up]
    FE80::A:1
    2001:DB8:CAFE:170A::1
<output omitted for brevity>
Switch1#
```

IPv6 First Hop Redundancy Protocols

With a single IPv4 default gateway, if that device fails, a host is unable to send packets beyond its own network. Even if there is another IPv4 router on the segment, there is no dynamic method in host operating systems to automatically determine the address of a new default gateway.

IP routing redundancy is designed to allow for transparent failover at the first-hop IP router. First hop redundancy protocols (FHRPs) are a set of routers or Layer 3 switches that work together to present the illusion of a single virtual default gateway (router) to a host on the LAN. This is done with two or more physical routers sharing an IP address and a virtual MAC address acting as a single virtual router. FHRP is available for both IPv4 and IPv6.

Cisco IOS offers three FHRP protocols:

- Hot Standby Router Protocol (HSRP)
- Virtual Router Redundancy Protocol (VRRP)
- Gateway Load Balancing Protocol (GLBP)

Note The details and configuration of FHRP protocols is beyond the scope of this book. For each FHRP protocol, the references section of this chapter provides suggestions for more information.

FHRP protocols are available for both IPv4 and IPv6, but there is another option inherent in IPv6: ICMPv6 Neighbor Discovery. ICMPv6 Neighbor Discovery provides dynamic default gateway information in the form of Router Advertisements messages. ICMPv6 ND and each of the FHRP options are discussed next.

ICMPv6 Neighbor Discovery

An IPv6 router sends an ICMPv6 Router Advertisement message every 200 seconds or upon receiving a Router Solicitation message. A router sends an RA message to suggest to devices on the segment how to obtain address information dynamically and provides its own IPv6 link-local address as a default gateway.

Figure 17-3 shows how failover can occur with multiple routers sending ICMPv6 RA messages.

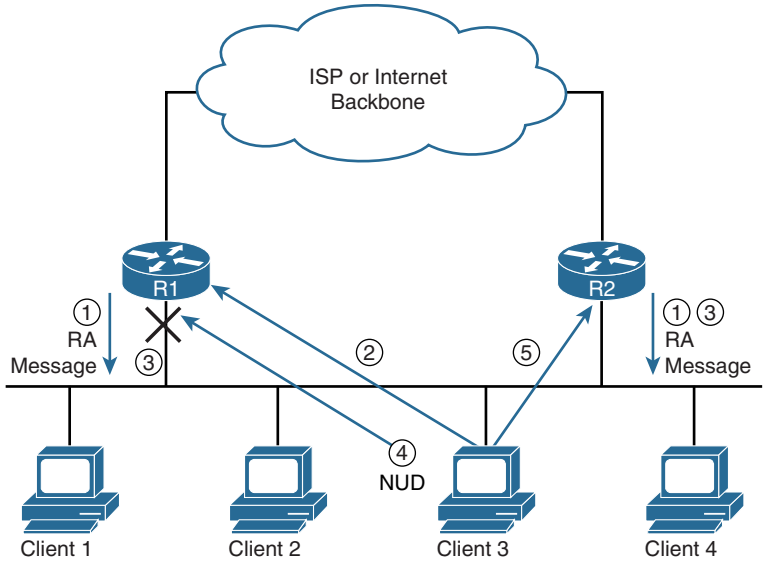


Figure 17-3 *Failover with ICMPv6 Neighbor Discovery*

Figure 17-3 shows two routers, R1 and R2, involved in the following steps:

Step 1. Routers R1 and R2 send RA messages, announcing themselves as default routers for the network.

Step 2. Client3 receives RA messages from both routers and installs both R1 and R2 as options in its Default Router List. Client3 received the RA from R1 first and selects it as its primary default gateway. (Other options may affect this choice, such as the Default Router Preference in the RA message.) The RA message includes a Router Lifetime field, which informs a host of the duration, in seconds, that the router should be used as the default gateway. The default is 1800 seconds (30 minutes). The host refreshes its timer, indicating that a router is a valid default gateway, every time it receives an RA message from that router.

Step 3. For some reason, router R1 is now no longer available and therefore is no longer sending RA messages.

Step 4. Client3 has packets to send to another network. Client3 still considers R1 as its primary default gateway because it has been less than 30 minutes since R1 has failed (Router Lifetime). Although Client3 is not receiving RA messages from R1, R1 is still the primary router in its Default Router List. Client3 checks its Neighbor Cache for the MAC address of R1. Whether this entry is in the Reachable or Stale state, ICMPv6 Neighbor Unreachability Detection (NUD) determines that R1 is no longer available. Client3 sends three ICMPv6 Neighbor Solicitation (NS) messages in an attempt to confirm the MAC address of the default gateway of R1. Since it does not receive a Neighbor Advertisement (NA) message in reply from R1, Client3 determines that R1 is no longer available.

Step 5. Client3 attempts address resolution by sending an NS message to R3, the next router in its Default Router List. Upon receiving an NA from R3, Client3 updates its Neighbor Cache and uses R3 as its new default gateway. (Again, this is dependent on the client's operating system.)

Note The Neighbor Cache and Neighbor Unreachability Detection are discussed in Chapter 13, "ICMPv6 Neighbor Discovery."

ICMPv6 Neighbor Discovery provides some default gateway redundancy, but it does have some drawbacks. Unless the Router Lifetime timer has already expired, it is dependent on the clients to determine whether the primary default gateway has failed. It takes about 40 seconds for a client using NUD to determine that the primary default gateway has failed.

Faster failover can be achieved by modifying two timers: the interval of the Router Advertisement and the duration of the Router Lifetime. RA messages are sent out an interface every 200 seconds by default, and the default Router Lifetime is 1800 seconds. Example 17-6 shows both of these timers modified to 1 second each.

Example 17-6 *Configuring the RA Interval and Router Lifetime*

```
R1(config-if)# ipv6 nd ra interval ?
  <4-1800>  RA Interval (sec)
  msec      Interval in milliseconds

R1(config-if)# ipv6 nd ra interval 1
R1(config-if)# ipv6 nd ra lifetime ?
  <0-9000>  RA Lifetime (seconds)

R1(config-if)# ipv6 nd ra lifetime 1
R1(config-if)#
```

Note Both the **ipv6 nd ra interval** and **ipv6 nd ra lifetime** commands are discussed in Chapter 9, "Stateless Address Autoconfiguration (SLAAC)."

Before implementing Neighbor Discovery as a first hop failover method, you should consider the following:

- **Client behavior:** As mentioned earlier, what happens when the Router Lifetime timer expires depends on the client's operating system.
- **CPU processing:** Increasing the RA interval requires every device on the network to process RA messages more often. The changes in Example 17-6 mean that clients need to process RA messages every second instead of every 200 seconds. This can be an issue in data centers with thousands of virtual machines (VMs). This can also be an issue for a router having to generate more RA messages and also possibly process more RS messages. With multiple interfaces on the router, the amount of CPU processing on the router can easily add up.
- **Load balancing:** The router that a client chooses as its default gateway is typically based on which RA message it receives first. Because Neighbor Discovery doesn't provide any type of load balancing, this can lead to one router performing a significant amount of the packet forwarding for the network.

Note There are two excellent articles regarding Neighbor Discovery as a failover technique: Jeremy Stretch's "Can IPv6 Neighbor Discovery Provide High Availability?" is available at packetlife.net/blog/2011/apr/18/ipv6-neighbor-discovery-high-availability, and Ivan Pepelnjak's "Do We Need FHRP (HSRP or VRRP) for IPv6?" is available at blog.ipspace.net/2012/12/do-we-need-fhrp-hsrp-or-vrrp-for-ipv6.html.

HSRP and VRRP

Both Hot Standby Router Protocol (HSRP) and Virtual Router Redundancy Protocol (VRRP) enable two or more devices to work together in a group, sharing a single IP address—the *virtual IP address*. The virtual IP address is used by end devices on the LAN as their default gateway address. In an HSRP or VRRP group, a single router is elected to handle all requests sent to the virtual IP address.

With HSRP, the router elected to handle requests sent to the default gateway is known as the *active router*. An HSRP group has one active router and one standby router. If the active router fails, then the standby router assumes the role of the active router. HSRP is a Cisco proprietary protocol and defined in the informational RFC 2281, *Cisco Hot Standby Router Protocol (HSRP)*. HSRP version 2 is required for IPv6.

VRRP is an IETF standard protocol similar in functionality to HSRP. There are only slight differences in terminology and operations. A VRRP group consists of one master router (equivalent to an HSRP active router) and one or more backup routers (equivalent to HSRP standby routers). VRRP is defined in RFC 4793, *Virtual Router Redundancy Protocol (VRRP)*, and in RFC 5798, *Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6*. VRRP version 3 is required for IPv6.

Figure 17-4 shows three routers: a virtual router and routers R1 and R2, the only physical routers on the network. The virtual router doesn't physically exist. The IPv6 address (link-local address) and virtual MAC address of the virtual router are used by hosts as their default gateway information. In the topology, R1 is the active router and handles all requests sent to the IPv6 address of the virtual router.

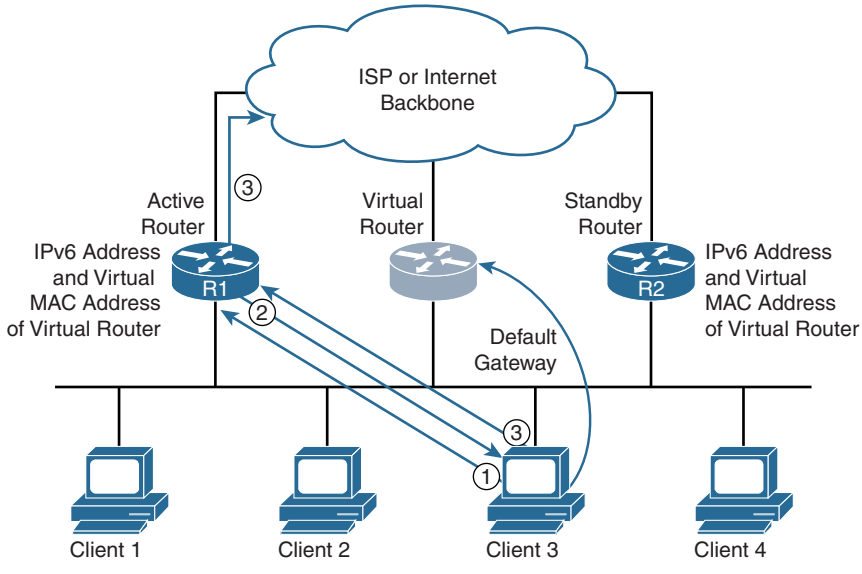


Figure 17-4 *Client Using Active Router in HSRP Topology*

The following steps describe Client3 sending a packet to the default gateway (refer to Figure 17-4):

Step 1. Client3 has a packet to send to a device on another network. It checks its Neighbor Cache for the MAC address of its default gateway, which is the IPv6 address of the virtual router. The entry doesn't exist in Client3's Neighbor Cache, so it sends a Neighbor Solicitation message to resolve the IPv6 address to its associated MAC address.

Step 2. R1, acting as the active router, responds with a Neighbor Advertisement message containing the virtual MAC address assigned to the virtual router.

Step 3. The client encapsulates the packet in an Ethernet frame, using the virtual MAC address, and sends the packet to the virtual router. R1 receives the packet and forwards it on toward the destination.

If the active router R1 fails, R2 assumes the role of the active router. The default gateway information on the clients remains the same. From the perspective of the client, there is no change in the status of the default gateway, and packet forwarding continues as normal.

Note The active router also responds to any messages sent to any of its own IPv6 addresses.

Note For more information on HSRP and VRRP, see *First Hop Redundancy Protocols Configuration Guide*, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipapp_fhrp/configuration/15-mt/fhp-15-mt-book.html.

GLBP

Gateway Load Balancing Protocol (GLBP) provides automatic router backup for IPv6 hosts configured with a single default gateway on an Ethernet LAN. Multiple first hop routers on the LAN combine to offer a single virtual first-hop IPv6 router while sharing the IPv6 packet-forwarding load.

GLBP performs a similar function to HSRP and VRRP. GLBP allows multiple routers to participate in a GLBP router group configured with a virtual IPv6 address.

With HSRP and VRRP, one router is elected to be the active router to forward packets sent to the virtual IPv6 address for the group. The other routers in the group are redundant until the active router fails. These standby routers remain mostly idle, with unused bandwidth. Although multiple active routers can be configured using different groups for the same set of routers, this requires additional configuration. Typically, each active router becomes the default gateway for one or more VLANs. The hosts in each VLAN use the virtual IPv6 address of the active router for their VLAN. This provides for per-VLAN load balancing.

The advantage of GLBP is that it natively provides load balancing over multiple routers (gateways) using a single virtual IPv6 address and multiple virtual MAC addresses. The forwarding load is shared among all routers in a GLBP group rather than being handled by a single router while the other routers stand idle. Each host is configured with the same virtual IPv6 address, and all routers in the virtual router group participate in forwarding packets.

GLBP elects one router to be the active virtual gateway (AVG) for the group. Other routers in the group are active virtual forwarders (AVFs) and provide backup for the AVG. Each AVF has its own virtual MAC address and acts as a default gateway for the LAN. The AVG is also an AVF. The AVG manages the virtual MAC address assignments for the AVFs and determines which AVF will forward packets for a client on the LAN.

All the clients on the LAN use the same virtual IPv6 address (a link-local address) for their default gateway. The AVG receives and replies to all Neighbor Solicitation messages for the virtual IPv6 address. In its Neighbor Advertisement messages, the AVG assigns the virtual MAC address of one of the AVFs. The load-balancing algorithm determines how the AVG assigns the virtual MAC addresses. GLBP provides several load-balancing options.

Figure 17-5 shows the AVG and AVFs forwarding packets. Routers R1, R2, and R3 are each receiving packets and forwarding packets for remote destinations.

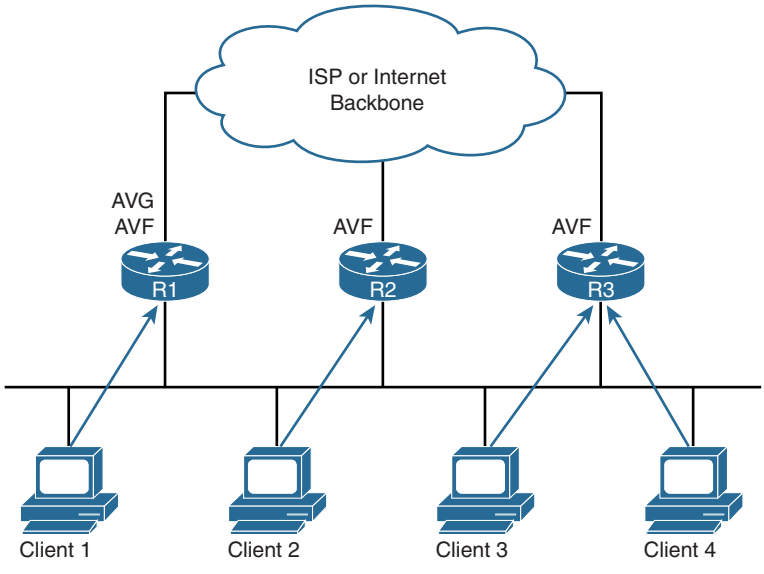


Figure 17-5 *GLBP Topology*

Note For more information on GLBP, see *First Hop Redundancy Protocols Configuration Guide*, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipapp_fhrp/configuration/15-mt/fhp-15-mt-book.html.

Selecting an FHRP

This section has introduced four methods for achieving transparent failover at the first-hop IPv6 router. It is considered best practice to use an FHRP protocol instead of relying on ICMPv6 Neighbor Discovery messages. Failover occurs with ICMPv6 messages, but convergence is slow, and fine-tuning can lead to other problems, as discussed earlier.

The question comes down to which FHRP to use. The requirements of your network and the specific features you require will typically dictate which FHRP is the best solution for your environment. In general, GLBP offers the advantage of native load balancing. Per-VLAN load balancing can be achieved with HSRP and VRRP, but you must configure it manually. However, GLBP is Cisco proprietary and doesn't work on non-Cisco devices.

If you don't need native load balancing but need only failover, your choices are either HSRP or VRRP. (HSRP and VRRP can be configured to provide per-VLAN load balancing.) What is the better choice between HSRP and VRRP? As it is commonly said, HSRP and VRRP are the same, only different. As you saw earlier, there isn't much difference between the two protocols. VRRP is standards-based, and HSRP is Cisco-proprietary. Because it is generally considered best practice to implement the standard, VRRP is typically the recommended solution.

Dual Stack

A dual-stacked device is a device that has complete support for both IPv4 and IPv6. It can be a host, printer, server, router, or any other device that can be configured to support both protocols. In the IPv4 world, this support includes IPv4 addresses, Address Resolution Protocol (ARP), and Internet Control Message Protocol (ICMP) for IPv4. An IPv4 router supports IPv4 static routes and IPv4 routing protocols such as Enhanced Interior Gateway Routing Protocol (EIGRP) and Open Shortest Path First version 2 (OSPFv2). An IPv4 router may also provide network address translation (NAT).

In the IPv6 realm, support means more than just a network header with longer addresses. IPv6 support includes one or more IPv6 global unicast addresses and a link-local address and ICMPv6 operations including Stateless Address Autoconfiguration (SLAAC) and Duplicate Address Detection (DAD). An IPv6 router can route IPv6 packets using static routes and implement IPv6 routing protocols such as EIGRP for IPv6 and OSPFv3. An IPv6 router sends out ICMPv6 Router Advertisement messages and can perform tunneling or translation services.

As illustrated throughout this book, there is much more to IPv6 than just 128-bit source and destination addresses. Support for IPv6 includes implementing new protocols and processes such as ICMPv6 Neighbor Discovery. IPv6 support for most host operating systems, such as Windows, Mac OS, and Linux, has been available for several years. Cisco introduced IPv6 support with Cisco IOS Release 12.2(2)T in 2000.

Dual-stacked devices are not new with IPv6. Before IPv4 became the predominant network protocol, many organizations ran IPX, AppleTalk, DECnet, or SNA on their devices, along with IPv4.

When you communicate with an IPv4 device, it behaves like an IPv4-only device. When you communicate with an IPv6 device, it acts like an IPv6-only device. Figure 17-6 illustrates an IPv4-only application that can be carried in a TCP or UDP segment identified by the transport layer's port number. The segment is then encapsulated in an IPv4 packet, with the data or payload identified by the value in the IPv4 protocol field. For transport over the link, the IPv4 packet is encapsulated in an Ethernet frame and identified by 0x0800 in the Ethernet Type field.

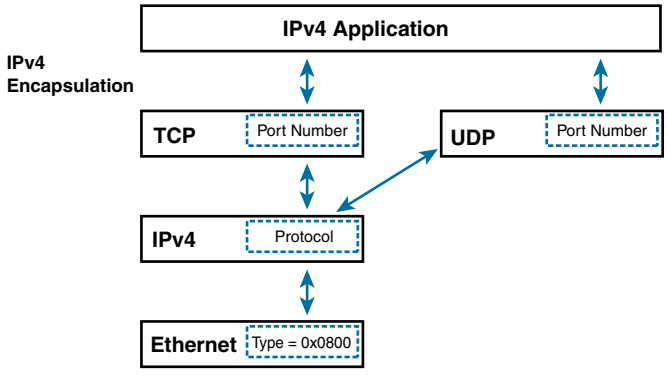


Figure 17-6 *IPv4 Application Using the IPv4 Stack*

On dual-stacked devices with an application that supports both IPv4 and IPv6 protocols, the process is almost identical, as shown in Figure 17-7. TCP and UDP port numbers identify the application. When the IPv6 stack is selected, the Next Header field in the IPv6 header identifies the transport protocol. The IPv6 packet is encapsulated in an Ethernet frame, which is using the Ethernet Type field 0x86dd. It is important to remember that Ethernet is at Layer 2. Although it has a different Type field when carrying an IPv6 packet, the Ethernet frame is not treated any differently by the switch. When a typical Layer 2 access switch does not support IPv6, it only means that you will not be able to do anything that requires Layer 3 awareness, such as network management, use ACLs, or implement QoS. The switch can still forward frames, regardless of whether its payload is an IPv4 or IPv6 packet.

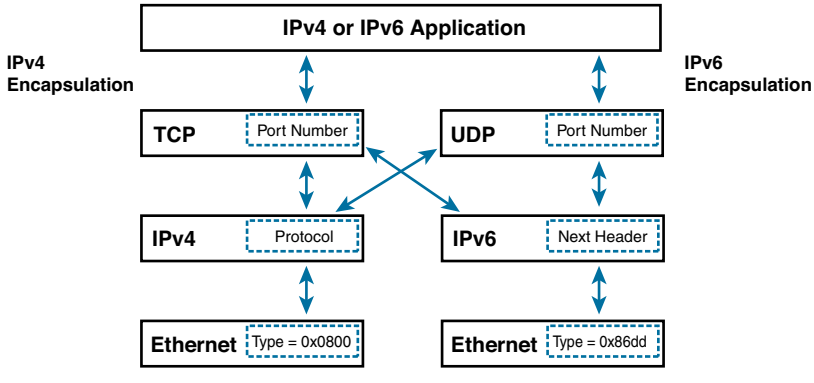


Figure 17-7 *Application Using Both IPv4 and IPv6 Stacks*

Although an application can support both IPv4 and IPv6, the device doesn't just randomly select which protocol to use. Figure 17-8 shows an example of a dual-stacked host. If an application supports both IPv4 and IPv6 protocol stacks, it requests all available addresses for the fully qualified domain name (FQDN) from a Domain Name System (DNS) server. The DNS server replies with all available addresses, which can include both IPv4 and IPv6 addresses. If the DNS server sends both IPv4 and IPv6 addresses, the operating system chooses which protocol to use. In most cases, the application defaults to the IPv6 address, but it depends on the operating system. The device then connects to the destination using the selected IP protocol stack.

In step 1 in Figure 17-8, dual-stacked host X sends a DNS query for the quad-A (AAAA) record for www.example.com. In step 2, the DNS server returns a DNS query response that contains both the quad-A and A records for www.example.com. In step 3, host X uses the quad-A record to begin communicating with the www.example.com server. (DNS is discussed later in this chapter.)

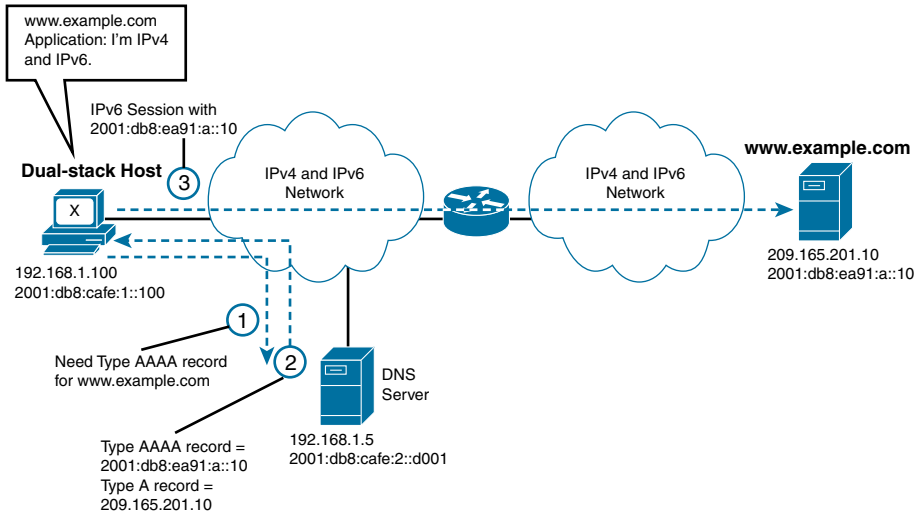


Figure 17-8 *An IPv4- and IPv6-Aware Application*

Note If you are interested in understanding the details of the application programming interface (API) for IPv6 and the operational issues with DNS, see RFC 3493, *Basic Socket Interface Extensions for IPv6*, and RFC 4472, *Operational Considerations and Issues with IPv6 DNS*. Again, it is typically left to the underlying operating system to determine the transport.

IPv6 Address Format in URL Syntax

When an IP address is used instead of a domain name, the application uses the protocol stack associated with the particular IP address. If the user enters an IPv4 address, the

application uses IPv4. If the user enters an IPv6 address, the application chooses IPv6. Most of the time when using a web browser, the address is entered by using the URL (uniform resource locator), such as http://www.example.com. The IP address could be implemented instead, but the whole idea of using domain names is to avoid having to remember IP addresses. (Now with content delivery networks and other services, it is not always easy or even possible to use an IP address instead of a domain name in the browser.) However, there are times when it is necessary to enter an IP address instead of the URL in the browser, such as when debugging or using the browser interface to configure a device. For example, using IPv4, you can enter **http://192.168.1.1** to connect and configure many home routers.

So, how do you enter an IPv6 address in a browser? IPv6 addresses are not directly compatible with URLs because of the pesky colon (:) they contain. A browser interprets a colon as a port number. For example, accessing a home router on port 5000 would mean entering the URL **http://192.168.1.1:5000**. RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, defines the format for literal IPv6 addresses in URLs. The idea is to make it easy to cut and paste IPv6 addresses into a browser's address bar. The IPv6 address is contained in brackets followed by any additional URL syntax. Table 17-1 shows examples of literal IPv6 addresses using the URL syntax described in RFC 3986.

Table 17-1 *Literal IPv6 Address Format in URL Syntax*

Literal IPv6 Address	URL
fedc:ba98:7654:3210:fedc:ba98:7654:3210	http://[fedc:ba98:7654:3210:fedc:ba98:7654:3210]:80/index.html
1080:0:0:0:8:800:200c:4171	http://[1080:0:0:0:8:800:200c:417a]/index.html
3ffe:2a00:100:7031::1	http://[3ffe:2a00:100:7031::1]
1080::8:800:200C:417A	http://[1080::8:800:200C:417A]/foo
2010:836B:4179::836B:4179	http://[2010:836B:4179::836B:4179]

Configuring a dual-stacked network means configuring devices for both IPv4 and IPv6. This can include hosts, routers, multilayer switches, printers, and any other devices that have IP addresses. As far as IP is concerned, these are two totally separate networks—just sharing the same "space." They are like roommates who don't really see or notice each other or who pass each other like "ships in the night."

Chapter 15, "EIGRP for IPv6," and Chapter 16, "OSPFv3," discuss dual-stack support for routing IPv4 and IPv6.

DNS

DNS is used with both IPv4 and IPv6 for name-to-address mappings. Like IPv4, IPv6 also supports the reverse mapping of IPv6 addresses to DNS names. In many ways, the need for DNS in the IPv6 world is much greater because of the longer IPv6 addresses.

A name server or DNS server is used to track information associated with domain names, including a database of host name–to–address mappings. Each name can map to one or more IPv4 addresses, IPv6 addresses, or both address types. As with IPv4, each host name is represented in the DNS server by two DNS records: a resource record (or address record) and a reverse mapping pointer record.

With IPv4, a DNS A resource record maps a domain name to an IPv4 address. Similarly, in IPv6, the AAAA (quad-A) resource record maps a domain name to an IPv6 address. The AAAA-type record is described in RFC 3596, *DNS Extensions to Support IP Version 6*. The four As (AAAA) indicate that the IPv6 address is four times the size of the IPv4 address. The AAAA record is structured in very much the same way as the A record, but it is much larger. IPv4 or IPv6 can be used to transport both record types.

Table 17-2 shows both types of records, using the fully qualified domain name (FQDN), which specifies the exact location in the DNS hierarchy, such as www.test.org or mail.example.com.

Table 17-2 *DNS Resource Records for IPv4 and IPv6*

Protocol	Resource Record	DNS Mapping
IPv4	A record	www.test.org A 209.165.200.225
IPv6	AAAA record	www.test.org AAAA 2001:db8:cafe:1234:0:0:0:a1

For reverse DNS lookups, IPv6 addresses use the special domain ip6.arpa. Equivalent to a pointer record (PTR) in IPv4, the IPv6 pointer record maps an IPv6 address to a host name. An IPv6 pointer record appears as a name in the domain with a sequence of nibbles (hexadecimal values), separated by periods (.), in reverse order. Table 17-3 shows the IPv6 pointer records for www.test.org IPv4 and IPv6 addresses.

Table 17-3 *DNS Pointer Records for IPv4 and IPv6*

Protocol	Pointer Record Format
IPv4	225.200.165.209 www.test.org
IPv6	1.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.4.3.2.1.e.f.a.c.8.b.d.0.1.0.0.2.ip6.arpa. www.test.org

BIND (Berkeley Internet Name Domain) is the most widely used DNS software on the Internet. BIND is open source software that implements the DNS protocols for the Internet. BIND has supported DNS for IPv6 since version 9.

DNS resolvers are the client-side of DNS communications responsible for initiating queries to servers for the translation of domain names to IP addresses. A DNS server that supports IPv6 AAAA records does not necessarily have to be queried over IPv6. It can reply to these requests using IPv4. However, it is highly recommended that a DNS server be reachable over IPv6.

As with IPv4, you can configure static host names for IPv6 addresses. The **ipv6 host** command is similar to the **ip host** command used with IPv4, except that it is IPv6 specific. The command syntax for the **ipv6 host** command, which defines a static host name–to–address mapping in the host name cache, is as follows:

```
Router(config)# ipv6 host name [port] ipv6-address1 [ipv6-address2...ipv6-address4]
```

This command sequence features the following terms:

- *name*: The name of the IPv6 host. The first character can be either a letter or a number. When it is a number, the operations you can perform are limited.
- *port* **(optional):** The default telnet port number for the associated IPv6 addresses. (It is highly recommended to use SSH instead of telnet.)
- *ipv6-address1*: The associated IPv6 address.
- *ipv6-address2...ipv6-address4* **(optional):** Additional associated IPv6 addresses. You can bind up to four addresses to a host name.

In Example 17-7, the **ipv6 host** command is applied on R1 to map the IPv6 interface addresses of routers R2 and R3. Mapping host names to addresses in IPv6 can be much more advantageous than with IPv4. The long IPv6 addresses make employing either a DNS server or static mapping using the **ipv6 host** command a more attractive option. A simple **ping** or **telnet/ssh** to an IPv6 address can be more cumbersome and prone to typing errors than with IPv4. As shown in Example 17-7, it is much easier to use a host name. Of course, it helps to have a naming scheme in your organization so that the names are easy to remember. Notice that the host names are not case sensitive.

Example 17-7 *Static Host Name–to–IPv6 Mappings on R1*

```
R1(config)# ipv6 host WinPC 2001:db8:cafe:1705:ddbd:6053:612e:77ab
R1(config)# ipv6 host R2-G00 2001:db8:cafe:3::1
R1(config)# ipv6 host R3-ANY 2001:db8:cafe:4::1 2001:db8:cafe:4::2
R1(config)# end

R1# ping r2-g00

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:3::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/8 ms
R1#

R1# ssh -l admin r3-any

Password:
R3#
```

You confirm the configuration by using the **show hosts** command, as shown in Example 17-8. The **show hosts** command is used to display the default domain name, the style of the name lookup service, a list of name server hosts, and the cached list of host names and addresses.

Example 17-8 show hosts *Command*

```
R1# show hosts
Default domain is not set
Name/address lookup uses domain service
Name servers are 255.255.255.255

Codes: UN - unknown, EX - expired, OK - OK, ?? - revalidate
       temp - temporary, perm - permanent
       NA - Not Applicable None - Not defined

Host                      Port  Flags      Age Type   Address(es)
WinPC                     None  (perm, OK)  7  IPv6   2001:db8:cafe:1705:ddbd:6053:
                                                      612e:77ab
R2-G00                    None  (perm, OK)  0  IPv6   2001:DB8:CAFE:3::1
R3-ANY                    None  (perm, OK)  0  IPv6   2001:DB8:CAFE:4::1
                                                      2001:DB8:CAFE:4::2
R1#
```

Using the **ipv6 host** command is convenient with a limited number of host name–to–IPv6 address mappings. But when the number of mappings is considerable, it might be more desirable to use a DNS server. The DNS resolver in Cisco IOS accepts either an IPv4 address and/or an IPv6 address as a name server. The **ip name-server** command, which specifies the address of one or more DNS servers to use for name and address resolution, is the same command used in IPv4. The syntax for this command is as follows:

```
Router(config)# ip name-server server-address1 [server-address2...server-address6]
```

This command sequence features the following terms:

- *server-address1*: The IPv4 or IPv6 address of a name server
- *server-address2...server-address6* **(optional):** The IP addresses of additional name servers (a maximum of six name servers)

Note If multiple DNS servers are configured, the router usually uses the first server listed. However, depending on the network activity level, the router can query multiple DNS servers listed in the configuration.

In Example 17-9, the **ip name-server** command is configured on R1. Notice that both IPv6 and IPv4 addresses are specified.

Example 17-9 *Specifying the Address of a Name Server*

```
R1(config)# ip name-server 2001:db8:cafe:2::77 192.168.2.77
R1(config)#
```

DNS Query and Response

Resolving domain names in the IPv6 world is really no different than it is in IPv4. For example, say that a user at an IPv6 client types **www.facebook.com** into his browser. The web browser recognizes the request for a name and invokes the client's local resolver to unite the name with an address. Before querying its DNS server, the client's resolver first checks its name cache and local host table. If the name cannot be found in either, the resolver sends the request on to the address of the local DNS server statically or dynamically configured on the client.

It is now the local DNS server's responsibility to resolve this name for the client. This can require a process of several recursive queries and responses between the local DNS server and the root, TLD (top-level domain), and authoritative name servers. After the local DNS server has a response with the IPv6 address of the queried domain name, it sends that information back to the client that originated the request. The client now has the IPv6 address it needs as the destination IP address of the packet.

Using the www.facebook.com example, Example 17-10 shows the DNS query from the client to the DNS server for domain name. When the DNS server has the answer, it sends the client a response with the IPv6 address, which is 2a03:2880:f122:83:face:b00c:0:25de for www.facebook.com. Example 17-11 shows the response from the DNS server.

Example 17-10 *DNS Query for www.facebook.com*

```
Domain Name System (query)
    Flags: 0x0100 Standard query
        0... .... .... .... = Response: Message is a query
        .000 0... .... .... = Opcode: Standard query (0)
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... .0.. .... = Z: reserved (0)
        .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
```

```
   Additional RRs: 0
    Queries
        www.facebook.com: type AAAA, class IN
        Name: www.facebook.com
        [Name Length: 16]
        [Label Count: 3]
        Type: AAAA (IPv6 Address) (28)
        Class: IN (0x0001)
```

Example 17-11 *DNS Response for www.facebook.com*

```
Domain Name System (response)
    Transaction ID: 0xf520
    Flags: 0x8180 Standard query response, No error
        1... .... .... .... = Response: Message is a response
        .000 0... .... .... = Opcode: Standard query (0)
        .... .0.. .... .... = Authoritative: Server is not an authority for domain
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... 1... .... = Recursion available: Server can do recursive queries
        .... .... .0.. .... = Z: reserved (0)
        .... .... ..0. .... = Answer authenticated: Answer/authority portion was not
                               authenticated by the server
        .... .... ...0 .... = Non-authenticated data: Unacceptable
        .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 2
    Authority RRs: 0
    Additional RRs: 0
    Queries
        www.facebook.com: type AAAA, class IN
            Name: www.facebook.com
            [Name Length: 16]
            [Label Count: 3]
            Type: AAAA (IPv6 Address) (28)
            Class: IN (0x0001)
  Answers
        www.facebook.com: type CNAME, class IN, cname star-mini.c10r.facebook.com
            Name: www.facebook.com
            Type: CNAME (Canonical NAME for an alias) (5)
            Class: IN (0x0001)
            Time to live: 1018
```

```
        Data length: 17
        CNAME: star-mini.c10r.facebook.com
    star-mini.c10r.facebook.com: type AAAA, class IN, addr 2a03:2880:f122:83:
    face:b00c:0:25de
        Name: star-mini.c10r.facebook.com
        Type: AAAA (IPv6 Address) (28)
        Class: IN (0x0001)
        Time to live: 43
        Data length: 16
            AAAA Address: 2a03:2880:f122:83:face:b00c:0:25de
```

You can verify the name–to–address mapping by using the **nslookup** (name server lookup) command to query the DNS server, as shown in Example 17-12.

Example 17-12 *Resolving a Domain Name with the* **nslookup** *Command*

```
PC> nslookup
Default Server:  cdns01.comcast.net
Address:  2001:558:feed::1

> set type=AAAA
> www.facebook.com
Server:  cdns01.comcast.net
Address:  2001:558:feed::1

Non-authoritative answer:
Name:    star-mini.c10r.facebook.com
Address:  2a03:2880:f122:83:face:b00c:0:25de
Aliases:  www.facebook.com

> exit
PC>
```

For more information on DNS for IPv6, refer to RFC 3596, *DNS Extensions to Support IP Version 6*.

Happy Eyeballs

Although there has been significant growth in adoption of IPv6, many parts of the Internet are still IPv4-only. An application on a dual-stacked device can experience noticeable delays when unsuccessfully trying to reach a server over IPv6 and then timing out and having to attempt the same connection over IPv4.

With Happy Eyeballs, an application makes a more aggressive connection attempt over both IPv6 and IPv4 simultaneously but preferring IPv6. Happy Eyeballs is an algorithm published in RFC 6555, *Happy Eyeballs: Success with Dual-Stack Hosts*, written by Dan Wing and Andrew Yourtchenko.

When the dual-stacked device has received an IPv6 and IPv4 destination address, the Happy Eyeballs algorithm provides IPv6 with a short head start. The application, such as Chrome or Firefox, attempts a connection first with IPv6. If the connection doesn't complete within 300 ms, the device attempts the connection over IPv4. The first connection that is established is the one that is used. Ultimately, it depends on the host operating system and how the application wants to handle it.

IPv6 Access Control Lists

Configuring IPv6 access control lists (ACLs) is similar to configuring IPv4 ACLs. However, there are some notable differences. This section assumes that you are familiar with configuring IPv4 ACLs.

In IPv4, there are two basic types of ACLs: standard and extended. A standard ACL can permit or deny traffic based only on the source address. An extended ACL provides greater control and can permit or deny traffic based both on the source and destination addresses, as well as protocols and services (TCP/UDP port numbers). IPv6 supports named (versus numbered) ACLs only, equivalent to IPv4 extended named ACLs. An IPv4 ACL and an IPv6 ACL cannot share the same name.

In IPv4, there is an implicit **deny any any** statement at the end of each access list. IPv6 access control lists have a similar **deny ipv6 any any** statement but also includes two other statements by default:

- **permit icmp any any nd-na**
- **permit icmp any any nd-ns**

The IPv6 Neighbor Discovery (ND) process requires the use of the IPv6 network layer service. Therefore, by default, IPv6 ACLs need to implicitly allow IPv6 Neighbor Discovery packets to be sent and received on an interface. This means that both Neighbor Advertisement and Neighbor Solicitation messages must be permitted on the link. In IPv4, Address Resolution Protocol (ARP) messages, which are equivalent to the IPv6 Neighbor Discovery process, are not sent over IPv4. However, IPv6 Neighbor Discovery messages do use the services of IPv6 and must be implicitly permitted in the IPv6 ACLs.

Configuring IPv6 ACLs

Configuring IPv6 ACLs is very similar to configuring IPv4 extended named ACLs. Using the topology in Figure 17-9, say that you want to filter out traffic from the 2001:db8:cafe:4::/64 network, R4's LAN, from entering the 2001:db8:cafe:1::/64 network. You can configure an IPv6 ACL on router R1 to deny this traffic coming in from the R2 router.

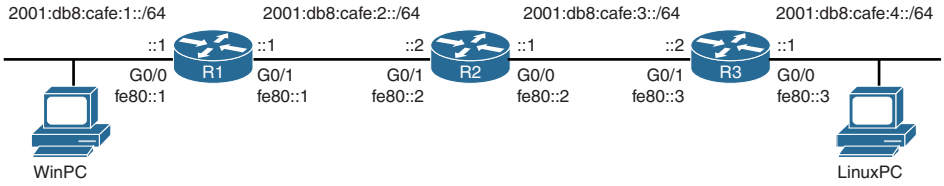


Figure 17-9 *IPv6 Topology for ACLs*

Begin by using the **ipv6 access-list** command on R1, which is similar to the **ip access-list** command, except that it is IPv6-specific. The syntax for this command is as follows:

```
Router(config)# ipv6 access-list access-list-name
```

In Example 17-13, an ACL on R1 is configured with the access list name NO-ACCESS-R3-LAN. In ACL configuration mode, you configure the permit and deny statements. Unlike IPv4 ACLs, IPv6 ACLs do not use a wildcard mask. Instead, the prefix length is used to indicate how much of the IPv6 source or destination address should be matched.

Again, these commands are similar to those in IPv4 but with IPv6-specific options. The ACL command **deny 2001:db8:cafe:4::/64 any** causes R1 to deny any packets with the source IPv6 address containing the 2001:db8:cafe:4::/64 prefix. The next statement is **permit ipv6 any any**, which allows all other IPv6 packets.

Example 17-13 *Configuring an IPv6 ACL on R1*

```
R1(config)# ipv6 access-list NO-ACCESS-R3-LAN
R1(config-ipv6-acl)# deny 2001:db8:cafe:4::/64 ?
  X:X:X:X::X/<0-128>  IPv6 destination prefix x:x::y/<z>
  any                 Any destination prefix
  host                A single destination host

R1(config-ipv6-acl)# deny 2001:db8:cafe:4::/64 any ?
  auth                    Match on authentication header
  dest-option             Destination Option header (all types)
  dscp                    Match packets with given dscp value
  flow-label              Flow label
  fragments               Check non-initial fragments
  hbh                     Match on hop-by-hop option
  log                     Log matches against this entry
  log-input               Log matches against this entry, including input
  mobility                Mobility header (all types)
  mobility-type           Mobility header with type
  routing                 Routing header (all types)
  routing-type            Routing header with type
  sequence                Sequence number for this entry
```

```
 time-range              Specify a time-range
 undetermined-transport  Transport cannot be determined or is missing
 <cr>

R1(config-ipv6-acl)# deny 2001:db8:cafe:4::/64 any
R1(config-ipv6-acl)# permit ipv6 any any
R1(config-ipv6-acl)# exit
R1(config)# interface gigabitethernet 0/1
R1(config-if)# ipv6 traffic-filter NO-ACCESS-R3-LAN in
R1(config-if)#
```

Instead of the **ip access-group** command used in IPv4 for applying an ACL to an interface, IPv6 uses the **ipv6 traffic-filter** command. The **ipv6 traffic-filter** command filters traffic that is forwarded, not originated, by the router. To filter incoming or outgoing IPv6 traffic on an interface, use the **ipv6 traffic-filter** command in interface configuration mode:

```
Router(config-if)# ipv6 traffic-filter access-list-name {in | out}
```

To filter the packets with the IPv6 prefix 2001:db8:cafe:4::/48, the **ipv6 traffic-filter NO-ACCESS-R3-LAN in** command is configured on R1's G0/1 interface facing the R2 router, as shown in Example 17-13.

The **ping** commands in Example 17-14 verify that the ACL is filtering traffic from R3's LANs. The first ping succeeds because the source address of the ping is R3's G0/1 2001:db8:cafe:3::2 address. The second ping fails because this ping is sourced from R3's LAN 2001:db8:cafe:4::1, which is being denied by the ACL on R1.

Example 17-14 *Applying an ACL to an Interface*

```
R3# ping 2001:db8:cafe:1::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:1::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
R3# ping 2001:db8:cafe:1::1 source 2001:db8:cafe:4::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:CAFE:1::1, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:CAFE:4::1
SSSSS
Success rate is 0 percent (0/5)
R3#
```

The IPv6 test character on R3 shows all *S*s, indicating that these packets failed because of the source address, which is usually a result of being blocked by an access list.

Example 17-15 verifies this ACL configuration, using the **show ipv6 access-list** command. Notice the default sequence number associated with each statement. The **show running-config** command is also used to verify the configuration of the ACL.

Example 17-15 *Verifying R3's ACL*

```
R1# show ipv6 access-list
IPv6 access list NO-ACCESS-R3-LAN
    deny ipv6 2001:DB8:CAFE:4::/64 any (10 matches) sequence 10
    permit ipv6 any any (206 matches) sequence 20
R1# show running-config
<partial output>
interface GigabitEthernet0/1
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:2::1/64
 ipv6 traffic-filter NO-ACCESS-R3-LAN in
!
!
ipv6 access-list NO-ACCESS-R3-LAN
 deny ipv6 2001:DB8:CAFE:4::/64 any
 permit ipv6 any any
!
```

Example 17-16 shows an IPv6 ACL configured on R1 to deny FTP traffic from R1's LAN to 2001:db8:cafe:4::/64. Ports for both FTP data (port 20) and FTP control (port 21) need to be blocked. Because the filter is applied inbound on the G0/0 interface on R1, only FTP traffic from the 2001:db8:cafe:1::/64 network is denied.

Example 17-16 *IPv6 ACL Denying FTP Traffic to R3's LAN*

```
R1(config)# ipv6 access-list NO-FTP-TO-R3-LAN
R1(config-ipv6-acl)# deny tcp any 2001:db8:cafe:4::/64 eq ftp
R1(config-ipv6-acl)# deny tcp any 2001:db8:cafe:4::/64 eq ftp-data
R1(config-ipv6-acl)# permit ipv6 any any
R1(config-ipv6-acl)# exit
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ipv6 traffic-filter NO-FTP-TO-R3-LAN in
R1(config-if)#
```

Note IPv6 ACLs allow you to filter on information in the IPv6 header and extension headers. This is platform-dependent and beyond the scope of this book.

Note IPv6 also includes support for other types of ACLs, including reflective, time-based, and zone-based ACLs. These are beyond the scope of this book. For information on these and other types of IPv6 ACLs, refer to the chapter "Implementing Traffic Filters and Firewalls for IPv6 Security" in *Cisco IOS IPv6 Configuration Guide,* http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-sec-trfltr-fw.html.

Transition Technologies

The worlds of IPv4 and IPv6 will coexist for the foreseeable future. How and when an organization migrates to IPv6 depends on its specific situation. Many service providers and larger enterprise networks were early adopters of IPv6. They began their planning, training, and transition strategy years ago. Because of the lack of IPv4 address space, many organizations, including mobile providers and those wanting a global Internet presence that reaches the customers of today and tomorrow, have already begun actively deploying IPv6.

Note Cisco Systems provides IPv6 adoption statistics at 6lab.cisco.com/stats. Google also provides statistics about IPv6 adoption on the Internet, at www.google.com/intl/en/ipv6/statistics.

Ideally, IPv6 should be run natively wherever possible, with IPv6 devices communicating with each other directly over IPv6 networks. However, the move from IPv4 to IPv6 will happen over time. The Internet Engineering Task Force (IETF) has developed several transition techniques to accommodate a variety of IPv4-to-IPv6 scenarios, including dual stack, translation, and tunneling. Dual stack is discussed earlier in this chapter, this section covers translation techniques, and tunneling is covered later in this chapter.

This section focuses on NAT64, with a brief discussion of other translation methods. Appendix A, "Configuring NAT64 and IPv6 Tunnels," includes additional information on NAT64, including a configuration example.

The section "Tunneling IPv6," later in this chapter, provides an overview of tunneling and different types of tunnels for IPv6. Appendix A includes detailed information on manual, 6to4, and ISATAP tunnels. Appendix A, also includes configuration examples for each of these tunneling methods.

Note Configuration examples for NAT64 and different tunneling methods are also provided in Appendix A. We present the information this way so we can focus this chapter on the transition technologies without getting lost in the details. If you are interested in more detail and configuration examples, see Appendix A.

Translation with NAT64

Network address translation (NAT) is a familiar method in IPv4 that is commonly used to translate between private (RFC 1918) addresses and public IPv4 address space. Address Family Translation (AFT), or simply translation, provides communication between IPv6-only and IPv4-only hosts and networks. AFT performs IP header and address translations between these two network layer protocols. Like other transition methods, translation is not a long-term strategy, and the ultimate goal should be native IPv6. However, for those that need translation at least for now, it offers two major advantages over tunneling:

- Translation provides a means for gradual and seamless migration to IPv6.
- Content providers can provide services transparently to IPv6 Internet users.

Note Tunneling IPv6 is discussed in the next section.

NAT64 transparently provides access between IPv6-only and IPv4-only networks. NAT64 is the replacement for the deprecated Network Address Translation–Protocol Translation (NAT-PT), as documented in RFC 6144, *Framework for IPv4/IPv6 Translation*. NAT-PT is similar to the NAT mechanism in IPv4 that is commonly used for translating private (RFC 1918) IPv4 addresses to public IPv4 addresses and vice versa. NAT-PT is defined in RFC 2766, *Network Address Translation–Protocol Translation (NAT-PT)*, and obsoleted by RFC 4966, *Reasons to Move the Network Address Translator–Protocol Translator (NAT-PT) to Historic Status*. Cisco highly recommends not using NAT-PT.

NAT-PT has been deemed deprecated by the IETF because of its tight coupling with DNS and general limitation in translation. These reasons are documented in RFC 4966. The IETF has proposed NAT64 as the successor to NAT-PT.

NAT64 is a mechanism for IPv4-to-IPv6 transition and IPv4/IPv6 coexistence. Together with DNS64, the primary purpose of NAT64 is to allow an IPv6-only client to initiate communications to an IPv4-only server, as shown in Figure 17-10. NAT64 can also be used for IPv4-only clients initiating communications with IPv6-only servers using static or manual bindings.

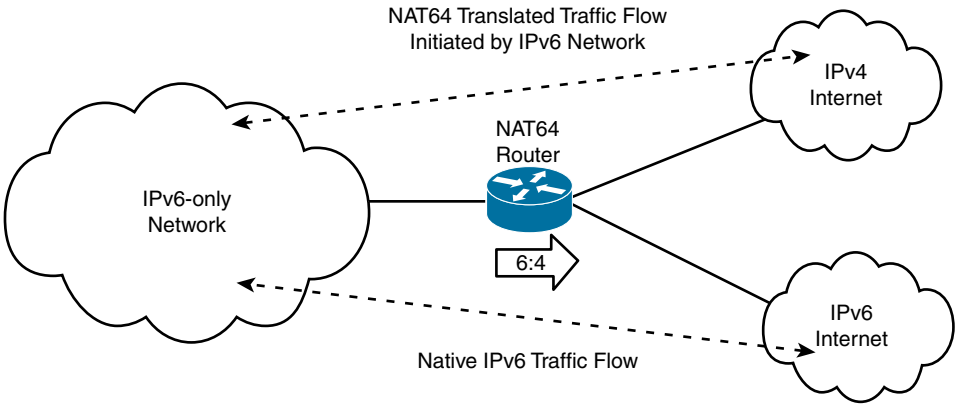


Figure 17-10 *IPv6-Only Network Accessing IPv4 and IPv6 Internet*

NAT64 is described in RFC 6146, *Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers*. Similar to stateful NAT used for IPv4, stateful NAT64 creates or modifies bindings between IPv6 and IPv4 while performing translation. NAT64 performs translation between IPv6 and IPv4 by performing the following:

- IP header translation between the two protocols, using an algorithm defined in RFC 6145, *IP/ICMP Translation Algorithm*
- IP address translation between the two protocols using an algorithm defined in RFC 6052, *IPv6 Addressing of IPv4/IPv6 Translators*

Note Stateless NAT64 is also an option. Unlike stateful NAT64, stateless NAT64 does not maintain any sort of bindings or session state while performing the translation. This chapter covers only stateful NAT64.

There are three components to NAT64:

- **NAT64 prefix:** This is any /32, /40, /48, /56, /64, or /96 prefix used with a converted IPv4 address to transmit the packet over the IPv6-only network. The NAT64 prefix can be a network-specific prefix (NSP) or a well-known prefix (WKP). An NSP is assigned by an organization and is usually a subnet from the organization's IPv6 prefix. The WKP for NAT64 is 64:ff9b::/96. If an NSP is not specified or configured, NAT64 uses the WKP to prepend the converted IPv4 address. The NAT64 prefix is also referred to as Pref64::/n.
- **DNS64 server:** The DNS64 server functions as a normal DNS server for IPv6 AAAA records but also attempts to locate an IPv4 A record when an AAAA record is not available. If an A record is located, DNS64 converts the IPv4 A record into an IPv6

AAAA record, using the NAT64 prefix. This gives the impression to the IPv6-only host that it can communicate with a server using IPv6.

- **NAT64 router:** The NAT64 device advertises the NAT64 prefix into the IPv6-only network and performs the translation between the IPv6-only and IPv4-only networks.

Traffic Initiated from IPv6-Only Clients to IPv4-Only Servers

Figure 17-11 shows a scenario where clients in an IPv6-only network, 2001:db8:cafe::/48, communicate with IPv4-only servers using NAT64. This is the most common scenario for use of NAT64. The following steps illustrate this process:

Step 1. Host A is an IPv6-only host that wants to communicate with the server www.example.com. This triggers a DNS query (AAAA: www.example.com) to the DNS64 server. The DNS64 is a key component in this process. A DNS64 server is a DNS server for both IPv6 and IPv4. It creates the illusion for the client that IPv4 servers can be reached using an IPv6 address. Figure 17-12 illustrates the DNS64 operation in steps 1 through 7 more closely. The steps in Figure 17-12 are identical to the steps in Figure 17-11. Both diagrams can be used to help you better understand the DNS64 process.

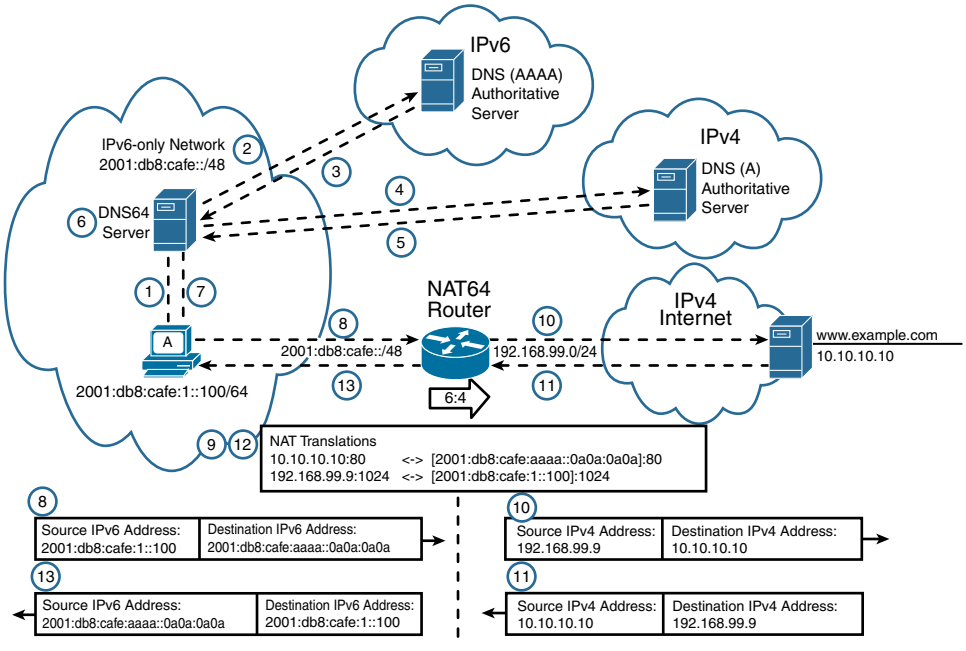


Figure 17-11 *NAT64 Router Translating IPv6 Traffic to IPv4 and Returning Traffic*

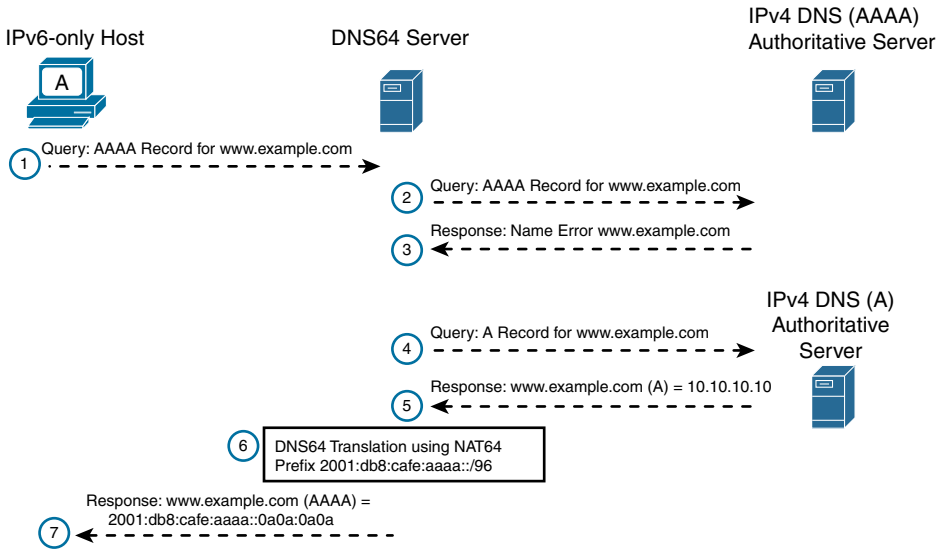


Figure 17-12 *DNS64 Operation*

Host A sends a DNS query (AAAA: www.example.com) to the DNS64 server. As far as host A is concerned, this is a normal DNS AAAA query for an IPv6 server.

Step 2. The DNS64 server receives the DNS AAAA query from host A. In an attempt to resolve the domain name, the DNS64 server sends a query to the DNS AAAA authoritative server for www.example.com.

Step 3. The IPv6 DNS AAAA authoritative server returns a response indicating that it does not have an AAAA resource record for www.example.com.

Step 4. On receiving an empty answer (name error) response to the AAAA query, this triggers the DNS64 server to send an A query (A: www.example.com) to the IPv4 DNS A authoritative server.

Step 5. The IPv4 DNS A authoritative server does have an A resource record for www.example.com and returns a response with the IPv4 address for the server (A: www.example.com 10.10.10.10).

Step 6. The DNS64 server receives the IPv4 address from the DNS A authoritative server and synthesizes an AAAA record by prefixing the address with its NAT64 prefix, 2001:db8:cafe:aaaa::/96, and converting the IPv4 address to hexadecimal, 0a0a:0a0a.

Figure 17-13 illustrates the DNS64 synthesis of the IPv4 A record into an IPv6 AAAA record. The DNS64 server receives the IPv4 A record 10.10.10.10 for www.example.com. The 32-bit IPv4 address is converted into its hexadecimal equivalent, 0a0a:0a0a. The DNS64 server prefaces the address with the prefix 2001:db8:cafe:aaaa::/96, resulting in a DNS64-synthesized

AAAA resource record of 2001:db8:cafe:aaaa::0a0a:0a0a for www.example.com. This address is then used by host A as the destination IPv6 address for reaching the www.example.com server.

Note A DNS server does not perform DNS64 services automatically. The DNS server must support and be specifically configured for DNS64 operations.

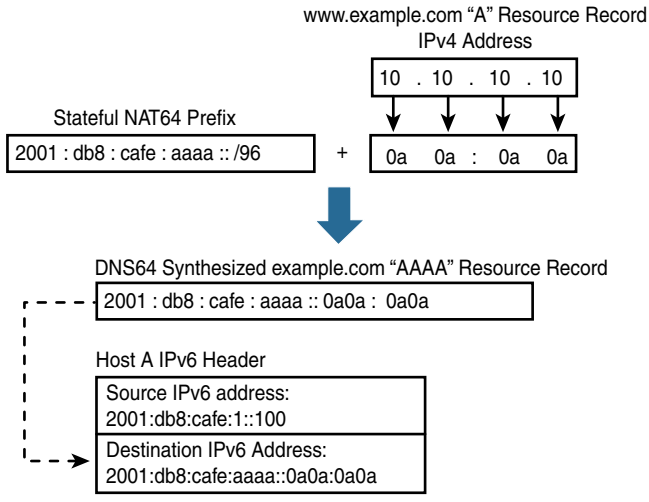


Figure 17-13 *DNS64 Synthesis of an A Record into an AAAA Record*

Note The NAT64 prefix can be a network-specific prefix (NSP) or a well-known prefix (WKP).

Step 7. The DNS64 server sends an AAAA response to host A with the address 2001:db8:cafe:aaaa::0a0a:0a0a for www.example.com.

Step 8. The synthesized AAAA record is completely transparent to host A. To host A, it appears as if www.example.com is reachable over the IPv6 network and Internet. Host A now has the address information necessary to transmit IPv6 packets to www.example.com with the following:

- **IPv6 destination address:** 2001:db8:cafe:aaaa::0a0a:0a0a
- **IPv6 source address:** 2001:db8:cafe:1::100

Step 9. The NAT64 router receives the IPv6 packet sent by host A on its NAT64-enabled interface. The NAT64 router is configured with the stateful NAT64 prefix 2001:db8:cafe:aaaa::/96. The router attempts to forward this IPv6 packet. If the first 96 bits do not match the configured stateful NAT64 prefix,

the packet is forwarded untranslated using normal IPv6 routing. If the packet's destination address does match the stateful NAT64 prefix, the packet undergoes the following translation:

- The IPv6 header is translated into an IPv4 header.
- The IPv6 destination address is translated into an IPv4 address by removing the IPv6 stateful NAT64 prefix 2001:db8:cafe:aaaa::/96. The lower 32 bits of the IPv6 address, 0a0a:0a0a, are represented as the dotted-decimal IPv4 address 10.10.10.10.
- The IPv6 source address is translated into an IPv4 address using the configured IPv4 address pool. Depending on the NAT64 configuration, this can be either a 1:1 address translation or can use IPv4 address overloading. This is similar to NAT for IPv4. In this scenario, host A's source IPv6 address is translated to the 192.168.99.9 IPv4 address.
- Stateful NAT64 IP address translation states are created for both the source and destination addresses, as shown in the NAT Translations table below the NAT64 router in Figure 17-11. These states are created the first time the translation is performed on the packet. This state is maintained for subsequent packets in the flow. The state ends when the traffic and the state maintenance timer expire.

Note This scenario uses private address space, but in the real world, publicly reachable addresses are necessary to ensure reachability. Also, if you only have a limited pool of publicly reachable IPv4 addresses, this might not be the optimal solution.

Step 10. After the NAT64 translation, the translated IPv4 packet is forwarded using the normal IPv4 route lookup process. In this scenario, the IPv4 destination address 10.10.10.10 is used to forward the packet.

Step 11. The www.example.com server at 10.10.10.10 replies, and the reply is ultimately received by the NAT64 router.

Step 12. The NAT64 router receives the IPv4 packet from the www.example.com server on one of its NAT64-enabled interfaces. The router examines the IPv4 packet to determine whether a NAT64 translation state exists for the IPv4 destination address. If a translation state does not exist, the packet is discarded. If a translation state does exist for the IPv4 destination address, the NAT64 router performs the following tasks:

- The IPv4 header is translated into an IPv6 header.
- The IPv4 source address is translated into an IPv6 source address using the existing NAT64 translation state. In this scenario, the source address is translated from an IPv4 address of 10.10.10.10 to the IPv6 address 2001:db8:cafe:aaaa::0a0a:0a0a. The destination address is translated from an IPv4 address 192.168.99.9 to the IPv6 address 2001:db8:cafe:1::100.

Step 13. After the translation, the IPv6 packet is forwarded using the normal IPv6 route lookup process.

The DNS64 server used with a NAT64 router creates a transparent environment for IPv6-only clients that need to communicate with IPv4 servers. The DNS64 server misleads the IPv6 hosts into thinking that the IPv4 destination is an IPv6 address. The DNS64 server has the capability of acting as a typical IPv6 DNS server and/or as a DNS64 translator using a NAT64 prefix.

Traffic Initiated from IPv4-Only Clients to IPv6-Only Servers

Figure 17-14 shows a scenario where clients in an IPv4-only network communicate with an IPv6-only server using NAT64, which is further detailed in steps 1 through 9. The goal is to provide access to IPv6 services transparently to the IPv4 clients. In this scenario, the DNS64 server is not required. Static mapping between the IPv6 and IPv4 addresses is configured on the NAT64 router.

Note This scenario is unlikely for the foreseeable future. Most servers that are enabled for IPv6 are also IPv4-capable. It is more likely that IPv6 servers will be dual stacked for quite some time. IPv6-only servers are becoming more common, with many large companies running IPv6 only (that is, supporting a single protocol stack, which is the protocol of the future).

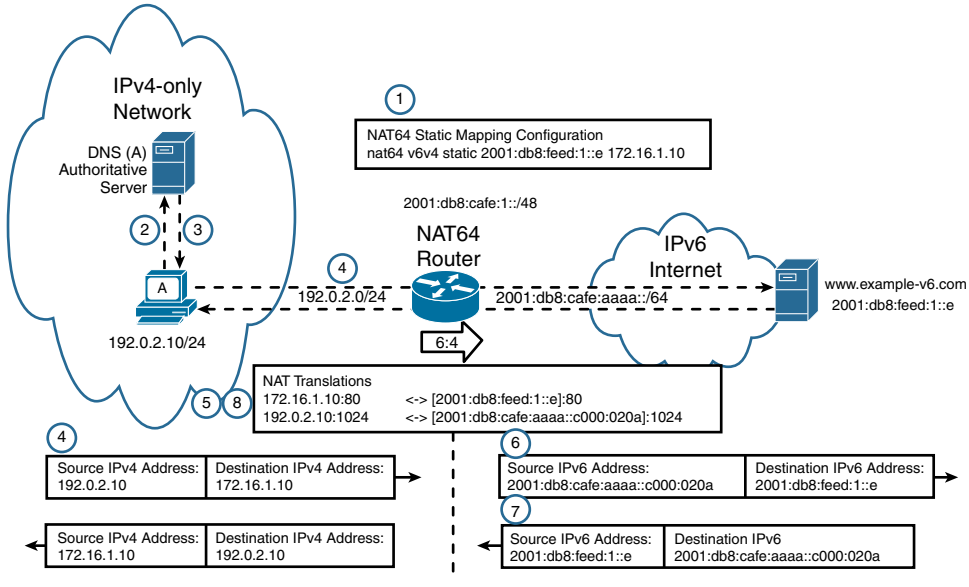


Figure 17-14 *NAT64 Router Translating IPv4 Traffic to IPv6 and Returning Traffic*

The following steps are shown in Figure 17-14:

Step 1. The first step is to configure IPv6–to–IPv4 static mapping to provide access to the IPv6 server 2001:db8:feed:1::e from the IPv4 address 172.16.1.10. Also, the IPv4 address 172.16.1.10 needs to be registered as a DNS resource record for www.example-v6.com on the DNS server. The static NAT64 mapping is created using this command:

```
NAT64-Router(config)# nat64 v6v4 static 2001:db8:feed:1::e 172.16.1.10
```

Step 2. Host A is an IPv4-only host that wants to communicate with the server www.example-v6.com. This triggers a DNS query (A: www.example-v6.com) to Host A's IPv4 DNS authoritative server.

Step 3. The DNS server responds with an A resource record for www.example-v6.com, 172.16.1.10.

Step 4. Host A now has the address information necessary to transmit IPv4 packets to www.example-v6.com:

- **IPv4 destination address:** 172.16.1.10
- **IPv4 source address:** 192.0.2.10

Step 5. The NAT64 router receives the IPv4 packet on its NAT64-enabled interface and performs the following tasks:

- The IPv4 header is translated into an IPv6 header.
- The IPv4 destination address is translated into an IPv6 address using the existing NAT64 translation state created by the static configuration in step 1. The destination IPv4 address 172.16.1.10 is translated to the IPv6 destination address 2001:db8:feed:1::e.
- The IPv4 source address is translated into an IPv6 address by adding the stateful NAT64 prefix 2001:db8:cafe:aaaa::/96 to the IPv4 address. This results in the IPv6 source address 2001:db8:cafe:aaaa::c000:020a. (c000:020a is the hexadecimal equivalent of 192.0.2.10.)

Step 6. After the translation, the IPv6 packet is routed using the normal IPv6 routing process. The packet is ultimately routed to the www.example-v6.com server at 2001:db8:feed:1::e.

Step 7. The server www.example-v6.com replies with a packet destined for Host A.

Step 8. The NAT64 router receives the IPv6 packet sent by the IPv6 server on its NAT64-enabled interface and performs the following tasks:

- The IPv6 header is translated into an IPv4 header.
- The IPv6 source address is translated into an IPv4 address by removing the NAT64 2001:db8:feed:1::e and converting it to 172.16.1.10.

- The IPv6 destination address is translated into an IPv4 address by removing the IPv6 stateful NAT64 prefix 2001:db8:cafe:aaaa::/96. The lower 32 bits of the IPv6 address, c000:020a, are represented as the dotted-decimal IPv4 address 192.0.2.10.

Step 9. After the translation, the NAT64 router forwards the packet to 192.0.2.10, using the normal IPv4 routing process.

Similar to the previous scenario, transparent communication is established between the IPv4-only client and the IPv6-only server using stateful NAT64. The configurations are similar except for the static mapping command discussed in step 1. In the case where the IPv6-only client initiates communications with the IPv4 server, the DNS64 server plays a significant role. That is not the case in this scenario, where static mapping of the IPv6-to-IPv4 address is required on the NAT64 router with the addition of DNS resource records for IPv4.

Note Appendix A contains additional configuration information for NAT64, including configuration examples.

Other Translation Techniques

Network Address Translation IPv6 to IPv4 (NAT64) is one of the most common IPv4–to–IPv6 translation methods, but it is not the only one. The IETF has defined additional translation techniques that give devices in IPv6-only networks the ability to communicate with devices in IPv4-only networks. The following are some of the other translation techniques available:

- **Mapping of Address and Port (MAP):** MAP is an IPv6 transition and interworking technology that leverages the natural aggregation capability of the address and port space to allow IPv4 addresses to be translated or encapsulated in IPv6, without requiring a stateful translator on the service provider network. MAP is a production-quality, deployable IPv6 transition mechanism, which allows service providers to share scarce IPv4 address resources while deploying an IPv6-only provider network. Two flavors of MAP exist today: MAP-T (which uses translation) and MAP-E (which uses encapsulation). The key benefit of MAP is its stateless implementation on the service provider router, which allows it to scale proportionally to the number of aggregated prefixes and the traffic volume instead of based on the number of end user connections or states. MAP-T is defined in RFC 7599, *Mapping of Address and Port Using Translation (MAP-T)*, and RFC 7597, *Mapping of Address and Port Using Encapsulation (MAP-E)*.
- **Transport Relay Translation (TRT):** TRT is similar to NAT-PT, using a TRT system to relay traffic between IPv4 and IPv6 domains. TRT enables IPv6-only hosts to exchange TCP and UDP traffic with IPv4-only hosts. TRT is defined in RFC 3142, *An IPv6–to–IPv4 Transport Relay Translator*.

- **IPv6 Rapid Deployment (6rd):** Like 6to4 tunnels, 6rd utilizes stateless IPv6-in-IPv4 encapsulation to transmit an IPv4-only network infrastructure. Unlike with 6to4 tunnels, a 6rd service provider uses one of its own IPv6 prefixes in place of a fixed 6to4 prefix. 6rd is defined in RFC 5569, *IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)*.
- **Dual-Stack Lite (DS-Lite):** Service providers use DS-Lite to connect their customers using private IPv4 addresses to the IPv4 Internet, as shown in Figure 17-15. Between the IPv4 customer and the service provider is an IPv6 link. When a customer sends an IPv4 packet using a private (RFC 1918) source IPv4 address, the packet is encapsulated inside an IPv6 packet by the router and sent over the IPv6 network. At the other end of the tunnel, the IPv4 packet is decapsulated. The private IPv4 source address is then translated to a public IPv4 address, using IPv4 NAT. DS-Lite and 6rd are tunneling mechanisms that typically work in conjunction with a translation mechanism.

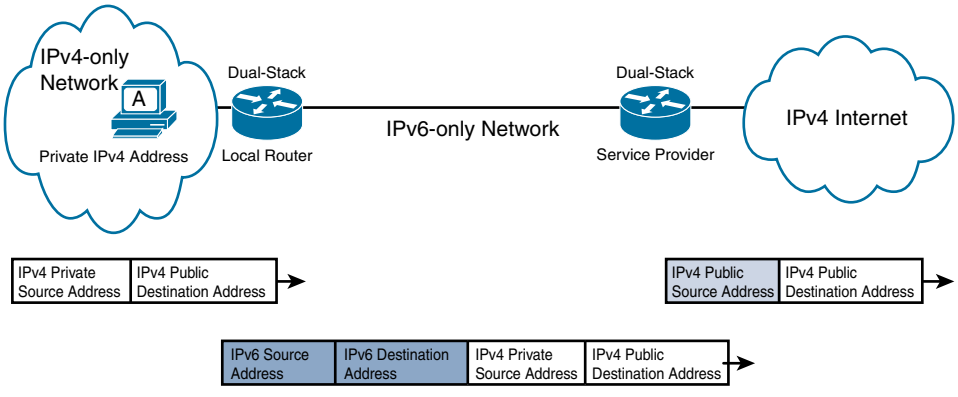


Figure 17-15 *Dual-Stack Lite*

Tunneling IPv6

Another type of IPv4–to–IPv6 transition mechanism is tunneling. Like other transition methods, tunneling should be considered a temporary solution until native IPv6 can be employed. A tunnel is nothing more than encapsulating one IP packet inside another. A tunnel can be an IPv4 packet encapsulated in another IPv4 packet or, for that matter, any network layer protocol over another network layer protocol. One of the challenges in integrating IPv6 into the current IPv4 networks is the ability to transport IPv6 packets over IPv4-only networks. One way to do this is to use a tunnel or, in IPv6, what is known as an *overlay* tunnel. Overlay tunnels encapsulate IPv6 packets in IPv4 packets for delivery across an IPv4 infrastructure.

Figure 17-16 shows an example of two IPv6-only networks, or "IPv6 islands," separated by an IPv4-only network.

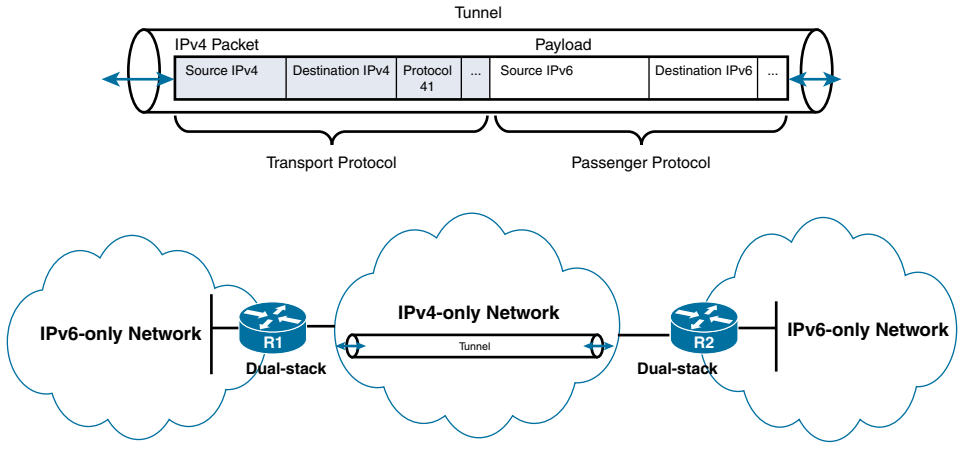


Figure 17-16 *Tunnel Connecting Islands of IPv6 Networks*

Tunneling is a technique that allows devices in isolated IPv6 networks to send IPv6 packets over the IPv4 network. A tunnel has two types of protocols: a transport protocol and a passenger protocol. Figure 17-16 illustrates these two protocols in an IPv6 overlay tunnel:

- **Transport protocol:** IPv4 is the transport protocol, in which the tunnel is created. Protocol 41 in the IPv4 header indicates that the encapsulated data portion is an IPv6 packet.
- **Passenger protocol:** IPv6 is the passenger protocol, encapsulated in the tunnel and carried through the tunnel.

A tunnel involves two devices, the endpoints of the tunnel, and a management protocol to manage the tunnel. These are the three components of a tunnel:

- **Tunnel entry point:** A tunnel entry point is where the encapsulation of the IPv6 packet (passenger protocol) inside an IPv4 packet (transport protocol) occurs. A tunnel entry point must be a dual-stacked device that receives an IPv6 packet and encapsulates it inside an IPv4 packet for transport over the IPv4 network.
- **Tunnel exit point:** The tunnel exit point is another dual-stacked device and is where the IPv4 packet is received. The IPv6 packet is then decapsulated for transport over the IPv6 network.

- **Tunnel management:** Tunnel management is done by the tunnel entry and exit points and deals with tunnel encapsulation/decapsulation, addressing, maximum transmission unit (MTU), fragmentation, and error processing.

Tunneling can involve any combination of routers and hosts, depending on the endpoints (entry and exit points) of the tunnel. Figure 17-17 illustrates the three scenarios that can exist:

- **Host-to-host:** Isolated dual-stacked hosts are separated by an IPv4-only network. The hosts, acting as tunnel endpoints, create a tunnel for transporting IPv6 packets between the two hosts.
- **Host-to-router:** The tunnel endpoints are a host and a router connected by an IPv4-only network. At one end of the tunnel is a dual-stacked host, and at the other end is a dual-stacked router. The router sends and receives the IPv6 packets natively over its IPv6 network. When communicating with the host, the IPv6 packets are encapsulated within an IPv4 packet and sent over the tunnel, where the host receives and processes the IPv6 packet.
- **Router-to-router:** Both tunnel endpoints are dual-stacked routers. A dual-stacked router can establish a tunnel with another dual-stacked router to interconnect islands of IPv6 hosts.

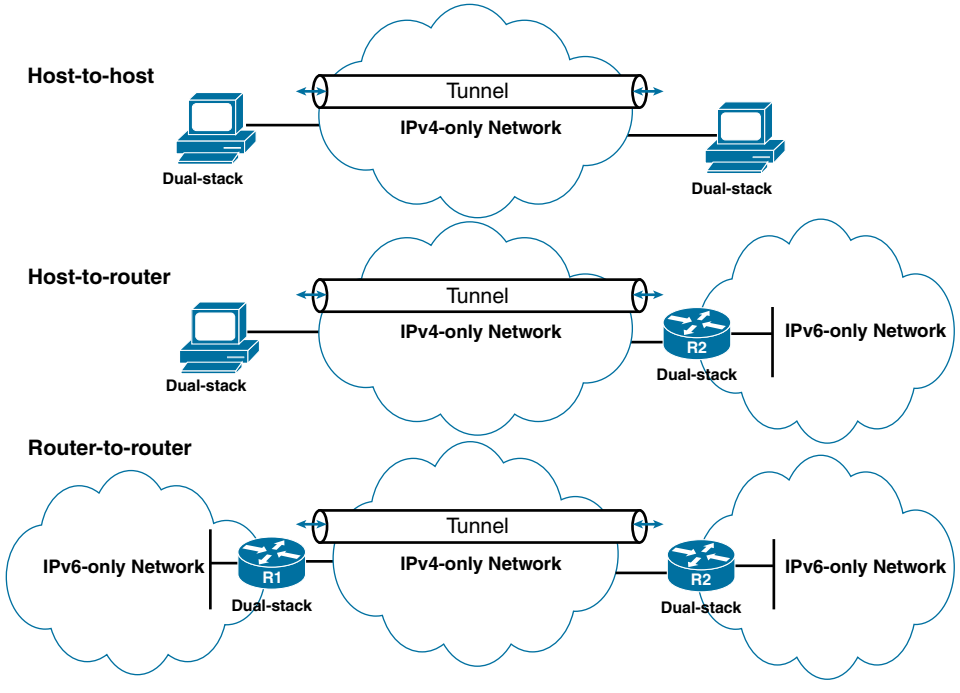


Figure 17-17 *IPv6 Tunnel Endpoints*

The IETF has defined several protocols and techniques for establishing tunnels between dual-stacked nodes. The number of tunneling options can be confusing and even a bit overwhelming. This chapter provides an overview for a better understanding of the similarities and differences with these different types of tunnels. Basically, all the tunnels do the same thing: transport an IPv6 packet inside an IPv4 packet between the two endpoints of the tunnel. After that, the choice comes down to the following:

- How many tunnels need to be configured? If there are just a few, manual tunnels can be fine; otherwise, automatic tunnels might be the better option.
- Are there any specific requirements that would make one tunnel type more desirable than another—such as the need to tunnel non-IP packets or multicast traffic?

The following are several tunneling types for IPv6:

- **Manual tunnel:** A manual tunnel is equivalent to a point-to-point link. The primary use is for stable connections that require regular secure communication between two edge routers (or between an end system and an edge router) or for connection to remote IPv6 networks.
- **6to4 tunnel:** The key difference with 6to4 or automatic 6to4 tunnels is that the tunnel is not point-to-point but a point-to-multipoint tunnel. The destination IPv4 address of the tunnel is determined from the destination IPv6 address of the packet. 6to4 tunnels require a relationship between the IPv6 prefix or network address and IPv4 tunnel addresses. The IPv6 address is reverse engineered from the IPv4 tunnel address using the format 2002:*tunnel-IPv4-address*::/48. (2002::/16 is a reserved prefix for 6to4 tunneling.) This allows for the creation of a single tunnel that has multiple destinations—a point-to-multipoint tunnel. 6to4 tunnels are defined in RFC 3056, *Connection of IPv6 Domains via IPv4 Clouds*.
- **ISATAP (Intra-Site Automatic Tunnel Addressing Protocol) tunnel:** ISATAP is designed for transporting IPv6 packets within a site where a native IPv6 infrastructure is not yet available, such as when sparse IPv6 hosts are deployed for testing. ISATAP is defined in RFC 5214, *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)*.

Note See Appendix A for detailed information on manual, 6to4, and ISATAP tunnels. Appendix A also includes configuration examples for each of these tunneling methods.

- **GRE (Generic Routing Encapsulation) tunnel:** Similar to a manual tunnel, a GRE tunnel is a point-to-point tunnel for secure communication between two endpoints but it can carry protocols other than IP. GRE tunnels allow other protocols besides IP to be the passenger protocol, such as IS-IS (Intermediate System–to–Intermediate System).

The configuration of GRE tunnels is similar to configuring manual tunnels. GRE is defined in RFC 2784, *Generic Routing Encapsulation (GRE)*. GRE adds an additional header in the tunneling process and is the default tunneling type for the Cisco IOS **tunnel mode** command.

- **IPv4-compatible tunnel:** IPv4-compatible tunnels do not scale well for large networks. Cisco does not recommend this tunnel type.
- **IPv6 Rapid Deployment (6rd) tunnels:** A 6rd tunnel is an extension of the 6to4 feature. 6rd allows an ISP or service provider to offer IPv6 service to customers over its IPv4 network. The main differences between 6rd and 6to4 tunneling are as follows:
  - 6rd does not require addresses to have a 2002::/16 prefix; therefore, the prefix can be from the service provider's own address block.
  - Not all 32 bits of the IPv4 destination address need to be carried in the IPv6 payload header. The IPv4 destination address is obtained from a combination of bits in the payload header and information configured in the router.
  - 6rd is defined in RFC 5569, *IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)*.
- **Teredo (RFC 4380) tunnel:** Teredo, also known as shipworm, allows two dual-stacked devices to send IPv6 packets when one is located behind an IPv4 NAT. Teredo uses UDP port 3544 to communicate with Teredo servers, which are used as dispatchers between Teredo clients and Teredo relays. Although Teredo was proposed by Microsoft, there is a version for Linux known as Miredo. Teredo is defined in RFC 4380, *Teredo: Tunneling IPv6 over UDP Through Network Address Translations (NATs)*.

Table 17-4 provides an overview of several of the tunneling types discussed in this section.

GRE, 6rd, and Teredo are all beyond the scope of this book. For more information on implanting tunneling for IPv6, read the following:

- *Cisco Self-Study: Implementing Cisco IPv6 Networks* by Regis Desmeules
- *Implementing Tunneling for IPv6*, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-tunnel.html

Table 17-4 *Types of IPv6 Tunnels*

Tunneling Type	Tunnel Mode Command	Tunnel Source	Tunnel Destination	Interface Prefix or Address	Notes
Manual	**ipv6ip**	An IPv4 address or a reference to an interface on which IPv4 is configured. This can be the IPv4 address of the physical interface or a loopback interface.	An IPv4 address.	An IPv6 address.	Can carry IPv6 packets only.
GRE	**gre ip**		An IPv4 address.	An IPv6 address.	Can carry IPv6, Connectionless Network Service (CLNS), and many other types of packets. Note that IPv4-compatible uses the ::/96 prefix. Cisco recommends not using this tunnel type.
IPv4-compatible	**ipv6ip auto-tunnel**		Not required. These are all point-to-multipoint tunneling types.	Not required. The interface address is generated as ::*tunnel-source*/96.	
6to4	**ipv6ip 6to4**		The IPv4 destination address is calculated on a per-packet basis, using the IPv6 destination address.	An IPv6 address.	Site uses the addresses from the 2002::/16 prefix. Point-to-multipoint tunnels can be used to connect isolated IPv6 sites.
ISATAP	**ipv6ip isatap**		—	An IPv6 prefix in modified EUI-64 format. The IPv6 address is generated from the prefix and the tunnel source IPv4 address.	Point-to-multipoint tunnels can be used to connect systems within a site. Allows individual dual-stacked hosts within a site to communicate. Sites can use any IPv6 unicast addresses.

Conclusion

All transition methods, translation or tunneling, should be considered temporary solutions until native IPv6 can be employed. ISPs are increasingly providing native IPv6 connectivity. Before implementing one of these options, find out from your ISP if it is providing native IPv6. If it is not, ask when it will be providing this service or see if there is another ISP in your area that does.

ARIN's IPv6 information states, "The number of IPv6-only Internet users has never been higher, and without IPv6, you run the risk of not being visible to them, or the entire Internet." The simple fact is that we have run out of IPv4 addresses. IPv6 is required to support the escalating numbers of mobile devices, IoT devices, and those in areas of the world just beginning to connect to the Internet.

Vint Cerf, recognized as one of the fathers of the Internet, in his talks about the origins of the Internet, refers to IPv4 as "the experimental version" and says that we need to move to "the production version" of the Internet, running IPv6. Knowing the past and seeing where the Internet is headed, Vint Cerf has been a long-time promoter of IPv6, stating, "Get your v6 in place so that you can run the 21st century version of the Internet!"

Summary

This chapter discusses the following topics common to many deployments:

- IPv6 address plan considerations
- Configuring IPv6 VLANs
- Choosing a first-hop redundancy protocol for IPv6
- Dual stack
- DNS for IPv6
- ACLs
- Translation protocols and tunneling

This chapter discusses several points to consider when creating an address plan including:

- Keep it simple
- Use an existing IP address schema or create a new one
- Embed information to help operations
- Think about the address plan structure and subnetting
- Consider prefix size and aggregation
- Plan for network growth and flexibility

Part of creating an address schema includes providing a hierarchical structure and embedding information within the Subnet ID, such as location, building, and VLAN ID.

Configuring IPv6 VLANs on a Layer 3 switch is much like configuring IPv4 VLAN information. There are differences, however. VLAN interfaces typically act as the default gateway on the VLAN. An IPv6 VLAN interface has the same function but serves the additional purpose of sending ICMPv6 Router Advertisement messages on the LAN, similar to a traditional router's IPv6 interface. End devices need the information in the RA for the default gateway address and dynamic address determination.

Cisco IOS offers three FHRP protocols to allow for transparent failover at the first-hop IP router for both IPv4 and IPv6:

- Hot Standby Router Protocol (HSRP)
- Virtual Router Redundancy Protocol (VRRP)
- Gateway Load Balancing Protocol (GLBP)

In addition, ICMPv6 Router Advertisement messages can provide failover but have certain drawbacks.

A dual-stacked device has complete support for both IPv4 and IPv6. When you communicate with an IPv4 device, it behaves like an IPv4-only device. When you communicate with an IPv6 device, it acts like an IPv6-only device. On dual-stacked devices with an application that supports both IPv4 and IPv6 protocols, the process is almost identical.

DNS is used in both IPv4 and IPv6 to do name-to-address mappings. Like IPv4, IPv6 also supports the reverse mapping of IPv6 addresses to DNS names. In IPv4, a DNS A resource record maps a domain name to an IPv4 address. Similarly, in IPv6, the AAAA (quad-A) resource record maps a domain name to an IPv6 address.

With Happy Eyeballs, the application makes a more aggressive connection attempt over both IPv6 and IPv4 simultaneously, preferring IPv6. When a dual-stacked device receives an IPv6 or IPv4 destination address, the Happy Eyeballs algorithm provides IPv6 with a short head start. The application, such as Chrome or Firefox, attempts a connection first with IPv6. If the connection doesn't complete within 300 ms, the device attempts the connection over IPv4. The first connection that is established is the one that is used.

As shown in the examples in this chapter, configuring IPv6 ACLs is very similar to configuring IPv4 ACLs. As with IPv4, IPv6 access control lists have an implicit **deny ipv6 any any** statement at the end of each sequence of statements. Because the IPv6 Neighbor Discovery process requires the use of the IPv6 network layer, two other access list statements are also implicitly added to the end of each ACL:

- **permit icmp any any nd-na**
- **permit icmp any any nd-ns**

The **ipv6 access-list** command is used to configure IPv6-specific ACLs. IPv6 ACLs cannot be numbered; they can only be configured as named access lists. The **ipv6 traffic-filter** interface command is used to apply an IPv6 ACL to an interface.

Although several mechanisms have been established for the transition and coexistence of IPv4 and IPv6, native IPv6 infrastructures should be preferred and should be the ultimate goal. Although dual-stacked networks and tunneling have limitations, they are far less restrictive than NAT64. NAT64 should be used only when native IPv6 or other transitioning methods are not possible.

NAT64 is the recommended mechanism for IPv4–to–IPv6 transition and IPv4/IPv6 coexistence. Using DNS64, NAT64 allows IPv6-only clients to initiate communications to IPv4-only servers and IPv4-only clients to initiate communications with IPv6-only servers using static or manual bindings.

The DNS64 server functions as a normal DNS server for IPv6 AAAA records. If a quad-A (AAAA) record is not available, the DNS64 servers attempts to locate an IPv4 A record instead. If an A record is located, DNS64 converts the IPv4 A record into an IPv6 AAAA record, using the NAT64 prefix. This gives the impression to the IPv6-only host that it can communicate with a server using IPv6. NAT64 does IP header translation between the two protocols.

Although dual-stack, tunneling, and translation methods are not the ideal environment for networks to transition to IPv6, they are tools that can make this transition easier. IETF provides these and other methods to help make the transition to IPv6 a reality. A global Internet encompassing a vast variety of networks in every scenario imaginable makes it almost impossible for any one tool to work for every situation.

This chapter discusses tunnels and overlay tunnels, which also enable IPv4 and IPv6 devices to coexist. Tunneling allows IPv6 devices to communicate over an IPv4-only network by encapsulating the IPv6 packets inside an IPv4 packet.

Wherever and whenever possible, native IPv6 is the way to go, but realistically, this cannot be the case in every environment. Now is the ideal time for network administrators to become familiar with IPv6. Configuring devices (hosts, routers, and so on) as dual-stacked is a perfect way to begin familiarizing yourself with IPv6 while safely maintaining the existing IPv4 infrastructure.

Review Questions

1. What is the recommended prefix for most traditional LAN deployments in order to support SLAAC?
   a. /32
   b. /48
   c. /64
   d. /127
   e. There is no recommended prefix.

2. How many hexadecimal digits should be used to represent the maximum number of VLANs in the Subnet ID with the VLAN ID represented in hexadecimal?
   a. One digit
   b. Two digits
   c. Three digits
   d. Four digits
3. What command is required to send Router Advertisement messages out a VLAN interface?
   a. **ipv6 unicast-routing**
   b. **no switchport**
   c. **switchport**
   d. **interface vlan**
4. Which FHRP uses multiple default gateways simultaneously and performs load balancing without any additional configuration?
   a. HSRP
   b. VRRP
   c. GLBP
   d. ICMPv6
5. Which FHRP is an industry standard protocol that assigns one master router and one or more backup routers?
   a. HSRP
   b. VRRP
   c. GLBP
   d. ICMPv6
6. Which protocol provides failover for IPv6 only?
   a. HSRP
   b. VRRP
   c. GLBP
   d. ICMPv6
7. What resource record maps a domain name to an IPv6 address?
   a. A record
   b. AA record
   c. AAA record
   d. AAAA record
8. What algorithm makes a more aggressive connection attempt over both IPv6 and IPv4 simultaneously, giving IPv6 a head start?
   a. DUAL
   b. Happy Eyeballs
   c. DNS64
   d. FastDNS

9. What two IPv6 ACL statements are implicitly at the end of an ACL to allow IPv6 Neighbor Discovery packets to be sent and received on an interface?
   a. **permit icmp any any nd-na** and **permit icmp any any nd-ns**
   b. **deny icmp any any nd-na** and **deny icmp any any nd-ns**
   c. **permit icmp any any nd-ra and permit icmp any any nd-rs**
   d. **permit icmp any any nd and permit icmp any any nd**
10. Which transition technology transparently provides access between IPv6-only and IPv4-only networks utilizing the services of DNS64?
    a. NAT-PT
    b. NAT64
    c. Overlay tunnel
11. Which transition technology encapsulates an IPv6 packet inside an IPv4 packet?
    a. NAT-PT
    b. NAT64
    c. Overlay tunnel
12. Which transition technology has been deprecated by IETF?
    a. NAT-PT
    b. NAT64
    c. Overlay tunnel

References

RFCs

RFC 2766, *Network Address Translation - Protocol Translation (NAT-PT)*, G. Tsirtsis, Campio Communications, www.ietf.org/rfc/rfc2766, February 2000.

RFC 2784, *Generic Routing Encapsulation (GRE)*, D. Farinacci, Procet Networks, www.ietf.org/rfc/rfc2784, March 2000.

RFC 3056, *Connection of IPv6 Domains via IPv4 Clouds (6to4)*, B. Carpenter, www.ietf.org/rfc/rfc3056, February 2001.

RFC 3142, *An IPv6-to-IPv4 Transport Relay Translator*, J. Hagino, IIJ Research Laboratory, www.ietf.org/rfc/rfc3142, June 2001.

RFC 3493, *Basic Socket Interface Extensions for IPv6*, R. Gilligan, Intransa, Inc., www.ietf.org/rfc/rfc3493, February 2003.

RFC 3596, *DNS Extensions to Support IP Version 6*, S. Thomson, Cisco, tools.ietf.org/html/rfc3596, October 2003.

RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, T. Berners-Lee, W3C/MIT, www.ietf.org/rfc/rfc3986, January 2005.

RFC 4380, *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)*, C. Huitema, Microsoft, www.ietf.org/rfc/rfc4380, February 2006.

RFC 4472, *Operational Considerations and Issues with IPv6 DNS*, A. Duran, Comcast, www.ietf.org/rfc/rfc4472, April 2006.

RFC 4966, *Reasons to Move the Network Address Translator–Protocol Translator (NAT-PT) to Historic Status*, C. Aoun, Energize Urnet, www.ietf.org/rfc/rfc4966, July 2007.

RFC 5214, *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)*, F. Templin, Boeing Phantom Works, www.ietf.org/rfc/rfc5214.txt, March 2008.

RFC 5342, *IANA & IETF Use of IEEE 802 Parameters*, D. Eastlake, Eastlake Enterprises, www.ietf.org/rfc/rfc5342.txt, September 2009.

RFC 5569, *IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)*, R. Despres, RD-IPtech, tools.ietf.org/html/rfc5569, January 2010.

RFC 6145, *IP/ICMP Translation Algorithm*, X. Li, CERNET Center/Tsinghua, tools.ietf.org/html/rfc6145, April 2011.

RFC 6146, *Stateful NAT64: Network Address and Protocol Translation*, M. Bagnulo, UC3M, tools.ietf.org/html/rfc6146, April 2011.

RFC 7381, *Enterprise IPv6 Deployment Guidelines*, K. Chittimaneni, Dropbox, Inc., tools.ietf.org/html/rfc7381, October 2014.

RFC 7597, *Mapping of Address and Port with Encapsulation (MAP-E)*, O. Troan, Cisco, tools.ietf.org/html/rfc7597, July 2015.

RFC 7599, *Mapping of Address and Port with Translation (MAP-T)*, X. Li, Tsinghhua University, tools.ietf.org/html/rfc7599, July 2015.

Websites

IPv6 Design and Deployment LiveLessons, by Tim Martin, Cisco Press, www.ciscopress.com/store/ipv6-design-and-deployment-livelessons-9780134655512.

NAT64 Technology: Connecting IPv6 and IPv4 Networks, www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/enterprise-ipv6-solution/white_paper_c11-676278.html.

NAT64—Stateless Versus Stateful, www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/enterprise-ipv6-solution/white_paper_c11-676277.html.

Technical Guide to Mapping of Address and Port (MAP), www.cisco.com/c/en/us/solutions/collateral/ios-nx-os-software/enterprise-ipv6-solution/whitepaper_C11-729800.html.

Implementing Tunneling for IPv6, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-tunnel.html.

Implementing Traffic Filters and Firewalls for IPv6 Security, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-sec-trfltr-fw.html.

Cisco IPv6 Command Reference, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book.html.

Cisco FHRP Configuration Guide: HSRP for IPv6, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipapp_fhrp/configuration/15-mt/fhp-15-mt-book/ip6-fhrp-hsrp.html.

Cisco FHRP Configuration Guide, VRRPv3 Protocol Support, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipapp_fhrp/configuration/15-mt/fhp-15-mt-book/fhrp-vrrpv3.html.

Cisco FHRP Configuration Guide, FHRP—GLBP Support for IPv6, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipapp_fhrp/configuration/15-sy/fhp-15-sy-book/GLBP-IPv6.html.

Appendix A

Configuring NAT64 and IPv6 Tunnels

Configuring NAT64

This section describes the basic commands to configure NAT64, which is discussed in Chapter 17, Deploying IPv6 in the Network. There are various options and scenarios that can be used in the configuration of NAT64 such as:

- An IPv6 network to the IPv4 Internet
- The IPv4 Internet to an IPv6 network
- The IPv6 Internet to an IPv4 network
- An IPv4 network to the IPv6 Internet
- An IPv6 network to an IPv4 network
- An IPv4 network to an IPv6 network
- The IPv6 Internet to the IPv4 Internet
- The IPv4 Internet to the IPv6 Internet

Table A-1 shows the basic commands used to configure the NAT64 router.

Table A-1 *NAT64 Configuration Commands*

Command	Description
Router(config)# **interface** *type number*	Specifies an interface type and number, and places the router in interface configuration mode. This is the interface that faces the IPv6-only network and will be configured with an IPv6 address.
Router(config-if)# **ipv6 address** *ipv6-address/prefix-length*	Specifies the IPv6 address and prefix length to be assigned to the interface.

Command	Description
Router(config-if)# **nat64 enable**	Enables NAT64 translation on the interface.
Router(config)# **interface** *type number*	Specifies an interface type and number, and places the router in interface configuration mode. This is the interface that faces the IPv4-only network and will be configured with an IPv4 address.
Router(config-if)# **ip address** *ipv4-address subnet-mask*	Specifies the IPv4 address and subnet mask to be assigned to the interface.
Router(config-if)# **nat64 enable**	Enables NAT64 translation on the interface.
Router(config)# **nat64 prefix stateful** *ipv6-prefix/prefix-length*	Defines the prefix and a prefix length for stateful NAT64: ■ *ipv6-prefix:* IPv6 network number to include in router advertisements. This argument must be in the form documented in RFC 2373, where the address is specified in hexadecimal using 16-bit values between colons. ■ */prefix-length:* Length of the IPv6 prefix. A decimal value that indicates how many of the high-order contiguous bits of the address comprise the prefix (the network portion of the address). A slash mark must precede the decimal value.
Router(config)# **nat64 v4 pool** *pool-name start-address-range end-address-range*	Enables NAT64 IPv4 configuration: ■ **pool:** Configures an IPv4 address pool. ■ *pool-name:* Name of the IPv4 address pool. ■ *start-address-range:* Starting address of the address pool range. ■ *end-address-range:* Ending address of the address pool range.
Router(config)# **nat64 v6v4 list** *list-name* **pool** *pool-name* [**overload**]	Translates an IPv6 source address to an IPv4 source address and an IPv4 destination address to an IPv6 destination address for NAT64: ■ **list:** Associates an IPv4 pool with the filtering mechanism that decides when to apply an IPv6 address mapping. ■ *list-name:* Name of the IPv6 access list.

Command	Description
	■ **pool:** Specifies the NAT64 pool for dynamic mapping of addresses. ■ *pool-name:* Name of the NAT64 pool. ■ **overload:** (Optional) Enables NAT64 overload address translation.
Router(config)# **ipv6 access-list** *access-list-name*	Defines an IPv6 ACL, and enters IPv6 access list configuration mode. ■ *access-list-name:* This argument specifies the name of the IPv6 ACL.
Router(config-ipv6-acl)# **ipv6 permit** *ipv6-address/ipv6-prefix-length*	Specifies the IPv6 address and prefix length to be translated.

Example A-1 shows the configuration for the NAT64 router connecting an IPv6 network to the IPv4 Internet shown in Figure A-1. (The steps in Figure A-1 are described in Chapter 17, "Deploying IPv6 Networks.") The list that follows goes through the configuration commands. Routing and any redistribution command options are not shown.

Example A-1 *Sample NAT64 Configuration*

```
NAT64-Router(config)# interface GigabitEthernet 0/0/1
NAT64-Router(config-if)# description Connected to IPv6 Network
NAT64-Router(config-if)# ipv6 address 2001:db8:cafe:1::1/64
NAT64-Router(config-if)# nat64 enable
NAT64-Router(config-if)# exit
NAT64-Router(config)# interface GigabitEthernet 0/0/2
NAT64-Router(config-if)# description Connected to IPv4 Network
NAT64-Router(config-if)# ip address 192.168.99.1 255.255.255.0
NAT64-Router(config-if)# nat64 enable
NAT64-Router(config-if)# exit

NAT64-Router(config)# nat64 prefix stateful 2001:db8:cafe:aaaa::/96
NAT64-Router(config)# nat64 v4 pool pool1 192.168.99.9 192.168.99.10
NAT64-Router(config)# nat64 v6v4 list mylist pool1 overload
NAT64-Router(config)# ipv6 access-list mylist

NAT64-Router(config-ipv6-acl)# permit ipv6 2001:db8:cafe::/48 any
```

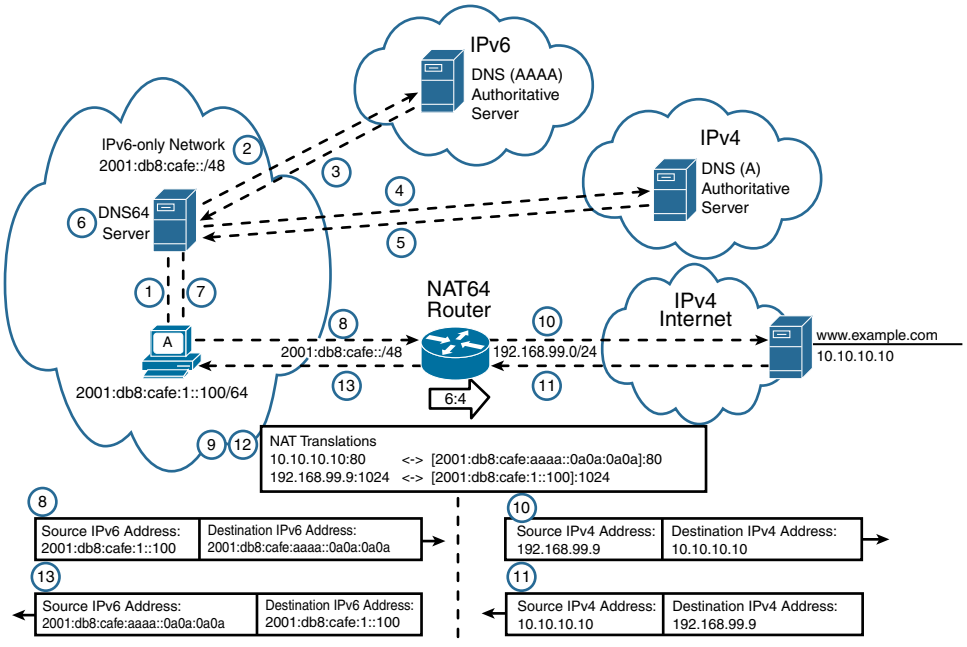


Figure A-1 *NAT64 Router Translating IPv6 Traffic to IPv4 and Returning Traffic*

```
NAT64-Router(config)# interface GigabitEthernet 0/0/1
```

This is the interface connected to the IPv6-only network.

```
NAT64-Router(config-if)# ipv6 address 2001:db8:cafe:1::1/64
```

This command configures the IPv6 global unicast routing address of the interface.

```
NAT64-Router(config-if)# nat64 enable
```

This command enables stateless NAT64 translation on an IPv6 interface.

```
NAT64-Router(config)# interface GigabitEthernet 0/0/2
```

This is the interface connected to the IPv4-only network.

```
NAT64-Router(config-if)# ip address 192.168.99.1 255.255.255.0
```

This command configures the IPv4 address of the interface.

```
NAT64-Router(config-if)# nat64 enable
```

This command enables stateless NAT64 translation on an IPv6 interface.

```
NAT64-Router(config)# nat64 prefix stateful 2001:db8:cafe:aaaa::/96
```

This command enables NAT64 IPv6-to-IPv4 address mapping. The NAT64 prefix 2001:db8:cafe:aaaa::/96 will be used with an IPv4 address appended.

```
NAT64-Router(config)# nat64 v4 pool pool1 192.168.99.9 192.168.99.10
```

This command defines the stateful NAT64 IPv4 address pool. These are the IPv4 addresses that can be used for NAT64 translation.

```
NAT64-Router(config)# nat64 v6v4 list mylist pool1 overload
```

This command dynamically translates an IPv4 source address to an IPv6 source address and an IPv6 destination address to an IPv4 destination address for NAT64. The **overload** statement enables NAT64 overload address translation similar to NAT overloading for IPv4.

```
NAT64-Router(config)# ipv6 access-list mylist
```

This command defines an IPv6 access list and enters IPv6 access list configuration mode.

```
NAT64-Router(config-ipv6-acl)# permit ipv6 2001:db8:cafe::/48 any
```

This command specifies the IPv6 address and prefix length to be translated.

For more information, read Stateful Network Address Translation 64, www.cisco.com/en/US/docs/ios-xml/ios/ipaddr_nat/configuration/xe-3s/iadnat-stateful-nat64.pdf. For stateless NAT64 configuration information, read Stateless Network Address Translation 64, www.cisco.com/en/US/docs/ios-xml/ios/ipaddr_nat/configuration/xe-3s/iadnat-stateless-nat64.pdf.

Configuring IPv6 Tunnels

Manual Tunnels

A manual tunnel is a bidirectional, point-to-point tunnel where the IPv4 endpoint addresses are determined in the configuration. In many ways, this is the easiest tunnel to implement. The advantage to manual tunnels is that configuration is very straightforward. The disadvantage is that manual tunnels don't scale well. Manual tunnels are point-to-point tunnels. If the environment requires multiple tunnels between several routers, a separate tunnel is required between every two endpoints or routers. This results in a full mesh of endpoints. The number of tunnels required would be as follows (n = number of endpoints):

$$\text{Number of tunnels} = n(n-1) / 2$$

Although manual tunnels might work fine for a limited number of routers and interconnected tunnels, after only a few connections, the number of tunnels could become unwieldy. Figure A-2 shows the topology for the manual tunnel to be created between two IPv6 islands, Rick's Café network of 2001:db8:cafe::/48 and Ace's Surfboards' 2001:db8:ace::/48 network. Between these two isolated IPv6 networks is an IPv4-only network. Routers R1 and R2 are dual-stack routers acting as the edge or border

routers to tunnel the IPv6 packets over IPv4. Again, Protocol 41 in the IPv4 header indicates that the encapsulated data portion is an IPv6 packet.

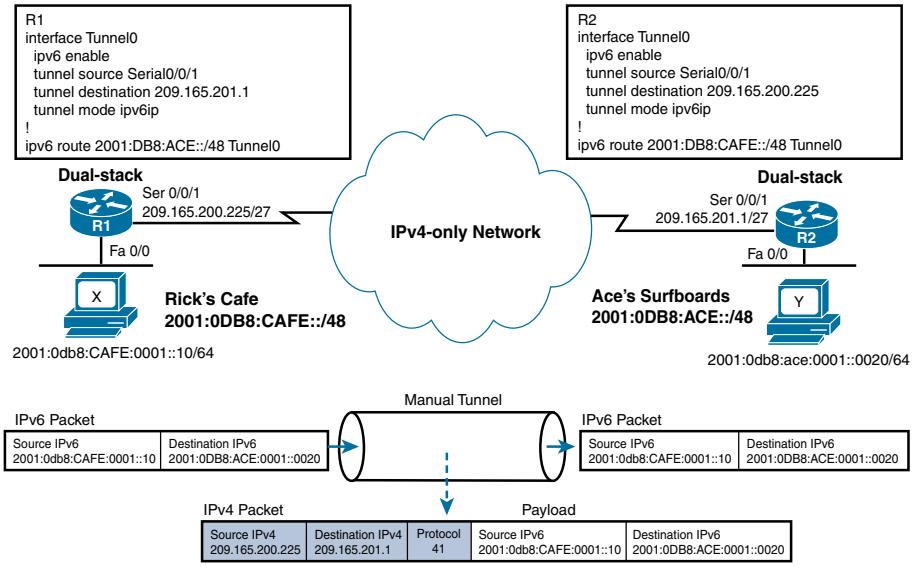


Figure A-2 *Manual Tunnel*

Although the tunnel will be a point-to-point virtual link, the IPv4-only network between the two border routers does not have to be a point-to-point network. In other words, the IPv4 interfaces on Routers R1 and R2 do not have to be on the same subnet. Just as in any IP network, reachability between both routers is required. R1 must be able to reach the 209.165.201.0/27 network of R2's serial interface, and R2 must be able to reach the 209.165.200.224/24 network of R1's serial interface.

The commands used to configure a manual tunnel are shown in Table A-2.

Table A-2 *Configuration Commands for a Manual Tunnel*

Command	Description
Router(config)# **interface tunnel** *tunnel-number*	Specifies a tunnel interface and number, and enters interface configuration mode.
Router(config-if)# **ipv6-address** *ipv6-prefix/prefix-length* [**eui-64**]	Specifies the IPv6 network assigned to the interface and enables IPv6 processing on the interface. Instead of specifying an IPv6 address, the **ipv6 enable** command can be used to create a link-local address and enable IPv6 on the interface.
Router(config-if)# **tunnel source** {*ip-address* | *interface-type interface number*}	Specifies the source IPv4 address or the source interface type and number for the tunnel interface. The source IPv4 address must be reachable from the other side of the tunnel.

Command	Description
	If an interface is specified, the interface must be configured with an IPv4 address. The address can be a physical or loopback address, but must be reachable from the other end of the tunnel.
Router(config-if)# **tunnel destination** *ip-address*	Specifies the destination IPv4 address or host name for the tunnel interface.
Router(config-if)# **tunnel mode ipv6ip**	Specifies a manual IPv6 tunnel. **Note:** The **tunnel mode ipv6ip** command specifies IPv6 as the passenger protocol and IPv4 as both the encapsulation and transport protocol for the manual IPv6 tunnel.
Router(config)# **ipv6 route** *ipv6-prefix/prefix-length* **tunnel** *tunnel-number*	Configures a static route for the IPv6 prefix using the tunnel number specified in the **interface tunnel** command.

In Example A-2, notice how these configuration commands are used on R1 and R2 to create their manual tunnel.

Example A-2 *Configuring a Manual Tunnel on R1 and R2*

```
R1(config)# interface Serial0/0/1
! The next command establishes the tunnel source for the tunnel from R1 to R2
R1(config-if)# ip address 209.165.200.225 255.255.255.224
R1(config-if)# exit

R1(config)# interface Tunnel 0
R1(config-if)# ipv6 enable
R1(config-if)# tunnel source Serial0/0/1
! The next command specifies the tunnel destination
R1(config-if)# tunnel destination 209.165.201.1
R1(config-if)# tunnel mode ipv6ip
R1(config-if)# exit

R1(config)# ipv6 route 2001:db8:ace::/48 Tunnel 0
-------------------------------------------------------
R2(config)# interface Serial0/0/1
! The next command is the tunnel destination
R2(config-if)# ip address 209.165.201.1 255.255.255.224
R2(config-if)# exit

R2(config)# interface Tunnel 0
R2(config-if)# ipv6 enable
R2(config-if)# tunnel source Serial0/0/1
```

```
R2(config-if)# tunnel destination 209.165.200.225
R2(config-if)# tunnel mode ipv6ip
R2(config-if)# exit

R1(config)# ipv6 route 2001:db8:cafe::/48 Tunnel 0
```

Look at the configuration of R1:

```
R1(config)# interface Serial0/0/1
R1(config-if)# ip address 209.165.200.225 255.255.255.224
```

This command configures the IPv4 address for R1's serial interface.

```
R1(config)# interface Tunnel 0
```

This command specifies the tunnel interface/number and puts it into interface configuration mode.

```
R1(config-if)# ipv6 enable
```

Configuring an IPv6 address on the tunnel interface enables the interface for IPv6 processing. It is not meant for reachability. So, there are several options that can be used. Configure an address using the **ipv6 address** command, specify a loopback interface on the router that has an IPv6 address, or use the **ipv6 unnumbered** command using the IPv6 address of one of the router's physical interfaces. Again, the IPv6 address itself is not what is important. It is necessary to enable the tunnel interface for IPv6 processing.

```
R1(config-if)# tunnel source Serial0/0/1
```

The **tunnel source** command specifies the IPv4 address of the tunnel. This address must be reachable from the router at the other end of the tunnel. A specific IPv4 address can be used or an IPv4 address of a physical interface can be used, as was done by using serial 0/0/1. Regardless of how the tunnel's IPv4 address is determined, it must be reachable from the other endpoint of the tunnel. This is the IPv4 source address of the transport protocol.

```
R1(config-if)# tunnel destination 209.165.201.1
```

The **tunnel destination** command specifies the IPv4 address of the other end of the tunnel. In this case, the address must be reachable by R1. This is the IPv4 destination address of the transport protocol.

```
R1(config-if)# tunnel mode ipv6ip
```

This command specifies IPv6 as the passenger protocol (the "ipv6" part of "ipv6ip") and IPv4 ("ip" at the end of "ipv6ip") as the transport protocol for the manual IPv6 tunnel.

```
R1(config)# ipv6 route 2001:db8:ace::/48 Tunnel 0
```

R1's IPv6 routing process must be told how to reach the 2001:db8:ace::/48 network on R2. This is done with an IPv6 static route using the tunnel as the exit interface.

As shown in Example A-2, the configuration of R2 is similar to that of R1. Example A-3 verifies the configuration of R1's tunnel, including the tunnel source, tunnel destination, and type of tunnel.

Example A-3 *Verifying Tunnel Configuration*

```
R1# show interface tunnel 0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  MTU 1514 bytes, BW 9 Kbit, DLY 500000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 209.165.200.225 (Serial0/0/1), destination 209.165.201.1
  Tunnel protocol/transport IPv6/IP
<Rest of output omitted for brevity>
```

In Example A-4, verify the status of Tunnel 0 using the commands **show ip interface brief** for IPv4 and **show ipv6 interface brief** for IPv6. Notice that the status and protocol are "up" for both protocols, but the Tunnel 0 interface only has an IPv6 address. The IPv6 address of Tunnel 0 is the link-local address fe80::d1a5:c8e1, which was automatically created when the **ipv6 enable** command was used on the tunnel interface. The low-order 32 bits of the IPv6 link-local address are the hexadecimal representation of the IPv4 address 209.165.201.225 (d1 = 209, a5 = 165, c8 = 201, and e1 = 225). The actual source IPv4 address of the tunnel is the serial 0/0/1 interface of R1.

Example A-4 *Verifying Tunnel 0 on R1*

```
R1# show ipv6 interface brief
FastEthernet0/0            [up/up]
    FE80::1
    2001:DB8:CAFE:1::1
Serial0/0/1                [up/up]
Tunnel0                    [up/up]
    FE80::D1A5:C8E1
R1#

R1# show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            unassigned      YES manual up                    up
Serial0/0/1                209.165.200.225 YES manual up                    up
Tunnel0                    unassigned      YES unset  up                    up
R1#
```

Example A-5 verifies R1's reachability to Ace's Surfboards' 2001:db8:ace::/48 network by pinging the IPv6 address on R2. Use the **debug ip packet detail** command to view the IPv4 source and destination addresses of the transport protocol. Next, issue the **ping 2001:db8:ace:1::1 source fastethernet 0/0** command using R1's FastEthernet 0/0 interface as the source IPv6 address of the ping. The debug output confirms that the IPv4 source address is 209.165.200.225, the source address of the tunnel, and the IPv4 destination address is 209.165.201.1, the destination address of the tunnel.

Example A-5 *Examining the Transport Protocol Header*

```
R1# debug ip packet detail
IP packet debugging is on (detailed)
R1# ping 2001:db8:ace:1::1 source fastethernet 0/0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:ACE:1::1, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:CAFE:1::1
!!!!!

02:08:40: IP: s=209.165.200.225 (Tunnel0), d=209.165.201.1
(Serial0/0/1), len 120, sending, proto=41
02:08:40: IP: s=209.165.201.1 (Serial0/0/1), d=209.165.200.225
(Serial0/0/1), len 120, rcvd 3, proto=41
```

By applying the same **ping** command, you can also verify the IPv6 source and destination addresses of the passenger protocol using the **debug ipv6 packet** command prior to issuing the ping, as shown in Example A-6. The IPv6 source address is 2001:db8:cafe::1, the IPv6 address of R1's FastEthernet 0/0 interface specified in the **ping** command. The IPv6 destination address in the debug output matches that issued in the **ping** command, 2001:db8:ace:1::1.

Example A-6 *Examining the Passenger Protocol Header*

```
R1# debug ipv6 packet
IPv6 unicast packet debugging is on

R1# ping 2001:db8:ace:1::1 source fastethernet 0/0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:ACE:1::1, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:CAFE:1::1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/67/68 ms
R1#
```

```
02:24:13: IPv6: SAS picked source 2001:DB8:CAFE:1::1 for 2001:DB8:ACE:1::1
  (FastEthernet0/0)
02:24:13: IPV6: source 2001:DB8:CAFE:1::1 (local)
02:24:13:       dest 2001:DB8:ACE:1::1 (Tunnel0)
02:24:13:       traffic class 0, flow 0x0, len 100+0, prot 58, hops
64, originating
```

The process of configuring a tunnel and verifying its operation has been addressed. You might be wondering how the actual routing occurs. After all, you are routing an IPv6 packet over an IPv4 network. When the **ping** command was issued, as shown in Example A-6, an IPv6 packet was created with the

- IPv6 source address: 2001:db8:cafe::1
- IPv6 destination address: 2001:db8:ace:1::1

Example A-7 illustrates R1's routing process to forward the IPv6 packet over the tunnel, and is further described in the list that follows. The commented numbers to the right coincide with the descriptions that follow.

Example A-7 *Routing Process*

```
R1# show ipv6 route
<Rest of output omitted for brevity>

S   2001:DB8:ACE::/48 [1/0]    ! (1)
     via ::, Tunnel0

R1# show ip route
<Rest of output omitted for brevity>

     209.165.200.0/27 is subnetted, 1 subnets
C       209.165.200.224 is directly connected, Serial0/0/1   ! (3)
     209.165.201.0/27 is subnetted, 1 subnets
S       209.165.201.0 is directly connected, Serial0/0/1  ! (4)
R1#

R1(config)# interface Tunnel 0
R1(config-if)# ipv6 enable
R1(config-if)# tunnel source Serial0/0/1  ! (3)
R1(config-if)# tunnel destination 209.165.201.1  ! (4)
R1(config-if)# tunnel mode ipv6ip         ! (2)
```

Passenger protocol:

1. R1's IPv6 routing table is searched to find the best match for the packet's IPv6 - destination address of 2001:db8:ace:1::1. Using the entry created by the static route, the exit interface is determined to be Tunnel 0.
2. The tunneling mode of Tunnel 0 indicates that this IPv6 packet will be encapsulated in an IPv4 packet. The manual tunnel mode means that the IPv4 source and destination addresses (transport protocol) are specified within the tunnel interface commands.

Transport protocol:

3. The **tunnel source** command points to the IPv4 address of the Serial 0/0/1 interface of 209.165.200.225. This will be the IPv4 source address used by the IPv4 transport protocol. The source address must be in the "up" state and therefore must be in R1's routing table.
4. The **tunnel destination** command gives the IPv4 address of the other end of the tunnel, 209.165.201.1, the IPv4 destination address of the transport protocol. This address must be reachable by R1; in other words, it can be located in its routing table. The tunnel destination address of 209.165.201.1 matches a static route in the IPv4 routing table with serial 0/0/1 as the exit interface.

The IPv4 packet with its IPv6 payload is now forwarded out the serial 0/0/1 interface. When R2 receives the packet, it decapsulates the IPv6 packet and uses its IPv6 routing table to forward the packet to its destination.

6to4 Tunnels

Manual tunnels are easy to configure, but as mentioned, they do not scale well when a large number of tunnels is necessary. A separate manual tunnel must be configured for every pair of routers. Any time another router is added to the mix, a new tunnel must be configured on all the existing routers. This is similar to the disadvantage of using only static routes for network reachability. For some organizations, managing a few statically configured manual tunnels is acceptable, while others might want a more scalable solution.

IETF has defined a mechanism called 6to4 to automatically connect to multiple IPv6 networks over one configured tunnel. 6to4 tunnels are defined in RFC 3056, *Connection of IPv6 Domains via IPv4 Clouds*. A 6to4 tunnel is a point-to-multipoint connection. A single 6to4 tunnel can be used to connect to any number of IPv6 networks—in other words, having any number of tunnel destinations, as shown in Figure A-3. Router R1 can be configured with a single 6to4 tunnel capable of reaching the IPv6 networks through Routers R2 through R6.

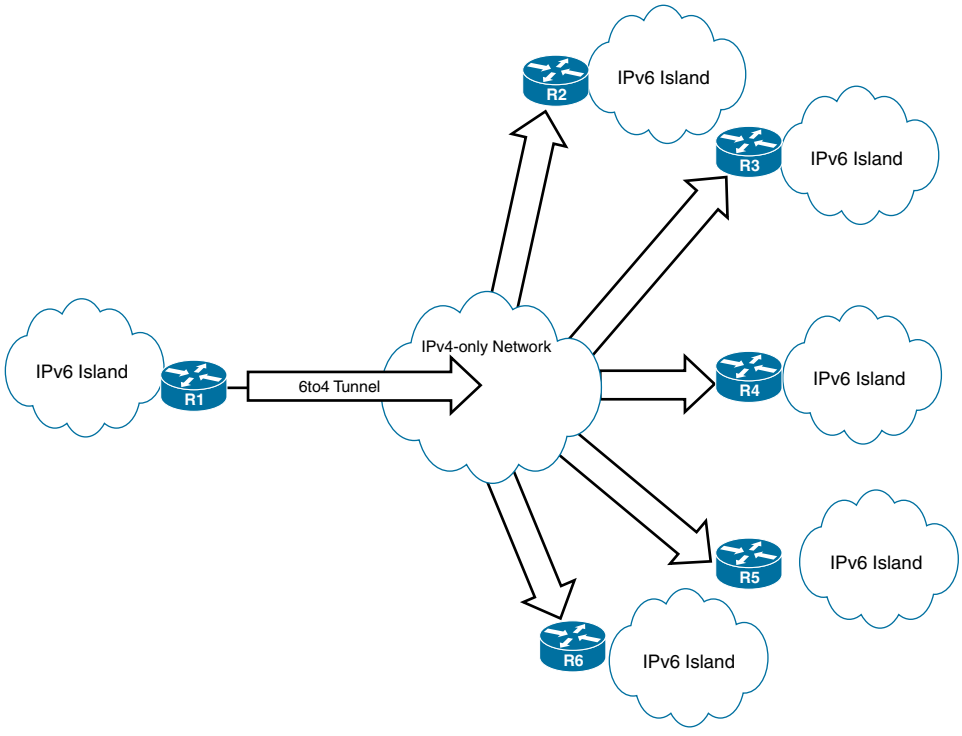


Figure A-3 *6to4 Tunnels*

The difference between 6to4 tunnels (also known as automatic 6to4 tunnels) and manual tunnels is that manual tunnels must statically configure the other end of the tunnel, the tunnel IPv4 destination address. With a 6to4 tunnel, the tunnel IPv4 destination address is automatically derived from the IPv6 destination of the packet. This means that there must be some sort of correlation between the two addresses. The IPv6 network address is based on a reachable IPv4 address plus a special prefix reserved for this purpose as defined in RFC 3056. The Internet Assigned Numbers Authority (IANA) has permanently assigned the 2002::/16 prefix to be used for IPv6 networks automatically accessible using a 6to4 tunnel.

Note A limitation of 6to4 tunneling is the inability to send traffic outside your domain. Both routers at the end of the tunnel must be configured as a 6to4 tunneling router; otherwise, your traffic is black holed. Also, 6to4 tunneling only supports static routes; there is no dynamic routing support.

Using the 32 bits of the IPv4 address results in an IPv6 address of 2002:*ipv4-address*::/48. Let's see how this is done.

Figure A-4 illustrates the process of reverse-engineering the IPv6 address from an IPv4 address illustrated in three simple steps that follow.

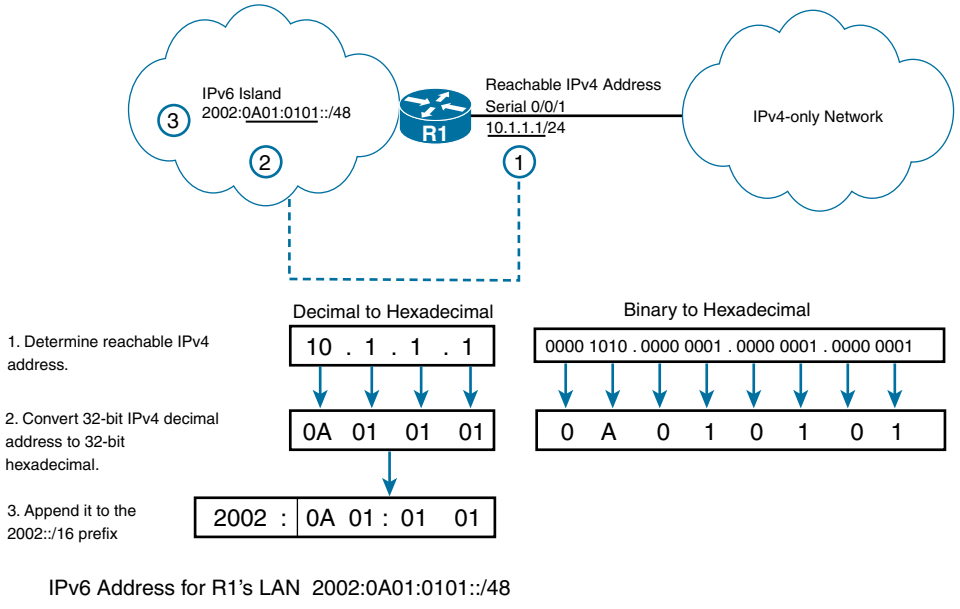


Figure A-4 *6to4 Reverse Engineering of IPv4 Address to Form IPv6 Address*

Step 1. Determine the IPv4-reachable address you want to use for the IPv6 address. This example uses the reachable 10.1.1.1/24 IPv4 address, R1's serial 0/0/1 interface. This is a routable address that can be reached by other routers at the other end of the tunnel.

Note This address was chosen to make the conversion easier to see, but if this were an Internet-facing router, the address would have to be a public IPv4 address instead of the RFC 1918 address used in this example.

Note The IPv4 address can be any routable address reachable by the other routers at the other end of the tunnel. Later in this chapter, an example of using a loopback address instead of the address of the physical interface is examined.

Step 2. Convert the 32-bit IPv4 address 10.1.1.1 expressed in decimal to hexadecimal. This results in the 32-bit hexadecimal value 0a01:0101.

Step 3. Append the 32-bit hexadecimal value 0a01:0101 to the IANA-reserved 6to4 prefix 2002::/16, making the IPv6 prefix or network address 2002:0a01:0101::/48. The hosts on R1's IPv6 network will have an IPv6 address using this prefix. The prefix can be subnetted and routed within R1's IPv6 domain.

Extend this method to other routers in the previous diagram. Figure A-5 shows the IPv6 addresses for each of the networks determined (reverse engineered) using the routable IPv4 address of their provider-facing interface. Although 10.0.0.0/8 addresses are not public addresses, they are used in this example to illustrate the conversion of an IPv4 address in decimal notation to the 32-bit hexadecimal notation used in the IPv6 address. Looking at Figure A-5, notice that all the IPv6 addresses have the 2002::/16 prefix followed by the converted 32-bit IPv4 address of their routable interface.

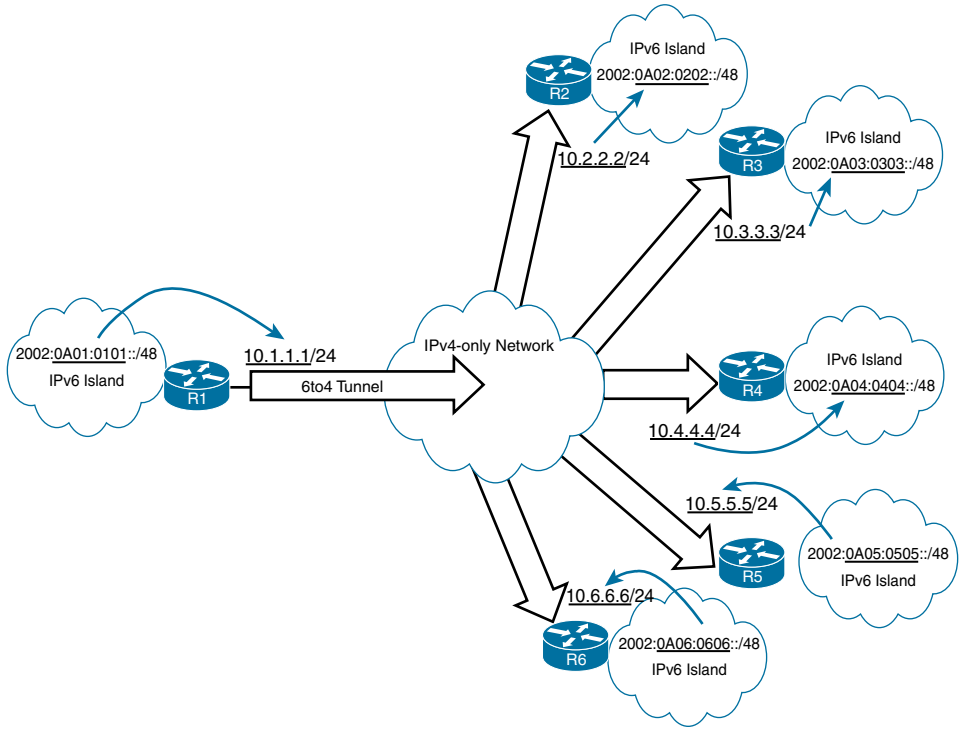


Figure A-5 *Examples of 6to4 Reverse Engineering of IPv4 Address to Form IPv6 Address*

The commands used to configure a 6to4 tunnel are shown in Table A-3. Note that these are very similar to the commands used to configure a manual tunnel. Example A-8 highlights the differences and the operation of the 6to4 tunnel.

Table A-3 *Configuration Commands for a 6to4 Tunnel*

Command	Description
Router(config)# **interface tunnel** *tunnel-number*	Specifies a tunnel interface and number, and enters interface configuration mode.
Router(config-if)# **ipv6-address** *ipv6-prefix/prefix-length* [**eui-64**]	Specifies the IPv6 network assigned to the interface and enables IPv6 processing on the interface. Instead of specifying an IPv6 address, the **ipv6 enable** command can be used to create a link-local address and enable IPv6 on the interface. If an IPv6 address is specified, the 32 bits following the initial 2002::/16 prefix must correspond to an IPv4 address assigned to the tunnel source.
Router(config-if)# **tunnel source** {*ip-address* | *interface-type interface-number*}	Specifies the source IPv4 address or the source interface type and number for the tunnel interface. The source IPv4 address must be reachable from the other side of the tunnel. If an interface is specified, the interface must be configured with an IPv4 address. The address can be a physical or loopback address, but must be reachable from the other end of the tunnel.
Router(config-if)# **tunnel mode ipv6ip 6to4**	Specifies an IPv6 tunnel using a 6to4 address. The IPv4 destination address will be determined using the 6to4 technique.
Router(config)# **ipv6 route** *ipv6-prefix/prefix-length* **tunnel** *tunnel-number*	Configures a static route for the IPv6 6to4 prefix 2002::/16 to the specified tunnel interface. The tunnel number specified in the **ipv6 route** command must be the same tunnel number specified in the **interface tunnel** command.

The best way to see how all of this comes together is to configure a 6to4 tunnel and see how it all works. Figure A-6 shows this scenario. Begin by configuring a 6to4 tunnel on Routers R1 and R2. All other routers will have similar configurations.

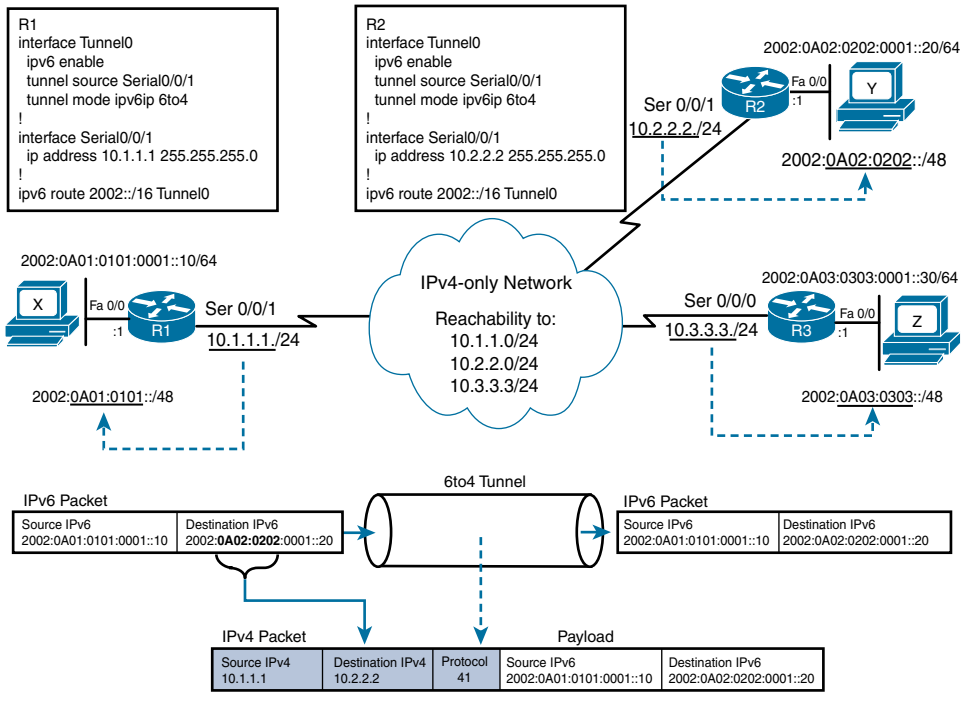


Figure A-6 *6to4 Tunnel Using Serial Interfaces*

Example A-8 shows the configurations for Routers R1 and R2. Most of this is similar to the configuration of the manual tunnel shown earlier. Notice that the IPv6 addresses on the FastEthernet 0/0 interfaces of R1 and R2 are their serial IPv4 addresses appended with the 2002::/16 prefix. Remember, the goal is the same as with a manual tunnel. You are encapsulating an IPv6 packet (passenger protocol) inside an IPv4 packet (transport protocol). This allows IPv6 packets to be sent between isolated IPv6 networks over an IPv4 network.

Example A-8 *Configuring a 6to4 Tunnel on R1 and R2*

```
R1(config)# interface Serial0/0/1
! IPv4 address matches 32 bits of FastEthernet0/0 IPv6 address
R1(config-if)# ip address 10.1.1.1 255.255.255.0
R1(config-if)# exit

R1(config)# interface FastEthernet0/0
! 32 bits of IPv6 address matches Serial0/0/1 IPv4 address
R1(config-if)# ipv6 address 2002:a01:101:1::1/64
R1(config-if)# exit
```

```
R1(config)# interface Tunnel 0
R1(config-if)# ipv6 enable
R1(config-if)# tunnel source Serial0/0/1
! Use 6to4 technique to determine tunnel destination IPv4 address
R1(config-if)# tunnel mode ipv6ip 6to4
R1(config-if)# exit

R1(config)# ipv6 route 2002::/16 Tunnel 0
```

```
R2(config)# interface Serial0/0/1
! IPv4 address matches 32 bits of FastEthernet0/0 IPv6 address
R2(config-if)# ip address 10.2.2.2 255.255.255.0
R2(config-if)# exit

R2(config)# interface FastEthernet0/0
! 32 bits of IPv6 address matches Serial0/0/1 IPv4 address
R2(config-if)# ipv6 address 2002:a02:202:1::1/64
R2(config-if)# exit

R2(config)# interface Tunnel 0
R2(config-if)# ipv6 enable
R2(config-if)# tunnel source Serial0/0/1
! Use 6to4 technique to determine tunnel destination IPv4 address
R2(config-if)# tunnel mode ipv6ip 6to4
R2(config-if)# exit

R2(config)# ipv6 route 2002::/16 Tunnel 0
```

Continuing with the configuration in Example A-8, notice that the tunnel interface commands are similar to what was used for the manual tunnel. However, there are two exceptions: the **tunnel mode ipv6ip 6to4** command and the lack of a **tunnel destination** command. The **tunnel mode ipv6ip 6to4** command tells the router to use the IPv6 destination address (passenger protocol) to determine the tunnel destination IPv4 address (transport protocol). Because the tunnel destination addresses are automatically ascertained from the IPv6 packet, there is no need for a **tunnel destination** command and a predetermined destination address. This is where the magic happens! Because the tunnel destination is automatically determined based on the destination of the IPv6 packet, any IPv6 network is now reachable as long as it adheres to the 2002:*ipv4-address*::/48 format.

The last part is the static route, **ipv6 route 2002::/16 Tunnel 0**. Any IPv6 packet with the 2002::/16 prefix is delivered to the Tunnel 0 interface for forwarding. As mentioned, the **tunnel mode ipv6ip 6to4** command determines the actual IPv4 destination address to be used when encapsulating the IPv6 packet into IPv4.

Example A-9 shows successful pings from R1 to the IPv6 networks of R2 and R3.

Example A-9 *Verifying Connectivity Using the* **ping** *Command*

```
R1# ping 2002:0a02:0202:1::1 source fastethernet 0/0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2002:A02:202:1::1, timeout is 2 seconds:
Packet sent with a source address of 2002:A01:101:1::1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/72/92 ms

R1# ping 2002:0a03:0303:1::1 source fastethernet 0/0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2002:A03:303:1::1, timeout is 2 seconds:
Packet sent with a source address of 2002:A01:101:1::1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/66/68 ms
R1#
```

Using the first **ping** command, notice how R1 determines the IPv4 destination address of the other end of the tunnel. The command issued on R1 is

```
R1# ping 2002:0a02:0202:1::1 source fastethernet 0/0
```

The best match in the IPv6 routing table for the destination address 2002:0a02:0202:1::1 is the entry from the configured static route:

```
R1(config)# ipv6 route 2002::/16 Tunnel 0
```

The resulting exit interface is Tunnel 0. Looking at R1's configuration in Example A-8, notice how this tunnel is used to forward the IPv6 packet. An IPv6 packet, the passenger protocol, is encapsulated inside an IPv4 packet, the transport protocol. Similar to a manual tunnel, the **tunnel source** is defined—in this case, the IPv4 address of serial 0/0/1, 10.1.1.1.

Unlike manual tunnels, the **tunnel destination** is not defined. The **tunnel mode ipv6ip 6to4** command directs the router to use the IPv6 packet's destination address to determine the IPv4 address of the other end of the tunnel. The 32 bits following the 2002::/16 prefix, 0a02:0202, are used as the destination IPv4 address for the transport protocol, 10.2.2.2. This is illustrated at the bottom of Figure A-6.

So, as long as the IPv6 addresses adhere to the 2002:*ipv4-address*::/48 format, any of our IPv6 islands are reachable using a single 6to4 tunnel.

6to4 Tunnels and Loopback Interfaces

An alternative to using the IPv4 address of a physical interface is to use a loopback address. A loopback address provides several advantages, including the fact that it never goes down as long as the router is still up. Just like the IPv4 address of a physical interface, the IPv4 address of a loopback interface must be reachable by the router at the other end of the tunnel. Figure A-7 shows the same topology used previously; however, loopback interfaces are used instead of the physical provider-facing interfaces to determine the IPv6 address of the network. The loopback address is used as the **tunnel source** IPv4 address. Example A-10 shows the configuration for R1 and R2 using loopback interfaces for the **tunnel source**.

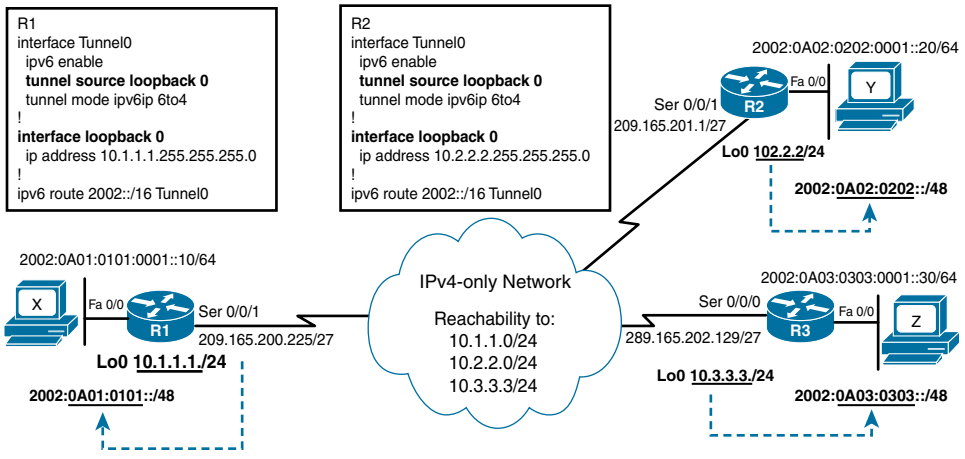


Figure A-7 *6to4 Tunnel Using Loopback Interfaces*

Example A-10 *Configuring a 6to4 Tunnel on R1 and R2 Using Loopback Interfaces*

```
R1(config)# interface Serial0/0/1
R1(config-if)# ip address 209.165.200.225 255.255.255.224
R1(config-if)# exit
R1(config)# interface loopback0
R1(config-if)# ip address 10.1.1.1 255.255.255.0
R1(config)# interface FastEthernet0/0
R1(config-if)# ipv6 address 2002:a01:101:1::1/64
R1(config-if)# exit

R1(config)# interface Tunnel 0
R1(config-if)# ipv6 enable
R1(config-if)# tunnel source loopback0
R1(config-if)# tunnel mode ipv6ip 6to4
R1(config-if)# exit
```

```
R1(config)# ipv6 route 2002::/16 Tunnel 0
```

```
R2(config)# interface Serial0/0/1
R2(config-if)# ip address 209.165.201.1 255.255.255.224
R2(config-if)# exit
R2(config)# interface loopback0
R2(config-if)# ip address 10.2.2.2 255.255.255.0
R2(config)# interface FastEthernet0/0
R2(config-if)# ipv6 address 2002:a02:202:1::1/64
R2(config-if)# exit

R2(config)# interface Tunnel 0
R2(config-if)# ipv6 enable
R2(config-if)# tunnel source loopback0
R2(config-if)# tunnel mode ipv6ip 6to4
R2(config-if)# exit

R2(config)# ipv6 route 2002::/16 Tunnel 0
```

ISATAP

ISATAP (Intra-Site Automatic Tunnel Addressing Protocol) tunnels are designed for transporting IPv6 packets within a site where a native IPv6 infrastructure is not yet available. Although the ISATAP tunneling mechanism is similar to other automatic tunneling mechanisms, such as IPv6 6to4, ISATAP is designed for transporting IPv6 packets within a site, but not between sites. ISATAP uses a well-defined IPv6 address format composed of any /64 unicast IPv6 prefix and a 64-bit Interface ID that includes the last 32 bits of the IPv6 address.

Like other tunneling methods, an ISATAP tunnel can exist between any two dual-stack devices: host-to-router, router-to-host, or host-to-host. However, the main feature of ISATAP tunnels is to provide dual-stack hosts with access to the IPv6 network over an IPv4-only network. ISATAP is an IPv6 transition technology that allows hosts on an IPv4 network to communicate with hosts on an IPv6-only network, and it will be the focus in this section.

Note ISATAP is considered a quick and temporary solution to give access to IPv6 resources. Imagine if there were ten developers in a building that needed access to IPv6 resources. Instead of enabling IPv6 everywhere, you could use ISATAP as a temporary solution to give IPv6 access only to those ten developers.

ISATAP is defined in RFC 5214, *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)*. The topology in Figure A-8 illustrates a dual-stack host connected to a dual-stack ISATAP router. In Figure A-8, a lot of information is provided to give you a view of ISATAP at a glance.

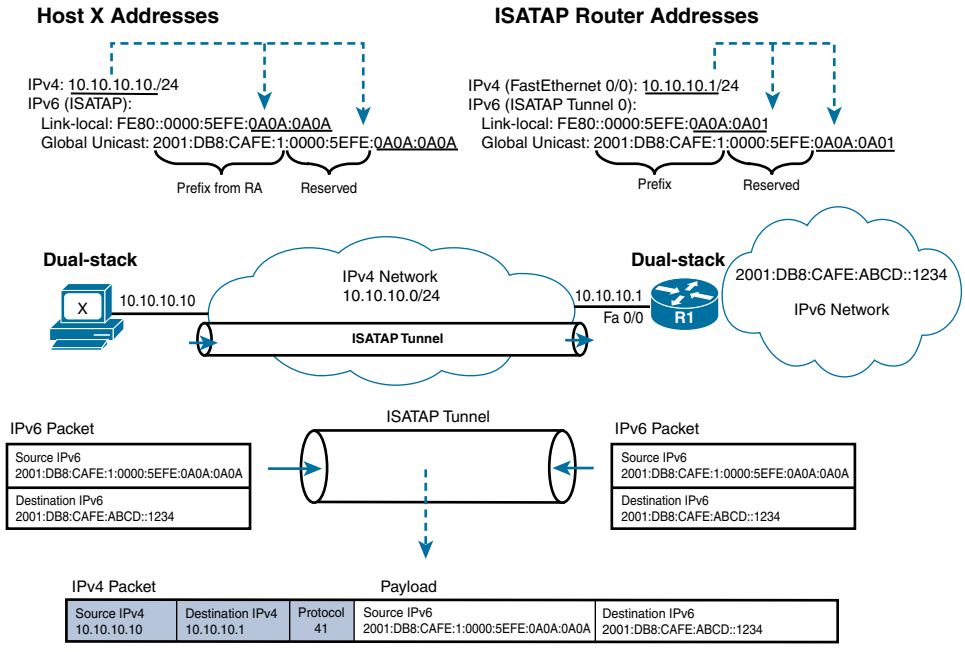


Figure A-8 *ISATAP Tunnel*

Router R1 is an ISATAP router, and at the other end of the tunnel is a dual-stack device, host X. The ISATAP router provides host X with IPv6 access over the IPv4 network using an ISATAP tunnel. R1, the ISATAP router, provides network configuration support to hosts over the ISATAP tunnel using Router Advertisements. This is the same Stateless Address Autoconfiguration process that takes place when a host and router are connected by a native IPv6 network. Let's look more closely at the ISATAP address format, which is shown in Figure A-9.

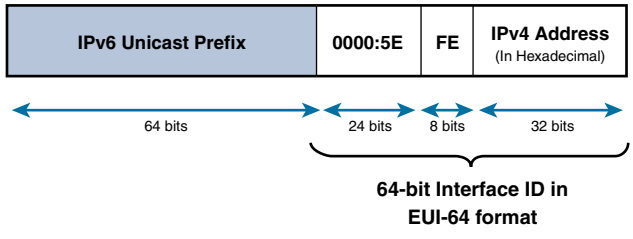


Figure A-9 *ISATAP Address Format*

The ISATAP address includes the following:

- **Prefix: IPv6 unicast prefix (64 bits):** This can be any valid unicast prefix, including a global routable prefix, a unique-local prefix, a link-local prefix, and even a 6to4 prefix. The prefix should be selected as part of the address plan for the network.

- **Interface ID (EUI-64 format):**
  - **00-00-5e (24 bits):** This is an Ethernet OUI (Organizationally Unique Identifier) reserved by IANA. This OUI is used by several protocols including ISATAP. This OUI is used for dynamic mapping between IP addresses and IEEE 802 MAC addresses.
  - **fe (8 bits):** This indicates that this address contains an embedded IPv4 address.
  - **IPv4 address (32 bits):** The last 4 bytes contain the IPv4 address in hexadecimal.

Note The EUI-64 format used by ISATAP is different than the one discussed in Chapter 6, "Link-Local Unicast Address," which embeds fffe in the middle of the Ethernet MAC address and flips the U/L bit to form the 64-bit Interface ID. In RFC 5342, *IANA & IETF Use of IEEE 802 Parameters*, IETF defines the EUI-64 format as any address constructed from an OUI owned by the IANA.

Table A-4 shows the commands for configuring an ISATAP tunnel interface on a router.

Table A-4 *Configuration Commands for an ISATAP Tunnel*

Command	Description
Router(config)# **interface tunnel** *tunnel-number*	Specifies a tunnel interface and number, and enters interface configuration mode.
Router(config-if)# **ipv6-address** *ipv6-prefix/prefix-length* [**eui-64**]	Specifies the IPv6 network assigned to the interface and enables IPv6 processing on the interface. Any IPv6 address will do. Using the **eui-64** option will create an IPv6 address using the ISATAP EUI-64 format.
Router(config-if)# **no ipv6 nd suppress-ra**	The sending of IPv6 Router Advertisements is disabled by default on tunnel interfaces. This command reenables the sending of IPv6 Router Advertisements to allow client autoconfiguration. This command can also be entered as **no ipv6 nd ra suppress**.
Router(config-if)# **tunnel source** {*ip-address* | *interface-type interface number*}	Specifies the source IPv4 address or the source interface type and number for the tunnel interface. The source IPv4 address must be reachable from the other side of the tunnel. If an interface is specified, the interface must be configured with an IPv4 address. The address can be a physical or loopback address, but must be reachable from the other end of the tunnel.
Router(config-if)# **tunnel mode ipv6ip isatap**	Specifies an IPv6 tunnel using an ISATAP address.

Using the topology shown in Figure A-8, Example A-11 shows the commands used to configure Router R1 as an ISATAP router.

Example A-11 *ISATAP Router Configuration for R1*

```
R1(config)# interface fastethernet 0/0
R1(config-if)# ip address 10.10.10.1 255.255.255.0
R1(config-if)# exit

R1(config)# interface tunnel 0
R1(config-if)# ipv6 address 2001:db8:cafe:1::/64 eui-64
R1(config-if)# no ipv6 nd suppress-ra
R1(config-if)# tunnel source fastethernet 0/0
R1(config-if)# tunnel mode ipv6ip isatap
R1(config-if)# exit

R1(config)# interface loopback 1
R1(config-if)# ipv6 address 2001:db8:cafe:abcd::1234/64
R1(config-if)#
```

Following the configuration in Example A-11:

```
R1(config)# interface fastethernet 0/0
R1(config-if)# ip address 10.10.10.1 255.255.255.0
```

This is the IPv4 address that the host X and other clients on the 10.10.10.0/24 LAN use as their default gateway address, usually from a DHCP server.

```
R1(config)# interface tunnel 0
```

This is the tunnel interface used to forward IPv6 packets over IPv4.

```
R1(config-if)# ipv6 address 2001:db8:cafe:1::/64 eui-64
```

This command enables IPv6 processing on the interface and also assigns an IPv6 address to the tunnel. Any IPv6 address will work. In this case, the **eui-64** option is used, which creates an IPv6 address using the ISATAP format.

```
R1(config-if)# no ipv6 nd suppress-ra
```

By default, Cisco IOS disables ICMPv6 Router Advertisement messages on tunnel interfaces. This command enables RA messages to be sent across the tunnel. These RA messages allow hosts to autoconfigure their IPv6 addresses and receive an IPv6 default gateway address.

```
R1(config-if)# tunnel source fastethernet 0/0
```

This command specifies FastEthernet 0/0 as the source interface for the tunnel. This interface must have an IPv4 address assigned. You previously configured R1's FastEthernet 0/0 interface as 10.10.10.1/24.

```
R1(config-if)# tunnel mode ipv6ip isatap
```

With this command, R1's Tunnel 0 is defined as an ISATAP tunnel. This allows transporting IPv6 packets over IPv4 and uses the ISATAP address format for the tunnel's IPv6 address.

```
R1(config)# interface loopback 1
R1(config-if)# ipv6 address 2001:db8:cafe:abcd::1234/64
```

This command creates a virtual address on R1 to demonstrate how the host reaches a remote IPv6 network. This command is not specific to the configuration of ISATAP.

Example A-12 shows several commands used to verify R1's address configuration. The **show ip interface brief** command displays the statically configured IPv4 address of FastEthernet 0/0 as 10.10.10.1.

Example A-12 *Verifying R1's Addresses*

```
R1# show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            10.10.10.1      YES manual up                    up
Tunnel0                    unassigned      YES unset  up                    up
R1# show ipv6 interface brief
FastEthernet0/0            [up/up]
Tunnel0                    [up/up]
    FE80::5EFE:A0A:A01
    2001:DB8:CAFE:1:0:5EFE:A0A:A01
R1#

R1# show ipv6 interface tunnel 0
Tunnel0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::5EFE:A0A:A01
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:CAFE:1:0:5EFE:A0A:A01, subnet is 2001:DB8:CAFE:1::/64 [EUI]
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF0A:A01
  MTU is 1480 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  ND DAD is not supported
```

```
  ND reachable time is 30000 milliseconds
  ND advertised reachable time is 0 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements live for 1800 seconds
  ND advertised default router preference is Medium
  Hosts use stateless autoconfig for addresses.
R1#
```

The **show ipv6 interface brief** command in Example A-12 shows the IPv6 addresses of R1's Tunnel 0 interface. The IPv6 address uses the ISATAP format. In Example A-11, you configured the global unicast address of Tunnel 0 using the **ipv6 address 2001:db8:cafe:1::/64 eui-64** command. This command specifies the prefix as 2001:db8:cafe:1. The **eui-64** option uses ISATAP's EUI-64 format to determine the 64-bit Interface ID. The first 32 bits are the ISATAP-reserved OUI plus fe, 0000:5efe. The low-order 32 bits are the IPv4 address converted to hexadecimal, 0a0a:0a01.

The same process occurred for the link-local address for Tunnel 0. The link-local address uses the reserved prefix fe80::/10 plus the same ISATAP EUI-64 Interface ID created for the global unicast address, 0000:5efe: 0a0a:0a01. The global unicast and link-local address for Tunnel 0 are also displayed by the **show ipv6 interface tunnel 0** command in Example A-12.

Using the **show interface tunnel 0** command in Example A-13, verify that the tunneling type for the Tunnel 0 interface is ISATAP.

Example A-13 *Verifying R1's Tunnel Protocol*

```
R1# show interface tunnel 0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  MTU 1514 bytes, BW 9 Kbit, DLY 500000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 10.10.10.1 (FastEthernet0/0), destination UNKNOWN
  Tunnel protocol/transport IPv6 ISATAP
<Rest of output omitted for brevity>
```

Now that R1's configuration has been examined, see how host X uses the ISATAP tunnel to receive its IPv6 configuration information. Similar to 6to4 tunneling, ISATAP automatically creates its IPv6 address from an IPv4 address but, in addition, uses the services of Router Advertisements for Stateless Address Autoconfiguration.

Using Figure A-10, you can see the process of how host X gets its IPv6 global unicast address, prefix length, and IPv6 default gateway address.

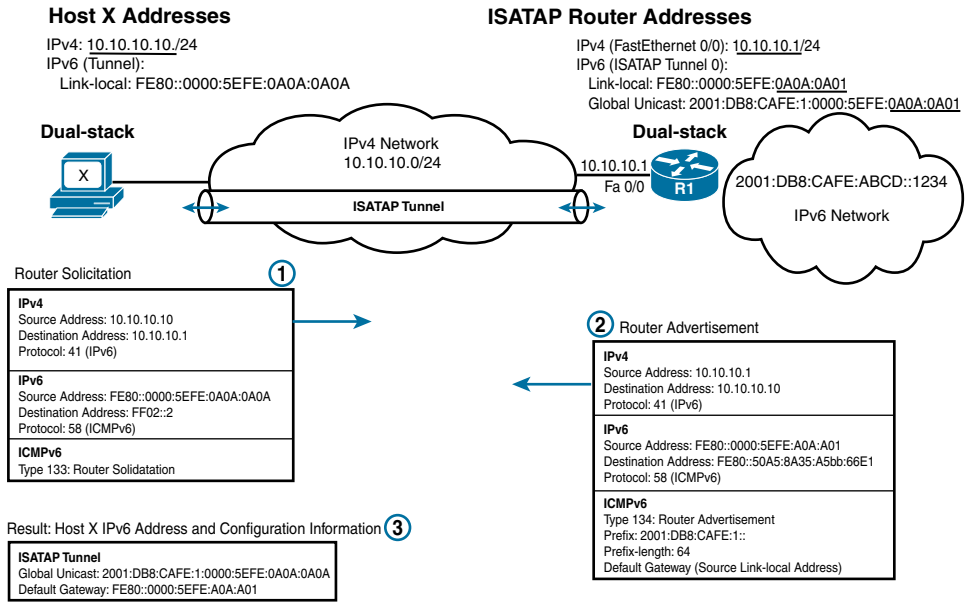


Figure A-10 *Router Solicitations and Router Advertisements over ISATAP*

Host X has been configured, either statically or dynamically, with an IPv4 address of 10.10.10.10/24 and enabled for ISATAP. ISATAP is supported in Windows and Linux operating systems, but not Mac OS. Part of host X's ISATAP tunnel configuration contains the IPv4 address of the other end of the tunnel. In this case, it is 10.10.10.1, the IPv4 address of Router R1. Host X's interface automatically creates for itself an IPv6 link-local address using the ISATAP format, as shown in the following steps:

Step 1. Using this link-local address, host X sends a Router Solicitation message over the ISATAP tunnel to the all-routers multicast address ff02::2. The RS message is encapsulated in an IPv4 packet and sent over the ISATAP tunnel to Router R1.

Step 2. R1 responds with a Router Advertisement message including the IPv6 prefix, prefix length, and default gateway address. The ICMPv6 Router Advertisement is encapsulated in an IPv4 packet before being sent over the tunnel to host X.

Step 3. Host X now has what it needs to create a routable IPv6 address and to communicate with an IPv6 default gateway. The Router Advertisement from R1 includes the prefix and prefix length 2001:db8:cafe:1::/64. Using the ISATAP format, host X creates its 64-bit Interface ID by appending the prefix with 0000:5efe and its IPv4 address in hexadecimal, 0a0a:0a0a. As a result, host X's global unicast address is 2001:db8:cafe:1:0000:5efe:0a0a:0a0a/64.

The default gateway address for host X is the link-local address of R1's FastEthernet 0/0 interface. This is the link-local address that R1 used to send its Router Advertisement

to host X, as shown in Figure A-10. Host X now has an IPv6 global unicast address and an IPv6 default gateway address on its ISATAP tunnel interface. Any packets for IPv6 networks that host X needs to reach can now be sent over this tunnel to Router R1.

Note To enable ISATAP on client operating systems, check the documentation or, better yet, search the Internet for information. For Windows, it involves using **netsh** at the command (DOS) prompt. At the time of this writing, Apple's Mac OS does not support ISATAP. It is available in prealpha version for Mac OS 10.6.4 or later, including 10.7.x (Lion), but is still in the development stages.

At this point, host X has what it needs to reach IPv6 devices across the IPv4 infrastructure. Host X has its necessary IPv6 addressing information, including an IPv6 default gateway address. At the bottom of Figure A-8, notice the encapsulation of an IPv6 packet for 2001:db8:cafe:abcd::1234 being sent across the ISATAP tunnel to Router R1.

Similar to 6to4 tunnels, ISATAP tunnel endpoints determine the IPv4 address of the other end of the tunnel, the destination IPv6 address of the packet. The IPv4 address is encoded in the last 32 bits of the IPv6 address, enabling automatic IPv6-in-IPv4 tunneling. The low-order 32 bits in the IPv6 address contain the necessary IPv4 addressing information; otherwise, the default gateway is used. The bottom part of Figure A-8 shows the encapsulation for a packet from host X to 2001:db8:cafe:abcd::1234.

Appendix B

IPv6 Command Quick Reference

Cisco IOS Commands

Addressing Commands

Global Unicast Address and Unique Local Unicast Addresses

To configure a global unicast address or unique local unicast address on an interface:

- Router(config-if)# **ipv6 address** *ipv6-address/prefix-length*

To configure a global unicast address or unique local unicast address on an interface using EUI-64 to generate the Interface ID:

- Router(config-if)# **ipv6 address** *ipv6-prefix/prefix-length* **eui-64**

To enable IPv6 processing on an interface without assigning an explicit IPv6 address to the interface:

- Router(config-if)# **ipv6 unnumbered** *interface-type interface-number*

To enable a router's interface to dynamically create an IPv6 address using SLAAC:

- Router(config-if)# **ipv6 address autoconfig**

Link-Local Unicast Address

To configure a link-local unicast address on an interface:

- Router(config-if)# **ipv6 address** *ipv6-address* **link-local**

To generate a link-local unicast address on an interface without requiring a global unicast or unique local unicast address:

- Router(config-if)# **ipv6 enable**

General Prefix

To define a general prefix:

- Router# **ipv6 general-prefix** *prefix-name ipv6-prefix/prefix-length*

To display the general prefix information:

- Router# **show ipv6 general-prefix**

DNS host commands

To configure static host names for IPv6 addresses:

- Router(config)# **ipv6 host** *name* [*port*] *ipv6-address1* [*ipv6-address2... ipv6-address4*]

To specify the address of one or more DNS servers for name and address resolution:

- Router(config)# **ip name-server** *server-address1* [*server-address2... server-address6*]

Verifying Address Information

To display a list of IPv6 addresses for all interfaces:

- Router# **show ipv6 interface brief** [*interface-type interface-number*]

To display detailed IPv6 address information including multicast group membership and RA parameters for an interface:

- Router# **show ipv6 interface** *interface-type interface-number*

ICMPv6 Router Advertisement Commands

Enabling ICMPv6 Router Advertisements

To enable ICMPv6 Router Advertisements to be sent out Ethernet interfaces:

- Router(config)# **ipv6 unicast-routing**

Modifying Router Advertisement Parameters on the Interface

To modify the RA message interval:

- Router(config-if)# **ipv6 nd ra interval** { *maximum-secs [minimum-secs]* | **msec** *maximum-ms [minimum-ms]* }

To enable the sending of solicited unicast RA messages:

- Router(config-if)# **ipv6 nd ra solicited unicast**

To suppress the sending of RA messages:

- Router(config-if)# **ipv6 nd ra suppress [all]**

To configure IPv6 prefix information that is included in the RA message (complete syntax):

- Router(config-if)# **ipv6 nd prefix** { *ipv6-prefix/prefix-length* | **default** } [**no-advertise** | [*valid-lifetime preferred-lifetime* [**off-link** | **no-rtr-address** | **no-autoconfig** | **no-onlink**]]] **at** *valid-date* | *preferred-date* [**off-link** | **no-rtr-address** | **no-autoconfig**]

To configure the Valid Lifetime and Preferred Lifetime of the IPv6 prefix included in the RA message:

- Router(config-if)# **ipv6 nd prefix** *ipv6-prefix/prefix-length* [*valid-lifetime*] [*preferred-lifetime*]

To set the RA's Address Autoconfiguration flag (A flag) to 0 (default is 1). This command disables the use of SLAAC for the specified prefix:

- Router(config-if)# **ipv6 nd prefix** *prefix/prefix-length* **no-autoconfig**

To set the RA's Other Configuration flag (O flag) to 1 (default is 0). This command suggests the use of a stateless DHCPv6 server:

- Router(config-if)# **ipv6 nd other-config-flag**

To set the RA's Managed Address Configuration flag (M flag) to 1 (default is 0). This command suggests the use of a stateful DHCPv6 server:

- Router(config-if)# **ipv6 nd managed-config-flag**

To modify the Default Router Preference (DRP):

- Router(config-if)# **ipv6 nd router-preference { high | medium | low }**

To include the DNS server address in the RA message:

- Router(config-if)# **ipv6 nd ra dns server** *ipv6-address seconds*

Verifying Router Advertisements

To verify ICMPv6 Router Advertisement information sent on an interface:

- Router# **show ipv6 interface** *interface-type interface-number*

To display the RA, RS, and other ICMPv6 Neighbor Discovery messages sent and received:

- Router# **debug ipv6 nd**

Configuring a DHCPv6 Server

Stateless DHCPv6 Configuration Pool Commands

To create a DHCPv6 pool and enter DHCPv6 pool configuration mode:

- Router(config)# **ipv6 dhcp pool** *poolname*

To specify the IPv6 DNS server available to a DHCPv6 client:

- Router(config-dhcp)# **dns-server** *ipv6-address*

To specify a domain name for the DHCPv6 client:

- Router(config-dhcp)# **domain-name** *domain*

Stateful DHCPv6 Configuration Pool Commands

To create a DHCPv6 pool and enter DHCPv6 pool configuration mode:

- Router(config)# **ipv6 dhcp pool** *poolname*

To specify the IPv6 prefix that will be used to allocate IPv6 addresses:

- Router(config-dhcp)# **address prefix** *ipv6-prefix/prefix-length* [**lifetime** { *valid-lifetime preferred-lifetime* | **infinite** }]

To specify the IPv6 DNS server available to a DHCPv6 client:

- Router(config-dhcp)# **dns-server** *ipv6-address*

To specify a domain name for the DHCPv6 client:

- Router(config-dhcp)# **domain-name** *domain*

Associating the DHCPv6 Pool to an Interface

To enable DHCPv6 service on an interface. The **rapid-commit** option enables the use of the two-message exchange for address allocation and other configuration.

- Router(config-if)# **ipv6 dhcp server** *poolname* [**rapid-commit**]

DHCPv6 Relay

To specify a destination address to which client packets are forwarded and enable DHCPv6 relay service on the interface:

- Router(config-if)# **ipv6 dhcp relay destination** *ipv6-address* [*interface-type interface-number*]

Verifying DHCPv6 Information

To display the DHCPv6 DUID:

- Router# **show ipv6 dhcp**

To display the DHCPv6 poolname and rapid commit option:

- Router# **show ipv6 dhcp interface** *interface-type interface-number*

IPv6 Access Control Lists

Configuring IPv6 ACLs

To specify the name of an IPv6 ACL and enter ACL configuration mode:

- Router(config)# **ipv6 access-list** *access-list-name*

To apply an IPv6 ACL to an interface:

- Router(config-if)# **ipv6 traffic-filter** *access-list-name* {**in** | **out**}

Note The permit and deny options in IPv6 address configuration mode are numerous and vary depending on the protocol.

Verifying IPv6 ACLs

To display IPv6 ACL information:

- Router# **show ipv6 access-list**

Static Routes, Displaying the Routing Table, and CEF for IPv6

Static Routes

To enable forwarding of IPv6 packets transiting the router:

- Router# **ipv6 unicast-routing**

To configure an IPv6 static route:

- Router(config)# **ipv6 route** *ipv6-prefix*/*prefix-length* {*ipv6-address* | *interface-type interface-number*} [*next-hop-address*]

The complete syntax to configure an IPv6 static route:

- **ipv6 route** [**vrf** *vrf-name*] *ipv6-prefix*/*prefix-length* {*ipv6-address* | *interface-type interface-number* [*ipv6-address*]} [**nexthop-vrf** [*vrf-name1* | **default**]] [*administrative-distance*] [*administrative-multicast-distance* | **unicast** | **multicast**] [*next-hop-address*] [**tag** *tag*] [**name** *name*]

Verifying Static Routes

To display the IPv6 routing table:

- Router# **show ipv6 route**

To display only static routes in the IPv6 routing table:

- Router# **show ipv6 route static**

To display a summary of the IPv6 routing table:

- Router# **show ipv6 route summary**

To display static route information:

- Router# **show ipv6 static** [**detail**]

CEF for IPv6

To enable CEF for IPv6:

- Router# **ipv6 unicast-routing**
- Router# **ipv6 cef**

To verify CEF for IPv6:

- Router# **show ipv6 cef**

EIGRP for IPv6

Classic EIGRP for IPv6

To create an EIGRPv6 routing process:

- Router(config)# **ipv6 router eigrp** *autonomous-system-number*

To configure the EIGRPv6 Router ID:

- Router(config-rtr)# **eigrp router-id** *router-id*

To suppress EIGRPv6 hello messages and routing updates from being sent out an interface:

- Router(config-rtr)# **passive-interface** *interface-type interface-number*

To enable EIGRPv6 directly on the interface:

- Router(config-if)# **ipv6 eigrp** *autonomous-system-number*

To advertise a summary route into the EIGRPv6 routing domain:

- Router(config-if)# **ipv6 summary-address eigrp** *autonomous-system-number prefix/prefix-length*

EIGRP Named Mode

To configure the EIGRP virtual instance name:

- Router(config)# **router eigrp** *virtual-instance-name*

To create an instance of EIGRP and enter address family configuration mode for the specific protocol (IPv4 or IPv6):

- Router(config-router)# **address-family** *address-family* **unicast autonomous-system** *autonomous-system-number*

To configure the 32-bit EIGRP Router ID:

- Router(config-router-af)# **eigrp router-id** *router-id*

To enter address-family interface configuration mode for a specific interface:

- Router(config-router-af)# **af-interface** *interface-type interface-number*

To suppress EIGRP message on an interface:

- Router(config-router-af-interface)# **passive-interface**

To disable EIGRP for IPv6 on an interface:

- Router(config-router-af-interface)# **shutdown**

EIGRP for IPv6 Verification Commands

To verify EIGRPv6 neighbor table and adjacencies:

- Router# **show ipv6 eigrp neighbors**

To display the EIGRPv6 topology table:

- Router# **show ipv6 eigrp topology**

To display EIGRPv6 routes in the IPv6 routing table:

- Router# **show ipv6 route eigrp**

To display EIGRPv6 routing protocol information:

- Router# **show ipv6 protocols**

To display the number of EIGRPv6 packets sent and received:

- Router# **show ipv6 eigrp traffic**

To display EIGRPv6 interface information:

- Router# **show ipv6 eigrp interfaces**

OSPFv3

Configuring Traditional OSPFv3

To enable the OSPFv3 routing process:

- R1(config)# **ipv6 router ospf** *process-id*

To configure the OSPFv3 Router ID:

- Router(config-rtr)# **router-id** *router-id*

To suppress OSPF hello messages and other OSPF messages from being sent out an interface:

- Router(config-rtr)# **passive-interface** *interface-type interface-number*

To distribute a default route to other routers in the OSPFv3 domain.

- Router(config-rtr)# **default-information originate**

To define an area as a stub area:

- Router(config-rtr)# **area** *area* **stub**

To define an area as a totally stubby area on an ABR:

- Router(config-rtr)# **area** *area* **stub** [**no-summary**]

To enable OSPFv3 directly on the interface:

- Router(config-if)# **ipv6 ospf** *process-id* **area** *area-id*

Verifying Traditional OSPFv3

To display IPv6 prefixes in the IPv6 routing table learned via OSPFv3:

- Router# **show ipv6 route ospf**

To display LSAs for IPv6 prefixes in the OSPFv3 LSDB:

- Router# **show ipv6 ospf database**

To display OSPFv3 (for IPv6) routing protocol information:

- Router# **show ipv6 protocols**

To display OSPFv3 (for IPv6) neighbor adjacencies:

- Router# **show ipv6 ospf neighbor**

To display OSPFv3 (for IPv6) interface information:

- Router# **show ipv6 ospf interface**

Configuring OSPFv3 with Address Families

To enter router configuration mode for OSPFv3:

- Router(config)# **router ospfv3** *process-id*

To enter address family configuration mode for the specific protocol (IPv4 or IPv6):

- Router(config-router)# **address-family** [**ipv4** | **ipv6**] **unicast**

To configure the 32-bit OSPFv3 Router ID:

- Router(config-router-af)# **router-id** *router-id*

To suppress OSPFv3 hello messages on an interface:

- Router(config-router-af)# **passive-interface** *interface-type interface-number*

To advertise the default static route into the routing domain:

- Router(config-router-af)# **default-information originate**

To define the area as a stub area:

- Router(config-router-af)# **area** *area* **stub**

To define the area as a totally stubby area on an ABR:

- Router(config-router-af)# **area** *area* **stub no-summary**

To enable OSPFv3 on the interface:

- Router(config-if)# **ospfv3** *process-id* [**ipv4** | **ipv6**] **area** *area-id*

Verifying OSPFv3 with Address Families

To display IPv4 prefixes in the IPv4 routing table learned via OSPFv3:

- **show ip route ospfv3**

To display IPv6 prefixes in the IPv6 routing table learned via OSPFv3:

- **show ipv6 route ospf**

To display OSPFv3 neighbor adjacencies for the IPv4 and IPv6 address families:

- **show ospfv3 neighbor**

To display LSAs in the OSPFv3 LSDB for the IPv4 and IPv6 address families:

- **show ospfv3 database**

For a complete listing of Cisco IOS IPv6 commands see *Cisco IOS IPv6 Command Reference*, www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book/ipv6-a1.html.

Host Operating System Commands

Windows OS

General Commands

Ping and traceroute:

- Windows> **ping** *ipv6-address*
- Windows> **traceroute** *ipv6-address*

Display IPv6 addresses:

- Windows> **ipconfig**
- Windows> **ipconfig /all**

Display IPv6 routing table:

- Windows> **netstat -rn**
- Windows> **netsh interface ipv6 show route**

Display IPv6 connections:

- Windows> **netstat -p ipv6**

Display IPv6 Neighbor Cache:

- Windows> **netsh interface ipv6 show neighbors**

Clear IPv6 Neighbor Cache:

- Windows> **netsh interface ipv6 delete neighbors**

Display IPv6 multicast group membership:

- Windows> **netsh interface ipv6 show joins**

Display IPv6 global parameters including default hop limit and Neighbor Cache limit:

- Windows> **netsh interface ipv6 show global**

Interface Addresses Information

Display interface SLAAC address states and lifetimes:

- Windows> **netsh interface ipv6 show addresses** [*zone-id*]

Display zone ID, interface index, MTU, and address state:

- Windows> **netsh interface ipv6 show interfaces**

Display IPv6 interfaces, privacy parameters, and lifetimes:

- Windows> **netsh interface ipv6 show privacy**

Display IPv6 default policy table:

- Windows> **netsh interface ipv6 show prefixpolicies**

SLAAC Interface ID

Modify the default behavior to use EUI-64 for the Interface ID:

- Windows> **netsh interface ipv6 set global randomizeidentifiers=disabled store=active**
- Windows> **netsh interface ipv6 set global randomizeidentifiers=disabled store=persistent**

Re-enable the default behavior to use the privacy extension for the Interface ID:

- Windows> **netsh interface ipv6 set global randomizeidentifiers=enabled store=active**
- Windows> **netsh interface ipv6 set global randomizeidentifiers=enabled store=persistent**

Modify the default behavior to disable the use of temporary addresses:

- Windows> **netsh interface ipv6 set privacy state=disabled store=active**
- Windows> **netsh interface ipv6 set privacy state=disabled store=persistent**

Re-enable the default behavior to use temporary addresses:

- Windows> **netsh interface ipv6 set privacy state=enabled store=active**
- Windows> **netsh interface ipv6 set privacy state=enabled store=persistent**

Linux OS

General Commands

Ping and traceroute:

- Linux$ **ping6** *ipv6-address*
- Linux$ **traceroute6** *ipv6-address*

Configure an IPv6 address on an interface:

- Linux$ **ip -6 addr add** *ipv6-address/prefix-length* **dev** *interface*
- Linux$ **ifconfig** *interface* **inet6 add** *ipv6-address/prefix-length*

Display IPv6 addresses:

- Linux$ **ifconfig –a**
- Linux$ **ip -6 addr**

Display IPv6 addresses including lifetimes:

- Linux$ **ip -6 address show**

Display IPv6 routing table:

- Linux$ **netstat -rnA inet6**

Display IPv6 default gateway:

- Linux$ **ip -6 route show**

Display IPv6 connections:

- Linux$ **netstat -a inet6**

Display IPv6 Neighbor Cache:

- Linux$ **ip -6 neigh show**

Clear IPv6 Neighbor Cache:

- Linux$ **ip -6 neigh flush**

Display IPv6 multicast group membership:

- Linux$ **ip -6 maddr show**
- Linux$ **netstat -g**

Display lifetimes for a SLAAC address:

- Linux$ **ip -6 addr show**

Address Configuration Commands

Configure IPv6 address manually:

- Linux$ **ifconfig** *interface* **inet6 add** *ipv6-address/prefix-length*

Configure IPv6 default gateway manually:

- Linux$ **route -A inet6 add default gw** *ipv6-address*

Enable privacy extension for an interface:

- Linux$ **sysctl net.ipv6.conf.***interface***.use_tempaddr=2**

Mac OS X

General Commands

Ping and traceroute:

- Mac$ **ping6** *ipv6-address*
- Mac$ **traceroute6** *ipv6-address*

Display IPv6 addresses:

- Mac$ **ifconfig -a inet6**

Display IPv6 addresses with lifetimes:

- Mac$ **ifconfig -L inet6**

Display IPv6 routing table:

- Mac$ **netstat -rnf inet6**

Display IPv6 connections:

- Mac$ **netstat -f inet6**

Display IPv6 Neighbor Cache:

- Mac$ **ndp -a**

Clear IPv6 Neighbor Cache:

- Mac$ **ndp -c**

Display IPv6 multicast group membership:

- Mac$ **netstat -g**

Address Configuration Commands

Enable privacy extension for an interface:

- Mac$ **sysctl net.inet6.ipv6.use_tempaddr=1**

Appendix C

Answers to Review Questions

Chapter 1

1. Access to IPv6-only customers: Some networks will be accessible only via IPv6.

   IPv4 address depletion: Four out of five RIRs have run out of IPv4 addresses. For the Internet to continue to evolve and grow, it is necessary to transition to IPv6.

   Better performance: NAT creates latency. Some content providers are seeing better performance with IPv6.

   Securing your current network: Host operating systems are already IPv6-enabled. If your network is not secured for IPv6, it is vulnerable to IPv6 MITM and DoS attacks.

2. They implemented CIDR, NAT, and private RFC 1918 IPv4 addressing.

3. NAT devices must perform IPv4, TCP, and UDP checksum recalculations.

   IPv4 addresses within the ICMP payload must also be translated.

   NAT cannot be used with IPsec in transport mode.

   NAT breaks end-to-end reachability.

   NAT creates latency.

4. NAT is not considered security. NAT requires state, and this statefulness means that any device performing NAT must also act as a stateful firewall. This appears to be security but is not.

5. Although it was never recognized as IPv5, the experimental Internet Stream Protocol was encapsulated in IPv4 using IP Protocol Number 5.

6. The IAB recommended OSI's Connectionless-mode Network Protocol (CLNP) as the replacement for IPv4. The CLNP proposal was called TCP and UDP with Bigger Addresses (TUBA).

7. The IESG recommended and the IAB selected Simple Internet Protocol Plus (SIPP) but with an address size of 128 bits.
8. IPv6 is more secure than IPv4.

   IPv6 is less secure than IPv4.

   There is a date set for IPv6 to replace IPv4.

   IPv6 isn't necessary.

   IPv6 is too complex.

   IPv6 improves QoS.

Chapter 2

1. A. 2

   B. C

   C. F

   D. D

   E. 0

   F. A
2. A. 1110

   B. 0011

   C. 1000

   D. 1011

   E. 0001

   F. 1001
3.

   - Global unicast address B, F
   - Link-local address A, G
   - Unspecified address C, E
   - Solicited-node multicast address D, G

4.

- Neighbor Solicitation B
- Neighbor Advertisement C
- Router Solicitation D
- Router Advertisement A

5.

- Method 1—SLAAC B, C, E
- Method 2—SLAAC and stateless DHCPv6 B, C, D
- Method 3—Stateful DHCPv6 A, B, F (C is also a correct answer here. As we will discuss later, it is possible for a device to use stateful DHCPv6 and obtain another GUA using SLAAC.)

Chapter 3

1. The IPv4 IHL field specifies the length of the IPv4 header, including any options or padding. Whereas the IPv4 header is variable in length, the IPv6 header has a fixed length of 40 bytes, so an IHL field is unnecessary.
2. The IPv4 Total Length field shows the number of bytes in the IPv4 header and the encapsulated data. The IPv6 Payload Length field only shows the number of bytes following the main IPv6 header, including any extension headers.
3. IPv4 source and destination addresses are 32-bit addresses, whereas the IPv6 addresses are 128 bits in length. In both protocols, the source address must be a unicast address. The destination address can be a unicast, multicast, anycast, or, in the case of IPv4, broadcast address.
4. With IPv4, both the source and intermediate devices, routers, can perform fragmentation. The IPv4 header provides for information to perform the fragmentation. After fragmentation, the destination device reassembles the fragmented packets. With IPv6, only the source device can perform fragmentation. Routers do not fragment IPv6 packets. Instead, they send an ICMPv6 Packet Too Big message back to the source, along with the MTU of the egress link that caused the message.
5. With IPv4 the checksum in UDP is optional, but when carried over IPv6, the UDP checksum is mandatory.
6. The rarely used IPv4 Options and Padding fields cause the IPv4 header to be variable in length. A fixed 40-byte IPv6 header means more efficient processing. IPv6 uses optional extension headers to add flexibility and allow future enhancements to IPv6 while still remaining the fixed main IPv6 header.
7. 0x86dd

Chapter 4

1. 2001:db8:cab0:234:34:4::
2. 2001:db8:cab::1:0
3. 2001:db8:cab:1234:230:1200:34::
4. fd00::1234:0:0:0
5. 2001:db8::1234:0:0:1000
6. 2001:0db8:0cab:0000:0000:0000:0000:0001
7. 2001:0db8:0000:0000:0234:0000:0000:0000
8. Prefix 2001:db8:80f:f425::/64
9. Prefix 2001:db8:80f:f425::/64
10. Prefix fe80::/64
11. Prefix 2001:db8:80f::/48
12. Prefix 2001:db8:80f::/48
13. Prefix 2001:db8::/32
14. Global Routing Prefix, Subnet ID, and Interface ID
15. 2000 to 3fff
16. Link-local address
17. fe80 to febf
18. For a device to be an "IPv6-enabled" device, it must have an IPv6 link-local address.

    Link-local addresses are not routable off the link.

    Link-local addresses only have to be unique on the link.

    There can be only one link-local address per interface.

    Windows, Mac OS, and Linux client operating systems dynamically (automatically) create their own IPv6 addresses upon startup.

    Link-local addresses can be manually configured.
19. Unspecified unicast address
20. It cannot be assigned to a physical interface.

    An unspecified source address indicates the absence of an address.

    It cannot be used as a destination address.

    A router never forwards a packet with an unspecified source address.

21. Unique local address
22. fc00 to fdff
23. NAT for IPv4 is stateful and was intended to help solve the problem of IPv4 address depletion. NAT is used with IPv6 as a mechanism to avoid renumbering, facilitate multihoming, make configurations homogenous, hide internal network details, and provide simple security. However, it is not the security a stateful firewall provides.
24. ff
25. Solicited-node multicast

Chapter 5

1. Global Routing Prefix, Subnet ID, and Interface ID
2. A /64 subnet prefix results in a 64-bit Interface ID, which allows devices to create their own GUA addresses using SLAAC.
3. 3 is the first three hextets, representing the Global Routing Prefix. 1 is the single hextet representing the Subnet ID. 4 is the last four hextets, representing the Interface ID.
4. a. Global Routing Prefix 2001:0db8:cafe, Subnet ID 0001, Interface ID 000a:000b:000c:000d

   b. Global Routing Prefix 2001:0db8:cafe, Subnet ID a100, Interface ID 0000:0000:0002:000d

   c. Global Routing Prefix 2001:0db8:cafe, Subnet ID 000a, Interface ID 0037:0000:0000:0009

   d. Global Routing Prefix 2001:0db8:cafe, Subnet ID 0000, Interface ID 000a:000b:000c:000d

   e. Global Routing Prefix 2001:0db8:cafe, Subnet ID 0001, Interface ID 0000:0000:0000:0100

   f. Global Routing Prefix 2001:0db8:0000, Subnet ID 0100, Interface ID 000a:000b:000c:000d

   g. Global Routing Prefix 2001:0db8:0000, Subnet ID 0000, Interface ID 0000:0000:0000:0100
5. This creates 16 subnets:

   2001:db8:cafe:0000::/52

   2001:db8:cafe:1000::/52

   2001:db8:cafe:2000::/52

   2001:db8:cafe:3000::/52

   2001:db8:cafe:4000::/52

2001:db8:cafe:5000::/52

2001:db8:cafe:6000::/52

2001:db8:cafe:7000::/52

2001:db8:cafe:8000::/52

2001:db8:cafe:9000::/52

2001:db8:cafe:a000::/52

2001:db8:cafe:b000::/52

2001:db8:cafe:c000::/52

2001:db8:cafe:d000::/52

2001:db8:cafe:e000::/52

2001:db8:cafe:f000::/52

6. This creates 256 subnets. The following are the first 3 of them and the last 3 of them:

   2001:db8:cafe:0000::/56

   2001:db8:cafe:0100::/56

   2001:db8:cafe:0200::/56

   ….

   2001:db8:cafe:fd00::/56

   2001:db8:cafe:fe00::/56

   2001:db8:cafe:ff00::/56

7. /56 Global Routing Prefix
8. /40 Global Routing Prefix
9. d. 2001:db8:face:b00c::2/127

   e. 2001:db8:face:b00c::3/127

10. The address remains with the customer if the customer changes service providers. No renumbering is necessary. Another advantage is that the end site can usually get a larger address space.

Chapter 6

1. False. An IPv6-enabled device only needs to have a link-local address. It doesn't have to have a global unicast address.
2. True. Link-local addresses are limited to the local link, or subnet. Routers must not forward any packets with source or destination link-local addresses to other links.

3. fe80 to febf
4. EUI-64 or a randomly generated 64-bit value
5. 1. Split the MAC address between the 24-bit OUI and the 24-bit Device Identifier.
   2. Insert 16 bits, fffe in hexadecimal, between the OUI and the Device Identifier.
   3. Flip the seventh bit, which changes the value of the second hexadecimal digit.
6. EUI-64
7. It is used to associate a link-local address with a specific interface.
8. It makes it easier to remember and recognize the link-local address. This can be helpful for recognizing default gateway addresses, analyzing routing table entries, and examining routing protocol messages. Using the same link-local address on all of the router's interfaces is practical when the router has only distribution links and is not used as a default gateway address for devices.
9. Devices use Duplicate Address Detection (DAD). DAD uses ICMPv6 Neighbor Solicitation messages to ask other devices for the Ethernet MAC address of the link-local address assigned to the interface. If no devices on the link respond with a Neighbor Advertisement message, the device is assured that the address is unique. This is equivalent to a gratuitous ARP request in IPv4. DAD is used for all unicast addresses, regardless of how they are assigned to the interface.
10. Devices dynamically receive the default gateway address from the ICMPv6 Router Advertisement message. The device uses the source IPv6 address (link-local address) of the RA message as the default gateway address.
11. No. Devices can only dynamically receive the default gateway address from the ICMPv6 Router Advertisement message.
12. This command dynamically creates a link-local address on an interface without requiring a global unicast or unique local unicast address.
13. Cisco IOS asks for the output interface. This is because the link-local address could exist on any of its links.

Chapter 7

1. ff00::/8
2. Link-local scope
3. Site-local scope
4. ff02::1
5. ff02::2
6. A multicast address is mapped to an Ethernet MAC address. This allows the Ethernet NIC to filter the frame.

7. Global unicast addresses, unique local unicast addresses, and link-local unicast addresses
8. Neighbor Solicitation message
9. Yes. This occurs only when the devices have the same lower-order 24 bits in the global unicast, unique local unicast or link-local unicast addresses.
10. Yes. This can occur when the global unicast and link-local unicast address share the same low-order 24 bits. This is common with SLAAC, which can use either EUI-64 or a random 64 bits for both addresses.
11. a. Solicited-node multicast: ff02::1:ffa1:1067

    Ethernet MAC: 33-33-ff-a1-10-67

    b. Solicited-node multicast: ff02::1:ffa1:1067

    Ethernet MAC: 33-33-ff-a1-10-67

    Notice that this is the same solicited-node multicast and MAC address as for the previous GUA address.

    c. Solicited-node multicast: ff02::1:ffbc:7

    Ethernet MAC: 33-33-ff-bc-00-07

    d. Solicited-node multicast: ff02::1:ff00:7000

    Ethernet MAC: 33-33-ff-00-70-00

    e. Solicited-node multicast: ff02::1:ff00:7000

    Ethernet MAC: 33-33-ff-00-70-00

    Notice that this is the same solicited-node multicast and MAC address as for the previous GUA address.
12. Multicast Listener Discovery (MLDv2)
13. MLD snooping

Chapter 8

1. DHCP for IPv4 (DHCPv4) is the only method available.
2. There are three methods:

   Stateless Address Autoconfiguration (SLAAC)

   SLAAC and a stateless DHCPv6 server

   A stateful DHCPv6 server
3. The Router Advertisement message suggests to a device which method it should use to obtain its global unicast address and other configuration information.

4. A device sends a Router Solicitation message to request a Router Advertisement message from the local router. It sends the RS message from either its link-local unicast address or an unspecified address. The destination address is the all-routers multicast address, ff02::2.
5. A router sends a Router Advertisement message to suggest to devices on the link where to obtain their global unicast address and other information. A Cisco router sends the RA message every 200 seconds or in response to a Router Solicitation message. The router sends the RA message from its link-local address. The destination address is the all-devices multicast address, ff02::1.
6. A. M flag

   B. A flag

   C. O flag
7. A. A flag = 1; O flag = 0; M flag = 0

   B. A flag = 1; O flag = 1; M flag = 0

   C. A flag = 0; O flag = 0; M flag = 1. If the O flag is set to 1, it has no effect because all addressing information is obtained from a DHCPv6 server. If the A flag is set to 1, the device still receives its GUA address from a stateful DHCPv6 server, but it may also create another address using SLAAC.
8. A DHCPv4 server provides the default gateway address, whereas a DHCPv6 server does not. With IPv6, that information can only be dynamically obtained from the Router Advertisement message.
9. DUID (DHCP Unique Identifier)
10. 1. Client sends SOLICIT

    2. Server sends REPLY

    3. Client sends REQUEST

    4. Server sends REPLY

Chapter 9

1. d. Every 200 seconds
2. a. Address Autoconfiguration flag (A flag)
3. b. How long this router should be used as a default gateway
4. d. By using the prefix in the RA with the L flag set to 1
5. EUI-64
6. b. Randomized Interface IDs

   d. Temporary addresses

7. a. Invalid address
   b. Preferred address
   c. Valid address
   d. Deprecated address
   e. Tentative address
8. Duplicate Address Detection (DAD)
9. Default Router Preference (DRP)
10. Its link-local address
11. A temporary address
12. A preferred address

Chapter 10

1. b. A flag and O flag
2. d. A flag only
3. **ipv6 nd other-config-flag**
4. c. DNS server address
   d. Domain name
5. DUID (DHCP Unique Identifier)
6. IAID (Interface Association Identifier)
7. b. REPLY
   d. SOLCIT
8. When the DHCPv6 clients and server are on different networks
9. The exit interface at the end of the **ipv6 dhcp relay destination** command is required. For example: R1(config-if)**# ipv6 dhcp relay destination fe80::55 g 0/1**
10. The ff02::1:2 all-DHCPv6 server multicast address with link-local scope and the ff05::1:3 all-DHCPv6 server multicast address with site-local scope

Chapter 11

1. a. A flag and M flag
2. **ipv6 nd managed-config-flag**
3. **ipv6 nd prefix** *ipv6-prefix/prefix-length* **no-autoconfig**
4. Because there is no prefix sent, there is also no L flag (On-Link flag) associated with the prefix. This means hosts on the subnet won't know their local subnet prefix and will need to send all packets to the default gateway. This can also generate a large number of ICMPv6 Redirect messages coming from the router.
5. b. address prefix 2001:db8:face:b00c:1eaf::/80
6. b. With an IOS router as a stateful DHCPv6 server you specify which addresses to *include*, and all other addresses are *excluded*.

   d. With an IOS router as a stateful DHCPv4 server you specify which addresses to *exclude*, and all other addresses are *included*.
7. d. Requesting Router
8. a. Delegating Router
9. d. 2001:db8:beef:99::99/64
10. e. 256 /48 prefixes

    The 256 prefixes the DR can provide range from 2001:db8:ab00::/48 to 2001:db8:abff::/48.

Chapter 12

1. a. Informational message

   b. Error message

   c. Informational message

   d. Informational message

   e. Error message

   f. Error message

   g. Informational message

   h. Informational message
2. f. Destination Unreachable
3. c. It drops the packet and sends an ICMPv6 Packet Too Big message back to the source.
4. a. Time Exceeded

5. e. Parameter Problem
6. When a device has both a public and a temporary GUA address created using SLAAC, it uses its temporary GUA address whenever it initiates communications and the destination address is a GUA address.
7. A device uses its link-local address when pinging a link-local address of another device.
8. b. The outgoing interface
9. d. Sequence
10. c. Identifier
11. The same link-local address can exist on any network, as long as it is unique. The exit interface tells the device on which interface to send the ping. The Windows **ping** command only requires the exit interface when the device has multiple interfaces. Cisco IOS, Linux, and Mac OS always require an exit interface for pinging a link-local address.

Chapter 13

1. d. Neighbor Advertisement
2. a. Router Solicitation
3. c. Neighbor Solicitation
4. e. Redirect
5. b. Router Advertisement
6. c. All-IPv6 routers multicast address (ff02::2)
7. b. All-IPv6 devices multicast address (ff02::1)
8. a. Solicited-node multicast address
9. a. Source Link- Layer Address
10. a. Source Link- Layer Address
    c. Prefix Information
    e. MTU
11. d. Reachable
12. a. Stale
13. c. Duplicate Address Detection
14. d. Neighbor Solicitation
15. The ARP Request is an Ethernet broadcast, whereas the Neighbor Solicitation message is an Ethernet multicast.

Chapter 14

1. c. Dynamic routing protocols can be configured.

   d. Router Advertisement messages are sent out Ethernet interfaces.

   e. The router forwards packets transiting the router.

   Note: An IPv6 static route can be configured without enabling the router as an IPv6 router.

2. NDp

3. C

4. No. Link-local unicast addresses are not displayed in the IPv6 routing table because they are not routable off the link.

5. Multicast packets that do not explicitly match a more specific multicast route will be discarded.

6. c. The interface type/number of the exit interface.

7. b. 1

8. a. ::/0

9. There are four directly connected routes, code C (one for each interface).

   There are also five local routes, code L (a local route for each interface and another local route for the multicast address ff00::/8).

   There are two static routes, code S.

   In all, there are 11 entries.

10. Steps 1 and 2:

    2001:db8:face:0001 0001 1010 0000::

    2001:db8:face:0001 0001 1011 0000::

    2001:db8:face:0001 0001 1100 0000::

    2001:db8:face:0001 0001 1101 0000::

    57 matching bits, or /57

    Step 3:

    2001:db8:face:0001 0001 1000 0000::

    Summary prefix and prefix length: 2001:db8:face:1180::/57

11. a. **ipv6-unicast routing**

Chapter 15

1. c. Both EIGRPv4 and EIGRPv6
2. b. Both EIGRPv4 named mode and EIGRPv6 named mode
3. c. Both EIGRPv4 and EIGRPv6
4. b. EIGRPv6
5. a. EIGRPv4
6. EIGRPv6 is enabled on an interface via the **ipv6 eigrp** *autonomous-system-number* interface command.
7. Router(config)# **ipv6 router eigrp** *autonomous-system-number*
8. c. Both classic EIGRPv4 and classic EIGRPv6
9. Router(config-router)# d

   Router(config-router-af)# c

   Router(config-router-af-interface)# a

   Router(config-router-af-topology)# b
10. Router(config-router)# **address-family** *address-family* **unicast autonomous-system** *autonomous-system-number*
11. All IPv6 interfaces are automatically enabled.

Chapter 16

1. a. OSPFv2

   c. OSPFv3 with AF
2. b. Traditional OSPFv3

   c. OSPFv3 with AF
3. c. OSPFv3 with AF
4. b. Traditional OSPFv3

   c. OSPFv3 with AF
5. a. OSPFv2
6. b. Traditional OSPFv3

   c. OSPFv3 with AF
7. d. **show ipv6 route ospf**

8. d. **show ipv6 route ospf**
9. a. **show ip route ospf**
10. b. **show ip route ospfv3**

Chapter 17

1. c. /64
2. c. Three digits
3. a. **ipv6 unicast-routing**
4. c. GLBP
5. b. VRRP
6. d. ICMPv6
7. d. AAAA record
8. b. Happy Eyeballs
9. a. **permit icmp any any nd-na** and **permit icmp any any nd-ns**
10. b. NAT64
11. c. Overlay tunnel
12. a. NAT-PT

This page intentionally left blank

Index

Numbers

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Z