O'REILLY®

3rd Edition

IPv6 Essentials

INTEGRATING IPV6 INTO YOUR IPV4 NETWORK

Silvia Hagen

O'REILLY®

IPv6 Essentials

If your organization is gearing up for IPv6, this in-depth book provides the practical information and guidance you need to plan for, design, and implement this vastly improved protocol. Author Silvia Hagen takes system and network administrators, engineers, and network designers through the technical details of IPv6 features and functions, and provides options for those who need to integrate IPv6 with their current IPv4 infrastructure.

The flood of Internet-enabled devices has made migrating to IPv6 a paramount concern worldwide. In this updated edition, Hagen distills more than ten years of studying, working with, and consulting with enterprises on IPv6. It's the only book of its kind.

IPv6 Essentials covers:

- Address architecture, header structure, and the ICMPv6 message format
- IPv6 mechanisms such as Neighbor Discovery, Stateless Address autoconfiguration, and Duplicate Address detection
- Network-related aspects and services: Layer 2 support, Upper Layer Protocols, and Checksums
- IPv6 security: general practices, IPSec basics, IPv6 security elements, and enterprise security models
- Transitioning to IPv6: dual-stack operation, tunneling, and translation techniques
- Mobile IPv6: technology for a new generation of mobile services
- Planning options, integration scenarios, address plans, best practices, and dos and don'ts

"Silvia easily distills complexity out of IPv6 to make it accessible to everyone."

—**Latif Ladid**
President, International IPv6 Forum

"The best vendor-independent IPv6 book available: unpretentious, casual, and powerful."

—**Joe Klein**
CEO Disrupt6, and Security SME for the IPv6 Forum

Silvia Hagen is the owner and CEO of Sunny Connection, a Swiss consulting company that specializes in IPv6 and network and application performance troubleshooting. She has been consulting enterprises in Europe and the United States on IPv6 for more than ten years.

NETWORKING / SYSTEM ADMINISTRATION

US $39.99 CAN $41.99

ISBN: 978-1-449-31921-2

Twitter: @oreillymedia
facebook.com/oreilly

Praise for *IPv6 Essentials, Third Edition*

"Silvia easily distills complexity out of IPv6 to make it accessible to everyone."

— Latif Ladid
President, International IPv6 Forum

"The best vendor-independent IPv6 book available: unpretentious, casual, and powerful."

— Joe Klein
CEO Disrupt6, and Security SME for the IPv6 Forum

"Silvia's ability to capture IPv6 in such detail while considering the business and market drivers really sets the stage for deployment, discovery, and innovation. *IPv6 Essentials* is a go-to resource for all of our students and employees, providing a foundation for the next generation of engineers."

— Erica Johnson
Director, University of New Hampshire InterOperability Lab

"As IPv6 enters mainstream deployment around the world, *IPv6 Essentials* is more essential than ever. This update contains critical new information for any network professional involved in transitioning a network from IPv4 to IPv6."

— Mark Townsley
Cisco Fellow

THIRD EDITION

IPv6 Essentials

Silvia Hagen

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

IPv6 Essentials, Third Edition

by Silvia Hagen

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editors: Mike Loukides and Meghan Blanchette
Production Editor: Kara Ebrahim
Copyeditor: Kiel Van Horn
Proofreader: Rachel Monaghan
Indexer: Ellen Troutman
Cover Designer: Randy Comer
Interior Designer: David Futato
Illustrator: Rebecca Demarest

June 2014: Third Edition

Revision History for the Third Edition:

2014-06-05: First release

See *http://oreilly.com/catalog/errata.csp?isbn=9781449319212* for release details.

ISBN: 978-1-449-31921-2

[LSI]

Table of Contents

Foreword

It is no exaggeration to say that the Internet has become an integral part of the lives of nearly three billion people on the planet. More important, it touches nearly everyone thanks to the ramifications of transactions, information exchange, and other Internet-based applications that produce indirect effects. The original Internet Protocol provided for a maximum of 4.3 billion terminal identifiers (addresses). This limit was stretched using a mechanism called Network Address Translation that permitted multiple parties to use private address space that would not be exposed in the public Internet but rather translated into a shared, publicly routable IPv4 address. The IPv4 address space was exhausted at the Internet Corporation for Assigned Names and Numbers (ICANN) in February 2011, leaving Regional Internet Registries to deal with the allocation of their remaining address space. IPv6 was developed in the mid-1990s and standardized by the Internet Engineering Task Force (IETF). It has provision for 340 trillion trillion trillion addresses. Its implementation has been slow, but two milestones are triggering an increased rate of uptake. One is the running out of the IPv4 address space. The other is the growing demand for Internet addresses to be assigned to mobiles, set-top boxes, automobiles, and literally tens of billions of other programmable devices. This is the so-called *Internet of Things*.

In addition to satisfying what will become an insatiable demand for address space, IPv6 has features that improve the Internet Protocol format for easier processing and provides for additional functionality in the way of configuration convenience and flow management, among other useful properties. Readers will find this book an easily approached guide to IPv6 implementation. That IPv6 must coexist for an uncertain period of time with IPv4 is a given, so attention is drawn to so-called dual-stack implementations. A thorough implementation of IPv6, however, must also demonstrate that the implementation can operate in a pure IPv6 environment in addition to coping with a mixed IPv4/IPv6 environment.

Like many exponential phenomena, IPv6 may well come to surprise us. It has been many years since its development, but there is indication that it is approaching 3% of traffic

on the Internet. While this seems very small, it will grow rapidly if history is any guide, presuming continued compounding growth of need for the larger address space.

Anyone serious about making a career in Internet-related applications and services will be wise to become familiar with this new protocol and its functionality and capability. You have this opportunity before you in Silvia Hagen's work.

—Vint Cerf
Internet Pioneer, Woodhurst, February 2014

Preface

This book is about the next-generation Internet Protocol. We have become familiar with the strengths and weaknesses of IPv4; we know how to design and configure it, and we have learned how to troubleshoot it. And now we have to learn a new protocol? Start from scratch? Not really. The designers of IPv6 have learned a lot from over 15 years of experience with IPv4, and they have been working on the new protocol since the early 1990s. They retained the strengths of IPv4, extended the address space from 32 bits to 128 bits, and added functionality that is missing in IPv4. They developed transition mechanisms that make IPv4 and IPv6 coexist peacefully and that guarantee a smooth transition between the protocols. In fact, this was one of the major requirements for the development of the new protocol version.

So you do not need to forget what you know about IPv4; many things will feel familiar with IPv6. When you get started, you will discover new features and functionalities that will make your life a lot easier. IPv6 has features that you will need in tomorrow's networks—features that IPv4 does not provide.

One of the cool features built into IPv6 is the Stateless Autoconfiguration capability. Haven't we always struggled with IP address assignment? The advent of DHCP made our lives easier, but now we need to maintain and troubleshoot DHCP servers. And when our refrigerator, swimming pool, and heating system as well as our smartphones and the TV set each have IP addresses, will we need a DHCP server at home? Not with Stateless Autoconfiguration. If you have an IPv6-enabled host, you can plug it into your network, and it will configure automatically for a valid IPv6 address. ICMP (Internet Control Message Protocol), which is a networker's best friend, has become much more powerful with IPv6. Many of the new features of IPv6, such as Stateless Autoconfiguration, optimized multicast routing and multicast group management, Neighbor Discovery, Path MTU Discovery, and Mobile IPv6, are based on ICMPv6.

I hope that this book will help you to become familiar with the protocol and provide an easy-to-understand entry point and guide to exploring this new area.

Audience

This book covers a broad range of information about IPv6 and is an excellent resource for anybody who wants to understand or implement the protocol. It is also a good read for people who develop applications. IPv6 offers functionality that we did not have with IPv4, so it may open up new possibilities for applications. Whether you are the owner or manager of a company or an IT department; whether you are a system or network administrator, an engineer, or a network designer; or whether you are just generally interested in learning about the important changes with IPv6, this book discusses economic and strategic aspects as well as technical details. I describe interoperability mechanisms and scenarios that ensure a smooth introduction of IPv6. If you are a company owner or manager, you will be most interested in Chapters 7 and 9. If you need to plan your corporate network strategy, you will be most interested in Chapters 1, 4, 5, 7, and 9. If you manage the infrastructure in your company, you will especially be interested in Chapters 4 and 5, which cover ICMPv6, Layer 2 issues, and routing, and in Chapters 7 and 9, which address transition mechanisms, interoperability, and planning. If you are a system or network administrator, all chapters are relevant: this book provides a foundation for IPv6 implementation and integration with IPv4.

About This Book

This book covers IPv6 in detail and explains all the new features and functions. It will show you how to plan for, design, and integrate IPv6 in your current IPv4 infrastructure.

This book assumes that you have a good understanding of network issues in general and a familiarity with IPv4. It is beyond the scope of this book to discuss IPv4 concepts in detail. I refer to them when necessary, but if you want to learn more about IPv4, there are a lot of good resources on the market. You can find a list of books in Appendix B.

In explaining all the advanced features of IPv6, this book aims to inspire you to rethink your networking and service concepts for the future and create the foundation for a real nex-generation network.

Organization

This book is organized so that a reader familiar with IPv4 can easily learn about the new features in IPv6 by reading Chapters 2 through 7. These chapters cover what you need to know about addressing, the new IPv6 header, ICMPv6, Layer 2, routing protocols, DNS and DHCPv6, security, Quality of Service (QoS), and the transition mechanisms that make IPv6 work with IPv4 in different stages of transition. Mobile IPv6 is discussed in Chapter 8. Chapter 9 covers the planning process and considerations to make, and puts all the technical pieces together. Here is a chapter-by-chapter breakdown of the book:

- Chapter 1, *Why IPv6?*, briefly explains the history of IPv6 and gives an overview of the new functionality. It draws a bigger picture of Internet and service evolution, showing that the large address space and the advanced functionality of IPv6 are much needed for different reasons. It then discusses the most common misconceptions that prevent people from exploring and integrating the protocol. Finally, it explains when it would be the right moment for you to start your IPv6 project and drive the integration.
- Chapter 2, *IPv6 Addressing*, explains everything you need to know about the new address architecture, the address format, address notation, address types, international registry services, and prefix allocation.
- Chapter 3, *The Structure of the IPv6 Protocol*, describes the new IPv6 header format with a discussion of each field and trace file examples. It also describes what Extension headers are, what types of Extension headers have been defined, and how they are used.
- Chapter 4, *ICMPv6*, describes the new ICMPv6 message format, the ICMPv6 Error messages and Informational messages, and the ICMPv6 header in the trace file. This chapter also discusses the extended functionality based on ICMPv6, such as Neighbor Discovery, Autoconfiguration, Path MTU Discovery, and Multicast Listener Discovery (MLD). You will learn how ICMPv6 makes an administrator's life easier.
- Chapter 5, *Networking*, covers several network-related aspects and services, such as Layer 2 support for IPv6, Upper Layer Protocols and Checksums, an overview of all multicast-related topics, an overview of routing protocols, Quality of Service (QoS), DHCPv6, and DNS.
- Chapter 6, *Security with IPv6*, begins with a short discussion of basic security concepts and requirements. It then covers the IPsec framework, security elements available in IPv6 for authentication and encryption, and how they are used. Our future networks will require new security architectures. This chapter provides an overview of considerations to make when defining the IPv6 security concept.
- Chapter 7, *Transition Technologies*, discusses the different transition mechanisms that have been defined, such as dual-stack operation and different tunneling, and translation techniques. It also shows how they can be used and combined to ensure peaceful coexistence and smooth transition. This is your toolkit to plan a cost- and labor-efficient transition.
- Chapter 8, *Mobile IPv6*, covers Mobile IPv6. This chapter explains why this technology could become the foundation for a new generation of mobile services. It also shows how the Extension header support of IPv6 can provide functionality that IPv4 can't.

- Chapter 9, *Planning for IPv6*, puts it all together in a big picture. It discusses the planning process, success criteria, integration scenarios, best practices, and a summary of do's and don'ts based on my long-time consulting experience.
- Appendix A, *RFCs*, includes a short introduction to the RFC process and authorities, and provides a list of relevant RFCs for IPv6.
- Appendix B, *Recommended Reading*, provides a list of books that I recommend.

Some important topics and information appear in multiple places in the book. This is not because I want to bore you, but because I assume that most readers will not read the book from the first page to the last page, but rather will pick and choose chapters and sections depending on interest. So if the information is important with regard to different sections and contexts, I may mention it again.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/ipv6-3e*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

There are many people all over the world who have contributed to this book. Without their help and input, it would not be what it is.

For the first edition: many thanks go out to Anja Spittler (Maggy). She spent hours, days, and weeks in our lab in the early days of IPv6, setting up SuSE Linux, getting BIND and other services to work, and writing parts of Chapters 9 and 12 in the first edition. I also want to thank the technical editors, who have made this book much better with their invaluable comments, corrections, and clarifications. They were great resources when I was struggling with a topic and needed some answers. The technical reviewers of the first edition were Patrick Grossetete, who worked as a product manager for the Internet Technology Division (ITD) at Cisco, and Neil Cashell, who is a great TCP/IP guy at Novell, today SuSE. Thanks also to Brian McGehee, who has been working with IPv6 for many years and has written numerous courses for IPv6. He did the final technical edits of the first edition and added a lot of useful information. I'd like to thank Cisco Switzerland, especially René Räber, both for providing an updated router and access to their technical resources, as well as for his support of my work for IPv6. Thanks to the guys at SuSE for providing software and supporting us in getting our SuSE host ready for IPv6; Microsoft for providing software and information about their implementations; Network General for providing Sniffer Pro Software for the trace files; Bob Fink for running the 6Bone website; Cricket Liu for answering my DNS questions; and Peter Bieringer for running a great Internet resource site and for answering my questions with lightning speed.

There were many additional supporters, writers, and reviewers for the second edition. They include: Jim Bound from HP, CTO of the IPv6 Forum and Chair of the NAv6TF; Latif Ladid, President of the IPv6 Forum; Tim Chown, Department of Electronics and Computer Science at the University of Southampton; and Vijayabhaskar from McAfee. Yurie Rich, John Spence, and Mike Owen from Native6 Inc. in Seattle provided substantial input into Chapters 1, 5, 6, and 10 of the second edition. Gene Cronk from the

Robin Shepherd Group gave substantial input into Chapters 5 and 10, and John Jason Brzozowski, North American IPv6 Task Force and Chair of the Mid-Atlantic IPv6 Task Force, contributed great input into Chapters 1 and 9. Thanks to David B. Green from SRI International for the permission to quote his Enterprise Security Model presentation in Chapter 5 and for reviewing different parts of the book. Thanks to Merike Kaeo, Chief Network Security Architect at Double Shot Security, for all her inputs and comments to Chapter 5. And thanks to Chris Engdahl from Microsoft for his review of Chapter 10. Thanks to Jimmy Ott from Sunny Connection for researching and writing all updates for Chapter 12. David Malone, author of the companion book *IPv6 Network Administration*, reviewed the whole book—thank you, David, for your great and clarifying comments. A great thank you goes out to all the people who were ready to share their experience with me and have provided case studies. They are Paolo Vieira from the University of Porto, Pierre David from the University of Strasbourg, Cody Christman from NTT Communications, and Flavio Curti and Ueli Heuer from Cyberlink AG in Zurich. Wolfgang Fritsche from IABG Germany and Karim El-Malki from Ericsson AB in Stockholm reviewed and provided input on Chapter 8 about Mobility. Thanks to the people at Checkpoint for providing information and connections, especially Patrik Honegger and Yoni Appel; and thanks also to Jean-Marc Uzé at Juniper for his information and connections. I also want to thank all the people and developers in the international working groups. Without their visionary power, enthusiasm, and tireless work, we would not have IPv6 ready.

I would like to honor *Jim Bound*, mentioned in the acknowledgments for the second edition. He was the key developer and driver of IPv6 for many years. He was the CTO of the International IPv6 Forum and a member of the IETF (Internet Engineering Task Force) IP Next Generation directorate. Without his drive, knowledge, and passion, IPv6 would not be where it is today. Unfortunately, Jim left this world way too early in 2009 at the age of 58. In honor of Jim, the *International IPv6 Forum* has created the *Jim Bound Award*, which is given to countries for World Leadership in IPv6 Deployment. I was honored to receive the first Jim Bound Award for the Swiss IPv6 Council, for Switzerland being the first country in the world reaching a double-digit IPv6 user penetration rate in April of 2013.

For this third edition, I was happy to have many great and knowledgeable helpers.

First of all, I would like to thank my three main reviewers who reviewed all chapters. They are Ed Horley, David Malone, and Niall Murphy. Thank you guys for your great inputs, your thoughts, and inspirations, and for taking the time to do this and answer my questions. Ed Horley is also the author of *Practical IPv6 for Microsoft Administrators*, a must-read for all who deal with Microsoft operating systems. I would like to thank Mark Townsley, Cameron Byrne, and Jan Zorz for reviewing and providing important input to Chapters 7 and 9, Chip Popoviciu for writing the MPLS section, Gerd Pflüger for writing the LISP section, and Eric Vyncke for his inputs and review of Chapter 6. I would also like to thank Jasper Bongertz, my network analysis guru and IPv6 trainer,

for helping with Wireshark cosmetics, and Uwe Lenz, my second IPv6 instructor. He created an awesome lab for my hands-on class and used it to create all sorts of trace files for this book. Thanks to Andrew Yourtchenko and Gert Döring for responding to my many questions and to Jeff Carrell for many interesting discussions about the inner workings of SLAAC and the subtleties of what we see in trace files. I would also like to thank Bea Leonhardt for managing my office when I was writing and for help with updating the RFC lists. And Robin Huber for being an enthusiastic IT guy helping me with my infrastructure, solving my PC issues, taking care of the logistics at our IPv6 conferences, and for updating me on the latest gaming devices. And last but not least, Latif Ladid for all his continuing work for the IPv6 community, for cheering me up when working on weekends, and for getting Vint Cerf on board for the foreword.

And to all the great people at O'Reilly: for the first edition, a special thank you goes to Jim Sumser, Mike Loukides, and Tatiana Apandi. Jim Sumser guided me through the whole writing process of the first edition with a lot of enthusiasm, patience, and experience. Thank you, Jim, for being there, and thank you for never hassling me when I was already struggling. You made a difference! Mike and Tatiana, with whom I worked on the second edition, have also been very supportive throughout the whole process. I also want to thank all the other folks at O'Reilly who contributed to this book, especially Tim O'Reilly for making it possible in the first place. For this third edition, I was mostly working with Meghan Blanchette. Meghan, I thank you for all your great work, your support, your humor, and your patience with my crazy schedule. You were always there when I reached out and helped me stay on track.

Another very special thank you goes to Hanspeter Bütler, who was my teacher back in school, for teaching me the beauty of the ancient Greek language. His insightful and sensitive way of guiding me into understanding and feeling the richness of old languages laid the foundation for my understanding of language in general, of different cultures and how the differences in viewing the world are expressed in language. I can probably make him partially responsible for my becoming an author. Language is made to communicate, and the more precisely we use our language, the better we can understand and be understood. Without communication, there can be no understanding. On a different level, TCP/IP is the protocol that enables communication in the network and therefore creates the foundation for Internet communication. And the Internet creates the physical foundation for global communication. It offers a great opportunity to communicate, share, and understand globally across all cultures. That is how we should be using it.

CHAPTER 1
Why IPv6?

The IP version currently used in networks and the Internet is IP version 4 (IPv4). IPv4 was developed in the early '70s to facilitate communication and information sharing between government researchers and academics in the United States. At the time, the system was closed with a limited number of access points, and consequently the developers didn't envision requirements such as security or quality of service. To its credit, IPv4 has survived for over 30 years and has been an integral part of the Internet revolution. But even the most cleverly designed systems age and eventually become obsolete. This is certainly the case for IPv4. Today's networking requirements extend far beyond support for web pages and email. Explosive growth in network device diversity and mobile communications, along with global adoption of networking technologies, new services, and social networks, are overwhelming IPv4 and have driven the development of a next-generation Internet Protocol.

IPv6 has been developed based on the rich experience we have from developing and using IPv4. Proven and established mechanisms have been retained, known limitations have been discarded, and scalability and flexibility have been extended. IPv6 is a protocol designed to handle the growth rate of the Internet and to cope with the demanding requirements on services, mobility, and end-to-end security.

When the Internet was switched from using Network Control Protocol (NCP) to Internet Protocol (IP) in one day in 1983, IP was not the mature protocol that we know today. Many of the well-known and commonly used extensions were developed in subsequent years to meet the growing requirements of the Internet. In comparison, hardware vendors and operating system providers have been supporting IPv6 since 1995 when it became a Draft Standard. In the decade since then, those implementations have matured, and IPv6 support has spread beyond the basic network infrastructure and will continue to be extended.

It is very important for organizations to pay attention to the introduction of IPv6 as early as possible because its use is inevitable in the long term. If IPv6 is included in

strategic planning; if organizations think about possible integration scenarios ahead of time; and if its introduction is considered when investing in IT capital expenditures, organizations can save considerable cost and can enable IPv6 more efficiently when it is needed.

An interesting and humorous overview of the history of the Internet can be found in RFC 2235, "Hobbes' Internet Timeline." The account starts in 1957 with the launch of *Sputnik* in Russia and the formation of the Advanced Research Projects Agency (ARPA) by the Department of Defense (DoD) in the United States. The RFC contains a list of yearly growth rate of hosts, networks, and domain registrations in the Internet.

Some excerpts from the RFC:

- 1969: Steve Crocker makes the first Request for Comment (RFC 1): "Host Software."
- 1970: ARPANET hosts start using Network Control Protocol (NCP).
- 1971: 23 hosts connect with ARPANET (UCLA, SRI, UCSB, University of Utah, BBN, MIT, RAND, SDC, Harvard, Lincoln Lab, Stanford, UIU©, CWRU, CMU, NASA/Ames).
- 1972: InterNetworking Working Group (INWG) is created with Vinton Cerf as Chairman to address the need for establishing agreed-upon protocols. Telnet specification (RFC 318) is published.
- 1973: First international connections to the ARPANET are made at the University College of London (England) and Royal Radar Establishment (Norway). Bob Metcalfe's Harvard PhD thesis outlines the idea for Ethernet. File transfer specification (RFC 454) is published.
- 1976: Queen Elizabeth II sends an email.
- 1981: Minitel (Teletel) is deployed across France by France Telecom.
- 1983: The cutover from NCP to TCP/IP happens on January 1.
- 1984: The number of hosts breaks 1,000.
- 1987: An email link is established between Germany and China using CSNET protocols, with the first message from China sent on September 20. The thousandth RFC is published. The number of hosts breaks 10,000.
- 1988: An Internet worm burrows through the Net, affecting 10 percent of the 60,000 hosts on the Internet.
- 1989: The number of hosts breaks 100,000. Clifford Stoll writes *Cuckoo's Egg*, which tells the real-life tale of a German cracker group that infiltrated numerous U.S. facilities.
- 1991: The World Wide Web (WWW) is developed by Tim Berners-Lee and released by CERN.

- 1992: The number of hosts breaks 1,000,000. The World Bank comes online.
- 1993: The White House comes online during President Bill Clinton's time in office. Worms of a new kind find their way around the Net—WWW Worms (W4) are joined by Spiders, Wanderers, Crawlers, and Snakes.
- 1994: Internet shopping is introduced; the first spam mail is sent; Pizza Hut comes online.
- 1995: The Vatican comes online. Registration of domain names is no longer free.
- 1996: 9,272 organizations find themselves unlisted after the InterNIC drops their name service as a result of their not having paid their domain name fees.
- 1997: The 2,000th RFC is published.

This is how far the RFC goes. But history goes on. According to *http://www.internetworldstats.com/emarketing.htm*, the worldwide online population reached 361 million users in 2000 (a penetration rate of 5.8%) and 587 million users in 2002. In 2003, the U.S. Department of Defense announced that they would be migrating the DoD network to IPv6 by 2008, and the Moonv6 project was started (now concluded). In 2005, Google registered a /32 IPv6 prefix, and Vint Cerf, known as "Father of the Internet," joined Google. By that time the number of Internet users had reached 1.08 billion. Today, at the time of writing in 2014, we are at approximately 2.4 billion Internet users, which corresponds to a penetration rate of 34%.

So while these numbers reflect all Internet users, independent of the IP protocol version, now we are starting to watch the growth of the IPv6 Internet. It is in its early days, but according to the growth numbers of the last two years, we expect growth to be exponential, and probably much faster than even the enthusiasts among us expect. The growth of the IPv6 Internet can be seen on the Google IPv6 Adoption statistics and the stats as of spring 2014 are shown in Figure 1-1.

The stats show that in early 2011 (when the IANA IPv4 pool ran out), the percentage of native IPv6 Internet users was at approximately 0.2%. The stats also show that the percentage of users that were not native IPv6 (e.g., 6to4 or Teredo, red line) dropped to almost zero and are since then insignificant. Within one year the number of IPv6 Internet users doubled to 0.4%—a small number but still growth. In January 2013, the IPv6 Internet had crossed the 1% mark, and we entered 2014 with almost 3% IPv6 Internet users, which corresponds to approximately 72 million users. At the time of delivering this chapter, in April 2014, we were at 3.5%. The number of IPv6 Internet users currently doubles approximately every nine months.

These are just a few selected events and milestones of the Internet's history. Keep watching as more history unfolds. We are all creating it together.

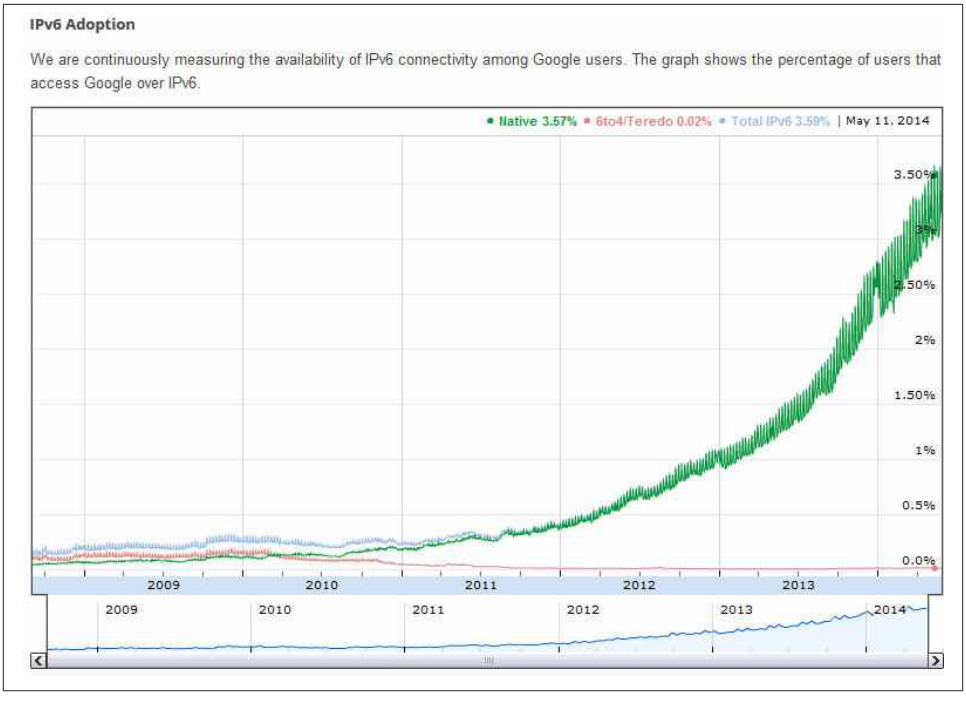

Figure 1-1. Google's global IPv6 adoption statistics as of spring 2014

The History of IPv6

The Internet Engineering Task Force (IETF) began the effort to develop a successor protocol to IPv4 in the early 1990s. Several parallel efforts to solve the foreseen address space limitation and to provide additional functionality began simultaneously. The IETF started the Internet Protocol Next Generation (or *IPng*) area in 1993 to investigate the different proposals and to make recommendations for further procedures.

The IPng area directors of the IETF recommended the creation of IPv6 at the Toronto IETF meeting in 1994. Their recommendation is specified in RFC 1752, "The Recommendation for the IP Next Generation Protocol." The Directors formed an Address Lifetime Expectation (ALE) working group to determine whether the expected lifetime for IPv4 would allow the development of a protocol with new functionality, or if the remaining time would allow only the development of an address space solution. In 1994, the ALE working group projected that the IPv4 address exhaustion would occur sometime between 2005 and 2011 based on the available statistics.

For those of you who are interested in the different proposals, here's some more information about the process (from RFC 1752). There were four main proposals: CNAT, IP Encaps, Nimrod, and Simple CLNP. Three more proposals followed: the P Internet

Protocol (PIP), the Simple Internet Protocol (SIP), and TP/IX. After the March 1992 San Diego IETF meeting, Simple CLNP evolved into TCP and UDP with Bigger Addresses (TUBA), and IP Encaps became IP Address Encapsulation (IPAE). IPAE merged with PIP and SIP and called itself Simple Internet Protocol Plus (SIPP). The TP/IX working group changed its name to Common Architecture for the Internet (CATNIP). The main proposals were now CATNIP, TUBA, and SIPP. For a short discussion of the proposals, refer to RFC 1752.

CATNIP is specified in RFC 1707; TUBA in RFCs 1347, 1526, and 1561; and SIPP in RFC 1710.

The Internet Engineering Steering Group approved the IPv6 recommendation and drafted a Proposed Standard on November 17, 1994. RFC 1883, "Internet Protocol, Version 6 (IPv6) Specification," was published in 1995. The core set of IPv6 protocols became an IETF Draft Standard on August 10, 1998. This included RFC 2460, which obsoleted RFC 1883.

Why isn't the new protocol called IPv5? The version number 5 could not be used, because it had been allocated to the experimental stream protocol.

One of the big challenges but also one of the main opportunities of IPv6 is the fact that we can redesign our networks for the future. This is what enterprises should focus on most when planning their IPv6 integration in order to make sure they don't just copy old concepts onto a new protocol. We have to rethink our architectures. This once-in-a-lifetime opportunity can be used to get rid of a lot of legacy. An interesting RFC that helps in the process of seeing the big picture is RFC 6250, "Evolution of the IP Model." It shows how much this model has changed in the many years of operating our networks. So it helps to free our minds for thinking in new ways. One funny little quote that demonstrates what I am talking about is included below.

In this RFC there is mention of the first IP model and addressing architecture, and it quotes RFC 791, which defined IPv4 and the IPv4 address:

> *Addresses are fixed length of four octets (32 bits). An address begins with a one-octet network number, followed by a three-octet local address. This three-octet field is called the "rest" field.*

This is how far we have come. Now project this into the future with the vast IPv6 address space in mind. Making meaningful use of the new address architecture and the enormous space will write the next chapter of the evolution of the IP model.

Here's a quote from Vint Cerf:

> The vast IPv6 address space opens up serious opportunities for the re-examination of the notion of address. The IETF has only allocated 1/8th of the IPv6 address space for current use. The remaining 7/8^{ths} of the address space is still to be allocated. In consequence we may be able to interpret new segments of the IP address space in ways that are different from topological end points. This is precisely the reason that a focus on the future of IPv6 is so important at this point in the evolution of the Internet.

What's New in IPv6?

IPv6 is an evolution of IPv4. The protocol is installed as a software upgrade in most devices and operating systems. If you buy up-to-date hardware and operating systems, IPv6 is usually supported and needs only activation or configuration. In many cases it is activated by default. Currently available transition mechanisms allow the step-by-step introduction of IPv6 without putting the current IPv4 infrastructure at risk.

Here is an overview of the main changes:

Extended address space
: The address format is extended from 32 bits to 128 bits. This provides multiple IP addresses for every grain of sand on the planet. In addition, it also allows for hierarchical structuring of the address space in favor of optimized global routing.

Autoconfiguration
: Perhaps the most intriguing new feature of IPv6 is its *Stateless Address Autoconfiguration (SLAAC)* mechanism. When a booting device in the IPv6 world comes up and asks for its network prefix, it can get one or more network prefixes from an IPv6 router on its link. Using this prefix information, it can autoconfigure for one or more valid global IP addresses by using either its MAC identifier or a private random number to build a unique IP address. In the IPv4 world, we have to assign a unique IP address to every device, either by manual configuration or by using DHCP. SLAAC should make the lives of network managers easier and save substantial cost in maintaining IP networks. Furthermore, if we imagine the number of devices we may have in our homes that will need an IP address in the future, this feature becomes indispensable. Imagine reconfiguring your DHCP server at home when you buy a new television! Stateless Address Autoconfiguration also allows for

easy connection of mobile devices, such as a smartphone, when moving to foreign networks.

Simplification of header format
: The IPv6 header is much simpler than the IPv4 header and has a fixed length of 40 bytes. This allows for faster processing. It basically accommodates two times 16 bytes for the Source and Destination address and only 8 bytes for general header information.

Improved support for options and extensions
: IPv4 integrates options in the base header, whereas IPv6 carries options in so-called *Extension headers*, which are inserted only if they are needed. Again, this allows for faster processing of packets. The base specification describes a set of six Extension headers, including headers for routing, Quality of Service, and security.

Why Do We Need IPv6?

For historic reasons, organizations and government agencies in the United States used the largest part of the allocatable IPv4 address space. The rest of the world had to share what was left over. Some organizations used to have more IPv4 address space than the whole of Asia (where more than 50% of the world's population live). This is one explanation of why the deployment of IPv6 in Asia is much more common than in Europe and the United States.

An interesting resource site for statistics can be found at the following link: *http://www.internetworldstats.com/stats.htm*.

The IPv4 address space has a theoretical limit of 4.3 billion addresses. However, early distribution methods allocated addresses inefficiently. Consequently, some organizations obtained address blocks much larger than they needed, and addresses that could be used elsewhere are now unavailable. If it were possible to reallocate the IPv4 address space, it could be used much more effectively, but this process is not possible, and a global reallocation and renumbering is simply not practical. In addition to that it would not buy much, as even 4.3 billion addresses would not suffice for long at the current growth rate. We have to take into account that in the future we will need IP addresses for billions of devices. Vendors in all industries are developing monitoring, control, and management systems based on IP.

As the previous section shows, the IPv6 working group has done more than just extend the address space. For many complex networks of today and tomorrow, and for the number of IP devices of all types, the Autoconfiguration capability of IPv6 will be a

necessity. The management of such services can't be accomplished with traditional addressing methods, and Stateless Address Autoconfiguration will also help to reduce administrative costs for organizations.

The extended address space and the restoration of the original end-to-end model of the Internet allows for the elimination of Network Address Translation (NAT), in which a single or a few public IPv4 address(es) are used to connect a high number of users with private addresses to the Internet by mapping the internal addresses to the public address(es). NATs were introduced as a short-term fix for solving the address space limitations with IPv4, since IPv6 was not ready yet (refer to RFC 1631; the original NAT specification was obsoleted by RFC 3022 in 2001). NATs have become pretty common in IPv4 networks, but they create serious disadvantages in management and operation: in order to do the address mapping, NATs modify end node addresses in the IP header. Very often, Application Level Gateways (ALG) are used in conjunction with NAT to provide application-level transparency. There is a long list of protocols and applications that create problems when used in a NAT environment. IPsec and peer-to-peer applications are two well-known examples. Another known issue with NAT is the overlapping of private address space when merging networks, which requires either the renumbering of one of the networks or the creation of a complex address-mapping scheme. The amplification of limited address space, the primary benefit of NAT, is not needed with IPv6 and therefore is not supported by design.

By introducing a more flexible header structure (Extension headers), the protocol has been designed to be open and extensible. In the future, new extensions can easily be defined and integrated in the protocol set. Based on the fact that IPv4 has been in use for almost 30 years, the development of IPv6 was based on the experience with IPv4 and focused on creating an extensible foundation; you can expect it to last a long time.

Broadband penetration rates in many countries continue to accelerate and, in some cases, have reached 65% or more. This level of always-on connectivity with substantial bandwidth capacity means that there is greater opportunity for devices to be connected. And many consumer electronic manufacturers have taken advantage of this. Online gaming is no longer the sole purview of games on PCs. Gaming stations, such as Sony's PlayStation 4, Xbox One, or Nintendo Wii U, have added capabilities to take them online. Many telecommunication carriers are providing television-type services (movies, audio content, etc.) over their IP networks. Even appliances, such as refrigerators, stoves, water heaters, and bathtubs, are getting connected. While it may seem rather silly to network-enable a bathtub, many of these devices are being connected to facilitate things such as power management, remote control, and troubleshooting, and for telemetry/monitoring purposes. We are entering the age of smart buildings and smart cities. The end result of this network-enablement process is a greater number of devices that need addressing, many of which will not have standard user interfaces. In these cases, the IPv6 address space, coupled with features such as Neighbor Discovery, Stateless Autoconfiguration, and Mobile IPv6, will help to usher in a new era of computerization in the home, but

hopefully without the enormous deployment headache that it would cause if it were attempted with the current protocol.

The growth of the wireless industry (both cellular and wireless networks) has been nothing short of phenomenal. In more and more countries the number of cell phones actually exceeds the number of people. In this world of continuous reachability and reliance on the ability to access information at any time, the mobility requirements for end users have become exceptionally important. From the carriers' perspective, especially those supporting multiple media access types (e.g., 3G, WiMax, LTE), leveraging IP as the method of transporting and routing packets makes sense. Smartphones access the Internet, play games with other users, make phone calls, and even stream video content. Instead of supporting all of these functions using different transport protocols and creating intermediary applications to facilitate communications, it is far more efficient to leverage the existing network infrastructure of the Internet and a company's network. We will see later that from a technical perspective, Mobile IPv6 is very elegant in its design, supporting mobile users in a highly efficient manner and providing the overlay mechanisms for users to maintain their connections when moving between networks, even if those networks do not use the same type of media access.

There still remain some questions about the value of IPv6 to the enterprise, and it is worth conceding that each organization needs to evaluate the benefits and best timing of IPv6 for their own internal use. In many instances, organizations can find clever ways to use IPv6 to solve "pain" issues without migrating their entire network. Adoption can occur in an incremental fashion with a plan that minimizes integration pain but also ensures that everything is ready when the time comes to "flip the switch." As many case studies show, well-planned introduction costs substantially less than you would expect; the main cost-saving aspect is the fact that the advance planning lets you use all your refresh cycles, which minimizes cost. The step-by-step introduction allows you to learn as you go, thereby saving a lot of money and headaches, and you can do it without putting the current IPv4 infrastructure at risk.

But with all these thoughts and considerations, let's not forget the most essential advantage of IPv6. With its new structure and extensions, IPv6 provides the foundation for a new generation of services. There will be devices and services on the market in the near future that cannot be developed with IPv4. This opens up new markets and business opportunities for vendors and service providers alike. The first-mover opportunities are substantial, as are the opportunities to extend current product life cycles by refreshing their technology with IPv6. On the other hand, it means that organizations and users will require such services in the mid-term. It is therefore advisable to integrate the new protocol carefully and in a nondisruptive manner, by taking one step at a time to prepare the infrastructure for these new services. This protects you from having to introduce a business-critical application based on IPv6 at unreasonably high cost with no time for thorough planning.

Common Misconceptions

When considering all these advantages, maybe the question should be: "Why not IPv6?" When talking to customers, we often find that they share a similar set of misconceptions preventing them from considering IPv6. Here are the most common ones:

"The introduction of IPv6 puts our current IP infrastructure—our networks and services—at risk."

This concern is unsubstantiated. A major focus in IPv6's development was to create integration mechanisms that allow both protocols to coexist peacefully. You can use IPv6 both in tandem with and independently of IPv4. It is possible to introduce IPv6 and use it for access to new services while retaining IPv4 to access legacy services. This not only ensures undisrupted access to IPv4 services, but it also allows a step-by-step introduction of IPv6. I discuss these mechanisms in Chapter 7. Your biggest risk is to not take advantage of all the opportunities IPv6 offers. You can only use these opportunities if you plan while there is time.

"The IPv6 protocol is immature and hasn't proven that it stands the test of time or whether it is capable of handling the requirements."

This was a concern of many people back in 2006 when we published the second edition of this book. Now in 2014 this is not true anymore. Many ISPs and organizations are deploying IPv6, vendors are getting up to speed, and the working groups have developed and optimized mechanisms that help with the integration. There is no technical reason not to do IPv6.

"The costs of introducing IPv6 are too high."

There will certainly be costs associated with adopting IPv6. In many cases, newer networks will find that the level of IPv6 support in their current infrastructure is actually high. Regardless, the transition will necessitate some hardware and software costs. Organizations will need to create new designs, review current concepts, train their IT staff, and may need to seek outside expertise in order to take full advantage of IPv6.

However, the cost savings associated with IPv6 are becoming easier to define. Networks based on IPv4 are becoming increasingly more complex. New IT services such as VoIP, instant messaging, video teleconferencing, IPTV, and unified communications are adding layers of middleware and complexity. Merging organizations or those conducting B2B transactions are implementing NAT overlap solutions that have high management costs and are difficult to troubleshoot. And a growing market of mobile devices and network appliances requires robust access models that are expensive and difficult to implement in an IPv4 world. In all of these cases, IPv6 presents a cleaner and more cost-effective model in the long run than IPv4 can provide. And the fact is that an investment in IPv4 is an investment

in an end-of-life technology, while an investment in IPv6 is an investment in the future technology.

"With Stateless Address Autoconfiguration, we will not be able to control or monitor network access."

While this statement may generally be true for networks that widely utilize Stateless Address Autoconfiguration, administrators will have a choice about their level of control. DHCPv6 as defined in RFC 3315 has been extended to support two general modes of operation, Stateful and Stateless. Stateful mode is what those who currently utilize DHCP (for IPv4) are familiar with, in which a node (DHCP client) requests an IP address and configuration options dynamically from a DHCP server. DHCPv6 also offers a Stateless mode in which DHCPv6 clients simply request configuration options from a DHCPv6 server and use other means, such as Stateless Address Autoconfiguration, to obtain an IPv6 address.

"Our Internet Service Provider (ISP) does not offer IPv6 services, so we can't use it."

You do not have to wait for your ISP to use IPv6 in your corporate or private network. If you want to connect to the global IPv6 Internet, you can use one of the transition mechanisms and tunnel your IPv6 packets over the IPv4 infrastructure of your ISP. This may be doable for smaller organizations. On the other hand, at the time of writing in 2014, you could expect a large ISP targeting enterprise customers to support IPv6. And this should be your standard requirement in any renewal of contract and SLAs (Service Level Agreements). If your ISP does not provide IPv6 services, consider finding a new provider.

"It would be too expensive and complex to upgrade our backbone."

The transition mechanisms make it possible to use IPv6 where appropriate without dictating an order of upgrade. Usually for the backbone it is advisable to wait for the regular life cycle, when hardware needs to be exchanged anyway. Make sure to choose hardware that supports performance IPv6 routing. In the meantime, you can tunnel your IPv6 packets over the IPv4 backbone. Networks that use MPLS have an easy way to tunnel IPv6 packets over their IPv4 MPLS backbone. Read more about it in Chapter 7. More and more organizations are considering migrating their backbone and data centers to IPv6 only with the next refresh or redesign cycle, because it substantially reduces operational cost. In this scenario we will start to tunnel IPv4 packets over IPv6 backbones. IPv4 as a service is the new keyword.

"It would be too complex and expensive to port all of our applications to IPv6."

The effort necessary to port applications to run over IPv6 is often much lower than expected. If an application is well written, it may simply run over IPv6 without modification. Instead of assuming that it won't work, test it to find out. For applications that need modifications that are not yet available, or for applications in which porting does not make sense, there are mechanisms available that support IPv4 applications in IPv6 networks and IPv6 applications in IPv4 networks.

Alternatively, you can run a dual-stack network, in which you use IPv4 to access IPv4 applications and IPv6 to access IPv6 applications. In any case it is recommendable for enterprise customers to start the planning process early and provide good labs for the application teams to test their applications before there is time pressure.

"We have enough IPv4 addresses; we don't need IPv6."
True—if you have enough IPv4 addresses, there may be no immediate need to integrate IPv6 today. But ignoring IPv6 for this reason is a perspective that assumes that your network stands completely isolated from the rest of the world, including your vendors, partners, and customers. IPv6 adoption is further along in Asia and Europe than in the United States, so even though you may have adequate address space for your operations in Denver, interconnecting with a partner organization in Tokyo may eventually become complicated if you do not support IPv6. Plus, the assumption that IPv6 is about address space only doesn't account for the advanced features that IPv6 brings to the table.

When Is It Time for IPv6?

The answer in 2014 is *now*! If the rest of the world moves to IPv6 while you insist on continuing to use IPv4, you will exclude yourself from global communication and reachability. The risks if you wait too long include losing potential customers and access to new markets and the inability to use new IPv6-based business applications.

There is a golden rule in IT: "Never touch a running system." As long as your IPv4 infrastructure runs well and fulfills your needs, there is no reason to change anything. But from now on, whenever you invest in your infrastructure, you should consider IPv6. An investment in the new technology gives it a much longer lifetime and keeps your network state-of-the-art.

These are the main indicators that it may be time for you to consider switching to or integrating IPv6:

- You need to extend or fix your IPv4 network or NAT implementation.
- You are running out of address space.
- You want to prepare your network for applications that are based on advanced features of IPv6.
- You need end-to-end security for a large number of users and you do not have the address space, or you struggle with a NAT implementation.
- You need to replace your hardware or applications that are at the end of their life cycles. Make sure you buy products that support IPv6 adequately, even if you don't enable it right away.

- You want to introduce IPv6 while there is no time pressure.

The following provisions can be taken in order to prepare for IPv6 adequately:

- Build internal knowledge, educate IT staff, and create a test network.
- Include IPv6 in your IT strategy.
- Design future-proof network, security, and service concepts while you have time.
- Create integration scenarios based on your network and requirements.
- Put IPv6 support on all of your hardware and software purchasing guidelines. Be specific about which features (RFCs) must be supported. Don't forget to add IPv6 requirements to outsourcing and service contracts, as well as SLAs.
- Compel your vendors to add IPv6 support to their products.

If you do this, you can determine the right moment for the introduction of IPv6 in your network. You can also assess whether a further investment in your IPv4 infrastructure makes sense or whether introducing IPv6 would be a better way to go.

There will be no "flag day" for IPv6 like there was for the 1983 move from NCP to IPv4. Probably there will be no killer application either, so don't wait for one. Or as some people like to say, the killer application for IPv6 is the Internet. IPv6 will slowly and gradually grow into our networks and the Internet. Taking a step-by-step approach to IPv6 may be the most cost-efficient way to integrate it, depending on your requirements. This method does not put your current infrastructure at risk or force you to exchange hardware or software before you are ready, and it allows you to become familiar with the protocol, to experiment, to learn, and to integrate what you've learned into your strategy.

You may want to enable IPv6 in your public services first. Due to the lack of IPv4 addresses, ISPs that want to grow their customer base (and who does not want to do that?) make use of NAT-type mechanisms to extend their IPv4 address space. This includes CGN (Carrier Grade NAT), which means multiple customers share one single public IPv4 address and sit behind multiple layers of NAT.

These users may have a bad user experience accessing your IPv4 website, and for e-commerce or other more complex services it may even fail. The users will not know that it is the provider's CGN causing the issue and will blame your website for their problems. If you provide your website dual-stack, these users can access it over IPv6 and bypass the IPv4 NATs.

Find more information on CGN in the "NAT to extend IPv4 address space" section in Chapter 7.

IPv6 Status and Vendor Support

As previously mentioned, IPv6 is implemented in most up-to-date versions of routing and operating systems. For standard applications, assume that IPv6 support has already been added or will be added with their next major release at the latest. For creating an IPv6 integration plan for your corporate network, you will need to assess the status and degree of IPv6 support with each vendor individually. Many vendors have an information site that can often be found at *http://www.<vendor>.com/ipv6*.

It can be said that IPv6 support up to the network layer is mature, tested, and optimized. This includes routing, transition mechanisms, DNS, and DHCPv6.

Development is most active in the security, transition mechanism, IPv4/IPv6 MIB integration, and Mobile IPv6 areas. More work needs to be done in the areas of network management and firewalls. Vendors such as Cisco, Checkpoint, Juniper, and many others are working on these areas. The application area is continuously developing, and new applications will appear on the market that will make use of the advanced features of IPv6. Thanks to the transition mechanisms, you can still use IPv4 applications in IPv6 networks.

Find more information on the planning process in Chapter 9.

Now you know why you should care about IPv6. The following chapters in this book aim to make learning about IPv6 a joy. So please read on.

References

Here's a list of the most important RFCs mentioned in this chapter. Sometimes I include additional subject-related RFCs for your further personal study.

RFCs

- RFC 1, "Host Software," 1969
- RFC 791, "Internet Protocol," 1981
- RFC 1347, "TCP and UDP with Bigger Addresses (TUBA)," 1992
- RFC 1526, "Assignment of System Identifiers for TUBA/CLNP Hosts," 1993
- RFC 1561, "Use of ISO CLNP in TUBA Environments,"1993
- RFC 1631, "The IP Network Address Translator (NAT)," 1994
- RFC 1707, "CATNIP: Common Architecture for the Internet," 1994
- RFC 1710, "Simple Internet Protocol Plus White Paper," 1994
- RFC 1752, "The Recommendation for the IP Next Generation Protocol," 1995
- RFC 1883, "Internet Protocol, Version 6 (IPv6) Specification," 1995
- RFC 2235, "Hobbes' Internet Timeline," 1997
- RFC 2324, "Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)," April 1, 1998
- RFC 2460, "Internet Protocol, Version 6 (IPv6) Specification," 1998
- RFC 2555, "30 Years of RFCs," 1999
- RFC 2663, "IP Network Address Translator (NAT) Terminology and Considerations," 1999
- RFC 3022, "Traditional IP Network Address Translator (Traditional NAT)," 2001
- RFC 3027, "Protocol Complications with the IP Network Address Translator," 2001
- RFC 4677, "The Tao of IETF: A Novice's Guide to the Internet Engineering Task Force," 2006
- RFC 5902, "IAB Thoughts on IPv6 Network Address Translation," 2010
- RFC 6250, "Evolution of the IP Model," 2011
- RFC 6269, "Issues with IP address sharing," 2011
- RFC 6921, "Design Considerations for Faster-Than-Light (FTL) Communication," April 1, 2013
- RFC 7168, "The Hyper Text Coffee Pot Control Protocol for Tea Efflux Appliances (HTCPCP-TEA)," April 1, 2014

CHAPTER 2

IPv6 Addressing

An IPv4 address has 32 bits and looks familiar. An IPv6 address has 128 bits and looks wild at first glance. Extending the address space was one of the driving reasons to develop IPv6, along with optimization of routing tables, especially on the Internet. This chapter will help you become familiar with the extended address space and will also explain how IPv6 addressing works and why it has been designed to be the way it is. There is a lot more to understand than just the 128-bit address. The address architecture has been extended and the large address space offers opportunity for new address designs. So make sure that you dive into this before you work on an IPv6 address plan. The IPv6 addressing architecture is defined in RFC 4291.

The IPv6 Address Space

The 32 bits of the IPv4 address space provide a theoretical maximum of 2^{32} addresses, equal to approximately 4.29 billion addresses. The current world population is over 7 billion people. So even if it were possible to use 100 percent of the IPv4 address space, we would not be able to provide an IP address for everyone on the planet. As a matter of fact, only a small fraction of this address space can be used. In the early days of IP, nobody foresaw the existence of the Internet as we know it today. Therefore, large address blocks were allocated without considerations for global routing and address conservation issues. These address ranges cannot be easily reclaimed, so consequently there are many unused addresses that are not available for allocation.

Are you aware that today (in 2014) only about 2.4 billion people have Internet access? They represent approximately 34 percent of the world's population.

The heated discussions about the end of the IPv4 address pool came to an end when the IANA (Internet Assigned Numbers Authority) declared on February 3, 2011, that the free pool was empty. This happened after IPv4 address consumption had more than doubled in 2010. On average the world had consumed approximately 10 /8 blocks per year for the last 10 years. In January 2010 there were 24 /8 blocks still available. So it should have lasted more than two years. But only one year later, in January 2011, the pool was empty. This is an indication of how fast the Internet is growing. And the Internet will continue to grow at that pace, if not faster. Only now, because the IPv4 pool is empty, the Internet's growth will to a large part happen over IPv6.

The evolution of the Internet and our services shows that in the future, not only will we need addresses for users and computers, but we'll also need more and more addresses for all sorts of devices that need permanent Internet connections, such as smartphones, tablets, webcams, refrigerators, cars, infusion pumps, water and electric meters, and many more items. Car manufacturers, as one example, which are designing the networked car of the future, need many IP addresses per car. How many cars do we have? According to *http://howmanyarethere.net* there were about 1 billion cars in the world in 2011. So, multiply this with, let's say, 50 IP addresses…there we go! These addresses will be used for monitoring and maintenance as well as for access to services such as weather and traffic information. There was a prototype Renault car with an integrated Cisco router and a Mobile IPv6 implementation built in the early years of the last decade. Most of the big car manufacturers have similar plans and prototypes.

The IPv6 address space uses a 128-bit address, meaning that we have a maximum of 2^{128} addresses available. Do you want to know what this number looks like? It equals 340,282,366,920,938,463,463,374,607,431,768,211,456, or in other words 6.65×10^{23} addresses per square meter on earth. It is pronounced as 340 undecillion addresses. For all of you who, like me, cannot imagine how much this is, it can be compared to providing multiple IP addresses for every grain of sand on the planet. The IPv4 address space with the originally defined address classes (A, B, C, D, E) allows for 2,113,389 network IDs. With the introduction of Classless Interdomain Routing (CIDR), this number was slightly extended. Let's compare this with IPv6. The address space with the current prefix for global unicast addresses (binary 001) allows for 2^{45} network IDs with a /48 prefix, or 35,184,372,088,832 networks. Each of these networks can further be divided into 65,536 subnets using the remaining 16 bits of the prefix.

And in a little while, when we are deeper into this chapter and discuss the address format, I'll show you another comparison that will help you to understand how big this address space really is.

Address Types

IPv4 knows unicast, broadcast, and multicast addresses. With IPv6, the broadcast address is not used anymore; multicast addresses are used instead. This is good news

because broadcasts are a problem in most networks. The anycast address, a type of address introduced with RFC 1546, has already been used in the IPv4 world but will probably be used on a wider basis with IPv6.

Unicast, Multicast, and Anycast Addresses

An IPv6 address can be classified into one of three categories:

Unicast
: A unicast address uniquely identifies an interface of an IPv6 node. A packet sent to a unicast address is delivered to the interface identified by that address.

Multicast
: A multicast address identifies a group of IPv6 interfaces. A packet sent to a multicast address is processed by all members of the multicast group.

Anycast
: An anycast address is assigned to multiple interfaces (usually on multiple nodes). A packet sent to an anycast address is delivered to only one of these interfaces, usually the nearest one.

Some General Rules

IPv6 addresses are assigned to interfaces as in IPv4, not to nodes as in OSI, so each interface of a node needs at least one unicast address. A single interface can also be assigned multiple IPv6 addresses of any type (unicast, multicast, and anycast). A node can therefore be identified by the address of any of its interfaces. It is also possible to assign one unicast address to multiple interfaces for load-sharing reasons, but if you do this, you need to make sure that the hardware and drivers support it.

With IPv6, all zeros and ones are legal values for any field in an address.

IPv6 supports addresses of different *scopes*. There are global and nonglobal (e.g., link-local) scopes. Operationally, the use of nonglobal addresses has been introduced with IPv4 by using IP addresses from the private range or administratively scoped multicast addresses. The design of IPv6 includes the address scope in the base architecture. Every IPv6 address other than the unspecified address has a specific scope, which is a topological span within which the address may be used as a unique identifier for an interface or set of interfaces. The scope of an address is encoded as part of the address. You can find a description of scopes in the section "Multicast Address" on page 37, and refer to RFC 4007, "IPv6 Scoped Address Architecture" for an explanation of scopes.

Address Notation

An IPv6 address has 128 bits, or 16 bytes. The address is divided into eight 16-bit hexadecimal blocks separated by colons. For example:

```
2001:0db8:0000:0000:0202:b3ff:fe1e:8329
```

To make life easier, some abbreviations are possible. For instance, leading zeros in a 16-bit block can be skipped. The example address now looks like this:

```
2001:db8:0:0:202:b3ff:fe1e:8329
```

A double colon can replace consecutive zeros or leading or trailing zeros within the address. If we apply this rule, our address looks as follows:

```
2001:db8::202:b3ff:fe1e:8329
```

Note that the double colon can appear only once in an address. The reason for this rule is that the computer always uses a full 128-bit binary representation of the address, even if the displayed address is simplified. When the computer finds a double colon, it expands it with as many zeros as are needed to get 128 bits. If an address had two double colons, the computer would not know how many zeros to add for each colon. So the IPv6 address `2001:db8:0000:0000:0056:abcd:0000:1234` can be represented in the following ways (note the two possible positions for the double colon):

```
2001:db8:0000:0000:0056:abcd:0000:1234
2001:db8:0:0:56:abcd:0:1234
2001:db8::56:abcd:0:1234
2001:db8:0:0:56:abcd::1234
```

There are so many different ways to write and abbreviate IPv6 addresses and this can cause operational troubles. If you want to do lookups in a database or a spreadsheet, you have to make sure that everybody uses the same format to store addresses; otherwise, you won't be able to find out if an address is already in the list. For this purpose the best option is probably to just use the full format, as it's the only unambiguous format. Also be aware that some systems are case sensitive, so you need to define whether to use capitals or not.

For the purpose of making administration easier, an RFC was written to standardize IPv6 address representation. It also discusses the issues that can arise when storing IPv6 addresses in databases and spreadsheets for lookup. You need solid rules for address representation to be able to find addresses. Probably for these cases it is best to use the full address representation, as it is the only unambiguous one. For all other cases, RFC 5952 recommends using the following rules:

- Leading zeros must be suppressed.
- A single 16-bit 0000 field must be represented as 0 and should not be replaced by a double colon.

- Shorten as much as possible. Use a double colon whenever possible.
- Always shorten the largest number of zeros.
- If two blocks of zero are equally long, shorten the first one.
- Use lowercase for a, b, c, d, e, f.

In environments where IPv4 and IPv6 nodes are mixed, another convenient form of IPv6 address notation is to put the values of an IPv4 address into the four low-order bytes of the address. An IPv4 address of `192.168.0.2` can be represented as `x:x:x:x:x:x:192.168.0.2`, and an address of `0:0:0:0:0:0:192.168.0.2` can be written as `::192.168.0.2`. If you prefer, you can also write `::c0a8:2`. This is the hex representation for 192.168.0.2.

For the representation of an IPv6 address followed by a port number, the best way is to put the IPv6 address in square brackets, followed by a colon and the port number. So it may look like this: `[2001:db8::1]:80`.

These recommendations are fully compliant with RFC 4291. But there are many more choices offered by RFC 4291. An IPv6 implementation must always be able to process an address written in any format possible as per RFC 4291.

Prefix Notation

The notation for prefixes has also been specified in RFC 4291. A *global routing prefix* is the high-order bits of an IP address used to identify the subnet or a specific type of address (refer to Table 2-2). It was called the *format prefix* in earlier RFCs. The prefix notation is very similar to the way IPv4 addresses are written in CIDR (Classless Interdomain Routing) notation, and it is also commonly used for subnetted IPv4 addresses. The notation appends the prefix length, written as a number of bits with a slash, which leads to the following format:

```
IPv6 address/prefix length
```

The prefix length specifies how many leftmost bits of the address specify the prefix. This is another way of noting a *subnet mask*. Remember, a subnet mask specifies the bits of the IPv4 address that belong to the network ID. The prefix is used to identify the subnet that an interface belongs to and is used by routers for forwarding. The following example explains how the prefix is interpreted. Consider the IPv6 prefix notation `2001:db8:1200::/40`. To understand this address, let's convert the hex into binary as shown in Table 2-1.

Table 2-1. Understanding prefix notation

Hex notation	Binary notation	Number of bits
2001	0010 0000 0000 0001	16
0db8	0000 1101 1011 1000	16
1200	0001 0010	8
		Total: 40

The *compressed notation* (replacing a sequence of zeros with a double colon) is also applicable to the prefix representation. It should be used carefully, though, because there are often two or more ranges of zeros within an address, and only one can be compressed. The rules in RFC 5952 dictate how to do it, but an IPv6 interface must still be capable of dealing with addresses not complying with RFC 5952, as mentioned above.

Global Routing Prefixes

Table 2-2 outlines the current assignment of reserved prefixes and special addresses, such as link-local addresses or multicast addresses. The major part of the address space (over 80 percent) is unassigned with the exception of a few special cases, mentioned below. This leaves room for future assignments.

Table 2-2. List of assigned prefixes

Allocation	Prefix binary	Prefix hex	Fraction of address space
Global unicast	001	2000::/3	1/8
Link-local unicast	1111 1110 10	fe80::/10	1/1024
Unique-local IPv6 address	1111 110	fc00::/7	
Multicast	1111 1111	ff00::/8	1/256

All address ranges not listed in Table 2-2 are currently reserved or unassigned (with the exceptions mentioned just below). The Internet Assigned Numbers Authority (IANA) currently assigns only out of the binary range starting with 001.

The updated list of address allocations can be found at the following link: *http://www.iana.org/assignments/ipv6-address-space*. You can also find an updated IANA list of special-purpose prefixes here: *http://bit.ly/1pDkTzo*.

Some special addresses are assigned out of the reserved address space with the binary prefix `0000 0000`. These include the *unspecified address*, the *loopback address*, and IPv6 addresses with embedded IPv4 addresses, which I discuss in detail later in this chapter.

Unicast addresses can be distinguished from multicast addresses by their prefixes. Globally unique unicast addresses have a high-order byte starting with `001`. An IPv6 address with a high-order byte of `1111 1111` (`ff` in hex) is always a multicast address. For more information about multicast addresses, refer to the section "Multicast Address" on page 37.

Anycast addresses are taken from the unicast address space, so they can't be identified as anycast just by looking at the prefix. If you assign the same unicast address to multiple interfaces, thereby making it an anycast address, you have to configure the interfaces so they all know that this address is an anycast address. For more information about anycast addresses, refer to the section "Anycast Address" on page 35.

Global Unicast Address

Global unicast addresses are identified by the binary prefix `001`, as shown earlier in Table 2-2. RFC 4291 defines the global unicast address format as shown in Figure 2-1.

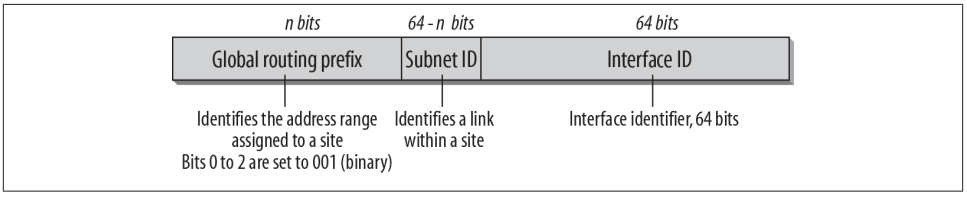

Figure 2-1. Format of the global unicast address

The *global routing prefix* identifies the address range allocated to a site. This part of the address is assigned by the international registry services and the Internet Service Providers (ISPs) and has a hierarchical structure. The *subnet ID* identifies a link within a site. A link can be assigned multiple subnet IDs. A local administrator of a site assigns this part of the address. The *interface ID* identifies an interface on a subnet and must be unique within that subnet. The interface ID is always 64 bits, so therefore an IPv6 subnet is always a /64 subnet. No more subnet mask issues with IPv6.

International Registry Services and Current Address Allocations

The international allocation of IPv6 addresses has been delegated to several regional registry services: ARIN (American Registry for Internet Numbers) for North America and Sub-Saharan Africa; RIPE NCC (Réseau IP Européens Network Coordination Centre) for Europe, the Middle East, Central Asia, and North Africa; APNIC (Asia Pacific Network Information Centre) for the Asia/Pacific region; and LACNIC (Latin American and Caribbean Internet Addresses Registry) for Latin America. AfriNIC (African Network Information Centre) went into operation in 2005 to cover Africa in the future.

Each of these registries has information on its site about address allocation issues, current practices, and procedures.

Several allocations have been made, as listed in Table 2-3.

Table 2-3. Current allocations

Prefix	Allocation	RFC
0100::/64	Discard-Only Address Block	RFC 6666
64:ff9b::/96	IPv4-IPv6 Translator	RFC 6052
2000::/3	Global Unicast Address space Allocations made out of the `2000::/3` space can be viewed at *http://bit.ly/ipv6-add*	RFC 4291
2001::/32	Teredo	RFC 4380
2001:db8::/32	For documentation purposes only, nonroutable	RFC 3849
2002::/16	6to4	RFC 3056
fc00::/7	Unique-local (ULA)	RFC 4193
fe80::/10	Link-scoped unicast	RFC 4291

http://www.iana.org/numbers is a great entry point for global IP address services, current address allocations for both IPv4 and IPv6, and information about how to request IPv6 address services. You can also find an updated IANA list of special-purpose prefixes here: *http://bit.ly/1pDkTzo*.

The address space for 6Bone operation (`3ffe::/16`) was phased out by June 2006 and the prefix returned to the unassigned address pool. It was created in order to allow for global testing of IPv6 while address allocation had not been standardized. Since IPv6 address allocation has been defined, 6Bone hosts have been moved to the official IPv6 address space.

For information on where and how to get IPv6 address space, refer to Chapter 9 on Planning.

So How Large Is This Address Space Again?

In the introduction to this chapter, when we discussed how many addresses IPv6 provides, I promised to make another example later on. So here it comes.

If an ISP gets a /32 prefix, this means there are 32 more bits in the prefix that can be administered by the ISP or its customers.

Imagine: one single /32 is more address space than we ever had with IPv4, because in IPv4 the 32 bits in the address include the host ID, whereas in IPv6 each of these /64 subnets has 64 bits for interface IDs.

Is this a lot? Well, obviously a lot more than we ever had in IPv4. But we still don't know how much it really is.

Let us have a look at the IANA IPv6 pool. By March 2014, there were 138,786 /32 blocks allocated. That seems like a lot if we consider that one of these is more than we ever had in IPv4.

If we calculate how much this is of the total IPv6 address space currently available (`2000::/3`), it makes up for 0.026%.

With the 138,786 /32 blocks, more than 9 billion customers can receive a /48, with which each can create 65,536 /64 subnets.

At *http://www.bgpexpert.com/addrspace-ipv6.php* you can find updated numbers about IPv6 allocations.

So I kindly ask you to use these numbers as your daily mantra when you start working on an IPv6 address plan. One of the biggest challenges when doing that is to get rid of all the address conservation rules that are ingrained in our body cells. Even if something feels really wasteful, it's probably not. And if we have to realize that we were too wasteful, because we have used up the `2000::/3` in 20 years, we still have 7 times more space available to do better. The `2000::/3` is only the binary 001 block, which is currently used for globally unique unicast IPv6 addresses.

The Interface ID

Addresses in the prefix range `001` to `111` should use a 64-bit interface identifier that follows the EUI-64 (Extended Unique Identifier) format (except for multicast addresses with the prefix `1111 1111`). The EUI-64 is a unique identifier defined by the Institute of Electrical and Electronics Engineers (IEEE). Appendix A of RFC 4291 explains how to create EUI-64 identifiers, and more details can be found in the link-specific RFCs, such as "IPv6 over Ethernet" or "IPv6 over FDDI." Chapter 5 contains a short discussion and a list of these RFCs.

An interface uses an identifier following the EUI-64 format during Stateless Address Autoconfiguration. For example, when an interface autoconfigures for a link-local address on an Ethernet interface using its MAC address, the 64-bit interface identifier has to be created from the 48-bit (6-byte) Ethernet MAC address. First, the hex digits `0xfffe` are inserted between the third and fourth bytes of the MAC address. Then the

universal/local bit, the second low-order bit of `0x00` (the first byte) of the MAC address, is complemented. The second low-order bit of `0x00` is `0`, which, when complemented, becomes `1`; as a result, the first byte of the MAC address becomes `0x02`.

Therefore, the IPv6 interface identifier corresponding to the Ethernet MAC address `00-02-b3-1e-83-29` is `02-02-b3-ff-fe-1e-83-29`. This example discusses only the EUI-64 creation process. Many other steps occur during Stateless Address Autoconfiguration.

To learn how IPv6 Stateless Address Autoconfiguration works, refer to Chapter 4.

The link-local address of an interface is the combination of the prefix `fe80::/64` and a 64-bit interface identifier expressed in IPv6 colon-hexadecimal notation. Therefore, the MAC-based link-local address of the previous example interface, with prefix `fe80::/64` and interface identifier `02-02-b3-ff-fe-1e-83-29`, is `fe80::202:b3ff:fe1e:8329`. This is shown in Figure 2-2 and described in RFC 2464, "Transmission of IPv6 Packets over Ethernet Networks."

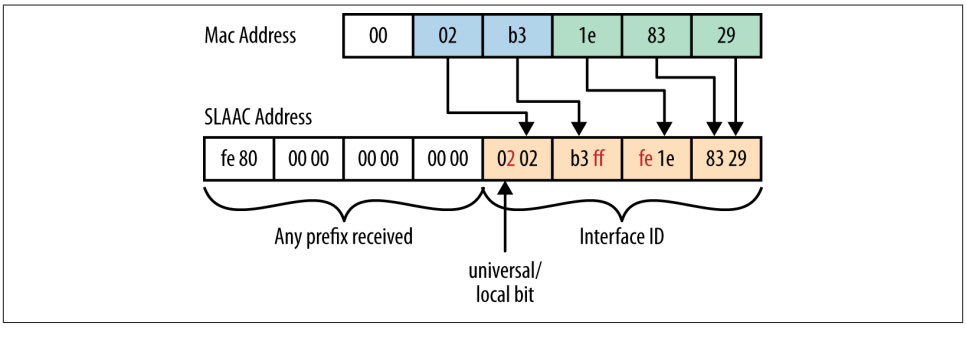

Figure 2-2. How the MAC address is converted to an interface ID

The figure shows how this Interface Identifier is combined with the link-local prefix. As indicated, it can be combined with any other prefix received. (More on this in Chapter 4 on Neighbor Discovery.) There is a discussion going on at the time of writing that advises against using Interface IDs that are based on hardware information. See below for more information.

Address Privacy

The privacy of autoconfigured IPv6 addresses using the interface identifier was an issue discussed in the IETF in the early days of IPv6. If an IPv6 address is built using the MAC identifier, your Internet access could be traced even across networks, because this identifier is unique to your interface. It is important to understand that an IPv6 node can have an address based on the interface identifier, but this is not a requirement. As an alternative, the IPv6 device can have an address like the ones currently used with IPv4, either static and manually configured or dynamically assigned by a DHCP server. RFC 4941, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," introduces another type of address available only in IPv6 that contains a random number in place of the hardware address. This interface ID can also change over time. It is sometimes also called a *temporary address*. It is generated in addition to the EUI-64 interface ID. The temporary address is then used for outgoing communications and the EUI-64 based address for server functions and incoming connections.

An Internet device that is a target for IP communication—for instance, a web or FTP server—needs a unique and stable IP address. But a host running a browser or an FTP client does not need to have the same address every time it connects to the Internet. Some DHCPv6 implementations support the generation of random interface identifiers according to RFC 4941. This way, they use DHCPv6 to manage their address space but prevent anyone from topology mapping their network or tracking their nodes.

With the address architecture in IPv6, you can choose between two types of addresses:

Unique stable IP addresses
: Assigned through manual configuration, a DHCP server, or Stateless Address Autoconfiguration using the EUI-64 interface identifier or another address type created with a permanent IID (see below).

Temporary transient IP addresses
: Assigned using a random number that changes in regular intervals and can be used in place of the stable interface identifier.

While the temporary privacy addresses provide some security by complicating the task of eavesdroppers and other information collectors (e.g., in logfiles, headers) to correlate the activities of a host, they can become challenging in other areas. From a network management perspective, they increase the complexity of event logging, troubleshooting, access control, and quality of service. As a result, some organizations disabled the use of temporary addresses even at the expense of reduced privacy. The need to have stable addresses that do not frequently change the way Privacy addresses do leads to the definition of stable privacy addresses in RFC 7217, "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)." The goal is to have IPv6 addresses that are stable within a subnet, but change when the host moves from one network to another and are not based on any hardware

identifier. This method then applies to all prefixes a host may use, such as link-local, global, or unique-local.

At the same time there is currently a draft in discussion (at the time of writing in early 2014) with the title "Recommendation on Stable IPv6 Interface Identifiers" (*draft-ietf-6man-default-iids-00*). It recommends using stable SLAAC addresses, which are not based on hardware identifiers. As mentioned before, concerns have been raised that embedding hardware information in an IPv6 address creates security and privacy risks. Draft "Privacy Considerations for IPv6 Address Generation Mechanisms" (*draft-ietf-6man-ipv6-address-generation-privacy-01.txt*) offers a great discussion on the pros and cons of each Interface ID generation option.

Be aware, when going through all these considerations, that the Interface ID is not the only way to track users. DNS names, cookies, browser fingerprints, and application-layer usernames can all be used to link a host's activities together.

Special Addresses

There are a number of special addresses that we need to discuss. The first part of the IPv6 address space with the prefix of `0000 0000` is reserved. Out of this prefix, special addresses have been defined as follows:

The unspecified address

The unspecified address has a value of `0:0:0:0:0:0:0:0` and is therefore also called the *all-zeros address*. It is comparable to `0.0.0.0` in IPv4. It indicates the absence of a valid address, and it can, for example, be used as a Source address by a host during the boot process when it sends out a request for address configuration information. If you apply the notation conventions discussed earlier in this chapter, the unspecified address can also be abbreviated as `::`. It should never be statically or dynamically assigned to an interface, and it should not appear as a Destination IP address or within an IPv6 Routing header. It is sometimes used in configuration files for software to tell a program to use any IPv6 address configured on an interface.

The loopback address

The IPv4 loopback address, `127.0.0.1`, is probably familiar to you. It is helpful in troubleshooting and testing the IP stack because it can be used to send a packet to the protocol stack without sending it out on the subnet. With IPv6, the loopback address works the same way and is represented as `0:0:0:0:0:0:0:1`, abbreviated as `::1`. It should never be statically or dynamically assigned to an interface.

The next sections describe different types of addresses that have been specified to be used with different transition mechanisms, which can be used in the migration to IPv6. These virtual interfaces are commonly called *pseudo-interfaces*.

A description of the transition mechanisms can be found in Chapter 7.

IPv6 Addresses with Embedded IPv4 Addresses

Because the transition to IPv6 will be gradual, two special types of addresses have been defined for backward compatibility with IPv4. Both are described in RFC 4291:

IPv4-compatible IPv6 address (deprecated)
: This type of address is used to tunnel IPv6 packets dynamically over an IPv4 routing infrastructure. IPv6 nodes that use this technique are assigned a special IPv6 unicast address that carries an IPv4 address in the low-order 32 bits. This address type has so far rarely been used and was deprecated in RFC 4291. New or updated implementations will no longer need to support this type of address.

IPv4-mapped IPv6 address
: This type of address is used to represent the addresses of IPv4-only nodes as IPv6 addresses. An IPv6 node can use this address to send a packet to an IPv4-only node. The address also carries the IPv4 address in the low-order 32 bits of the address.

Figure 2-3 shows the format of both these addresses.

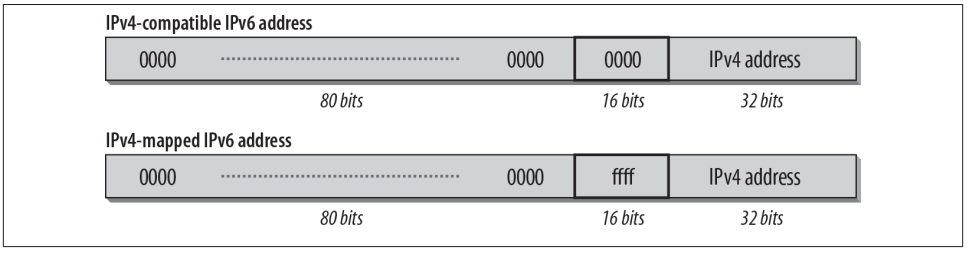

Figure 2-3. Format of IPv6 addresses with an embedded IPv4 address

The two addresses are pretty much the same. The only difference is the 16 bits in the middle. When they are set to 0, the address is an IPv4-compatible IPv6 address; if these bits are set to 1, it is an IPv4-mapped IPv6 address.

6to4 Addresses

The IANA has permanently assigned a 13-bit TLA identifier for 6to4 operations within the global unicast address range (001). 6to4 is one of the mechanisms defined to let IPv6 hosts or networks communicate over an IPv4-only infrastructure. I describe 6to4 in Chapter 7, and it is specified in RFC 3056. The 6to4 TLA identifier is `0x0002`. The address format is shown in Figure 2-4.

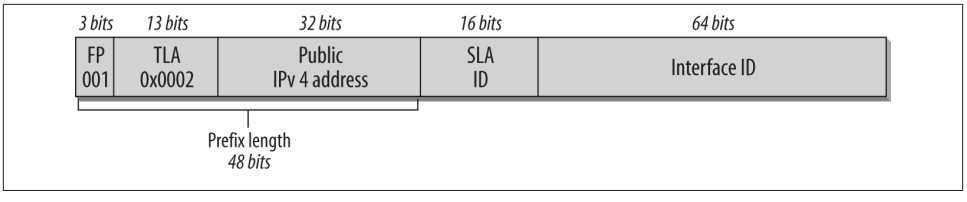

Figure 2-4. Format of the 6to4 address

The prefix has a total length of 48 bits. The IPv4 address in the prefix must be a public IPv4 address and is represented in hexadecimal notation. For instance, if you configure an interface for 6to4 with an IPv4 address of `62.2.84.115`, the 6to4 prefix is `2002:3e02:5473::/48`. Through this interface, all IPv6 hosts on this link can tunnel their packets over the IPv4 infrastructure.

The 6to4 specification was written when the global unicast address format according to RFC 2374 was current, so it uses the old terms and formats (format prefix, TLA, SLA).

6rd Addresses

In 2010, a specification was published called 6rd (IPv6 Rapid Deployment). It is described in Chapter 7 and specified in RFC 5969. The address format is based on 6to4 and shown in Figure 2-5.

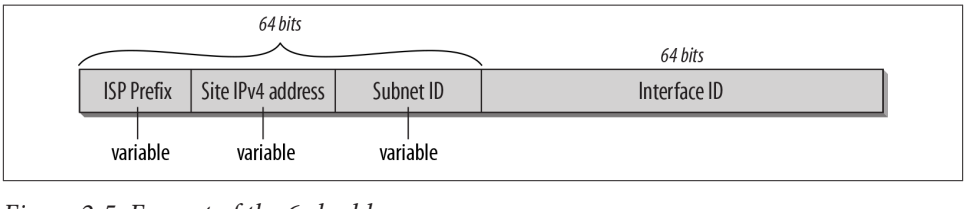

Figure 2-5. Format of the 6rd address

The main difference in the address is that 6rd does not use a special prefix like 6to4 and also doesn't have a fixed boundary at /48. The prefix has a total length of 64 bits and is divided into the ISP prefix and the site IPv4 address. As the figure shows, these two parts are of variable length. If the provider would use his /32 prefix and add the full IPv4 address of the site, he would end up giving out /64 subnets to his customers. In most cases this is not recommended. Even home sites will need multiple subnets in the future. Depending on the ISP's environment, address architecture, and customer structure, there are many ways to design the 6rd address. One option could be if the ISP has a /28, he can then add 32 bits of IPv4 address and give out /60 to his customers. Or, if his IPv4 address plan allows to aggregate customers in, let's say, a /8 prefix (the customer's IPv4 addresses), the size of the 6rd prefix would be one of the following:

- A /52 if the provider has a /28 (28 + 24 for the aggregated customer IPv4 block)
- A /56 if the provider has a /32 (32 + 24 for the aggregated customer IPv4 block)

There are discussions going on in several regions, such as RIPE and ARIN, to make this easier by assigning larger 6rd prefixes to ISPs.

The important point here is to assign prefixes to home users that allow them to have multiple subnets. Refer to Chapter 9 for a description of regional Registry policies and home networks.

ISATAP Addresses

The *Intra-Site Automatic Tunnel Addressing Protocol* (ISATAP) is an automatic tunneling mechanism specified in RFC 5214. It is designed for dual-stack nodes that are separated by an IPv4-only infrastructure. It treats the IPv4 network as one large link-layer network and allows those dual-stack nodes to automatically tunnel between each other using any format of IPv4 address. ISATAP uses a type identifier of `0xFE` for specifying an IPv6 address with an embedded IPv4 address. The format of an ISATAP address is shown in Figure 2-6.

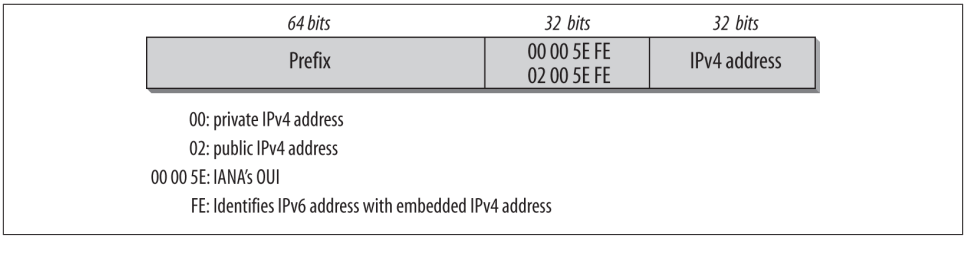

Figure 2-6. Format of the ISATAP address

The first 64 bits follow the format of the global unicast address. IANA owns the IEEE Organizationally Unique Identifier (OUI) `00-00-5E` and specifies the EUI-48 format interface identifier assignments within that OUI. Within the first 16 bits, a type identifier shows whether the IPv4 address is from the private range (`0000`) or a globally unique address (`0200`). The next eight bits contain a type identifier to indicate that this is an IPv6 address with an embedded IPv4 address. The type identifier is `0xFE`. The last 32 bits contain the embedded IPv4 address, which can be written in dotted decimal notation or in hexadecimal representation.

Assume we have a host with an IPv4 address of `192.168.0.1` and the host is assigned a 64-bit prefix of `2001:db8:510:200::/64`. The ISATAP address for this host is `2001:db8:510:200:0:5efe:192.168.0.1`. Alternatively, you can use the hexadecimal representation for the IPv4 address, in which case the address is written `2001:db8:510:200:0:5efe:c0a8:1`. The link-local address for this host is `fe80::5efe:192.168.0.1`.

Teredo Addresses

Teredo is a mechanism designed to provide IPv6 connectivity to hosts that sit behind one or more NATs. This is done by tunneling the IPv6 packet within UDP. The mechanism consists of Teredo clients, servers, and relays. The Teredo relays are IPv6 routers sitting between the Teredo service and the native IPv6 network. Teredo is specified in RFC 4380. It was expected that this service would be common until ISPs upgraded to native IPv6 services. Current Internet statistics show that this is not the case. You can refer to the Google statistics to see how the red line representing 6to4 and Teredo traffic declined to almost zero.

A Teredo address has the format shown in Figure 2-7.

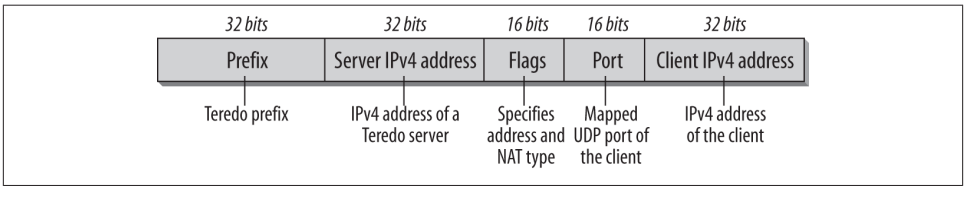

Figure 2-7. Format of the Teredo address

The prefix has a length of 32 bits. The global Teredo IPv6 Service Prefix is `2001:0000:/32`. The server IPv4 address field has a length of 32 bits and contains the IPv4 address of a Teredo server. The flags field has 16 bits and specifies the type of address and NAT in use. The 16-bit port field contains the mapped UDP port of the Teredo service on the client and the client IPv4 address field contains the mapped IPv4

address of the client. In this format, both the mapped UDP port and the mapped IPv4 address of the client are obfuscated: each bit in the address and port number is reversed.

To learn how IPv6 and IPv4 can coexist using these addresses, refer to Chapter 7.

Cryptographically Generated Addresses

To increase security for Neighbor Discovery (ND), which is discussed in Chapter 4, RFC 3972 defines cryptographically generated addresses (CGAs). RFC 3972 has been updated by RFCs 4581 and 4982. CGAs contain a cryptographic hash of the public key as part of the Interface ID. The corresponding private key can then be used to sign messages sent from this address. This prevents attackers from taking over an IPv6 address and can be used in environments where no PKI (Public Key) infrastructure is present.

Link-Local and Unique Local IPv6 Addresses

With IPv4, organizations often use IP addresses from the private ranges as defined in RFC 1918. The addresses reserved for private use should never be forwarded over Internet routers but should instead be confined to the organization's network. For connection to the Internet, Network Address Translation (NAT) maps internal private addresses to publicly registered IPv4 addresses.

The original IPv6 specification allocated two separate address spaces (scopes) for link- and site-local use, both identified by their prefixes. The prefix for site-local addresses was `fec0::/10`.

The site-local address has been deprecated in RFC 3879. Too many potential problems arose in the application of this address. It has been replaced by the Unique Local IPv6 Address, also called ULA (see below).

A *link-local* address is for use on a single link and should never be routed. It doesn't need a global prefix and can be used for Autoconfiguration mechanisms, for Neighbor Discovery, and on networks with no routers, so it is useful for creating temporary networks. Let's say you meet your friend in a conference room and you want to share files on your computers. You can connect your computers using a wireless network or a cross cable between your Ethernet interfaces, and you can share files without any special configuration by using the link-local address.

The replacement for site-local addresses is called *Unique Local IPv6 Unicast Address (ULA)*, or *Local IPv6 Address* for short. It is specified in RFC 4193. These addresses are globally unique but should not be routed to the global Internet. They are designed to be used within corporate sites or confined sets of networks.

The characteristics of unique local IPv6 unicast addresses are the following:

- Have a unique, global prefix, which allows for filtering at network boundaries
- Allow for private connection of networks without the risk of address conflicts and the consequence of having to renumber one of the sites
- Are independent of ISP
- Can be used for internal communication without an Internet connection
- Can be used by applications just like regular global unicast addresses

The format of these addresses is shown in Figure 2-8.

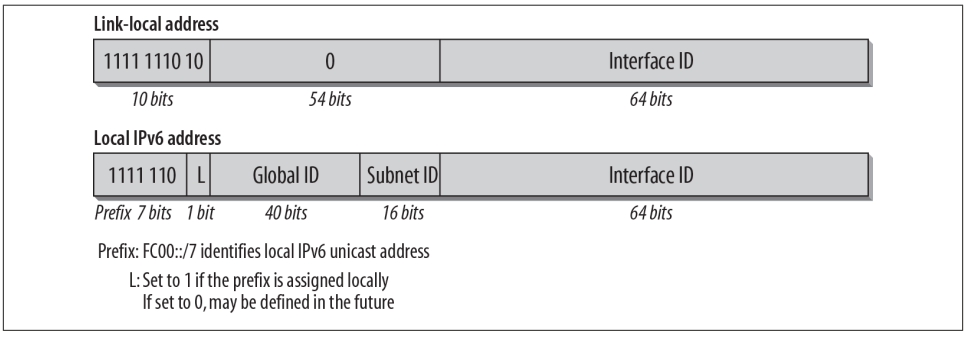

Figure 2-8. Address formats for link- and site-local use

In hexadecimal notation, a link-local address is identified by the prefix `fe80::/10`. For the unique local IPv6 address, RFC 4193 specifies a prefix of `fc00::/7`. The 8th bit is currently set to `1` and specifies local administration of the prefix. Setting the 8th bit to `0` may be used in the future for centrally administrated addresses. For the moment, it was decided to standardize only a locally assigned version. The centrally assigned form may be defined in the future if a strong need is identified.

In the meantime, you can use the Sixxs unofficial registration site and find out. You will also find other cool IPv6 information and tools there.

So for locally administered addresses, we currently have a hexadecimal prefix of `fd00::/8`. It is followed by the 40 bits for the global ID, which is randomly created to ensure a high probability of uniqueness; 16 bits used for subnet ID; and 64 bits for the interface identifier. You may still find the deprecated site-local address with the prefix `fec0::/10` if you use older implementations, but it should not be used for new implementations or deployments but be replaced by either global unicast addresses or ULAs.

Make sure that your global ID is generated using the *Pseudo-Random Global ID Algorithm* defined in RFC 4193. This algorithm includes values such as time, hardware identifiers, and other system-specific values, among others. This is to ensure that your prefix is going to be unique and there should be no ULA collision when merging your network with any other ULA network.

As mentioned previously, these local addresses should not be routed to the Internet. Border routers should be configured to filter these prefixes. Local addresses should not appear in global DNS servers. They can be used on your internal, private DNS server.

Link-local addresses (`fe80::/10`) are by default assigned through Autoconfiguration. ULAs have to be assigned by configuring the local prefix on your routers (Router Advertisement) or through DHCPv6.

If you are interested in the reasons for deprecating the site-local address, refer to RFC 3879. Find a discussion about whether and when to use ULAs in Chapter 9.

Anycast Address

Anycast addresses are designed to provide redundancy and load balancing in situations where multiple hosts or routers provide the same service. Anycast was not created for IPv6; it was defined in RFC 1546 in 1993 as an experimental specification to be used with IPv4. The RFC allots a special prefix for anycast, which would make an anycast address recognizable as such based on the prefix. Anycast was meant to be used for services such as DNS and HTTP. The RFC discusses possible modifications to TCP to deal with these addresses that are not globally unique.

In practice, anycast has not been implemented as it was designed to be. Often a method called *shared unicast address* is chosen. This method is implemented by assigning a regular unicast address to multiple interfaces and creating multiple entries in the routing table. In this case, the network and transport layer assume that it is a globally unique IP address. If it is not, the mechanism to deal with ambiguous addresses needs to be built into the application. An exception to this rule is if the application uses independent stateless request/reply transactions—for instance, DNS over UDP. The root DNS servers

in the Internet are set up using shared unicast addresses. As this procedure does not require any support from the network layer, it can also be used with IPv6.

From the beginning, the IPv6 developers considered anycast to be incorporated in the network layer according to RFC 1546. No special prefix was assigned. IPv6 anycast addresses are in the same address range as global unicast addresses, and each participating interface must be configured to have an anycast address. Within the region where the interfaces containing the same anycast addresses are, each host must be advertised as a separate entry in the routing tables.

Within one network where a group of routers can provide access to a common routing domain, they can be assigned a single address. When a client sends a packet to this address, it will be forwarded to the next available router. One example is the 6to4 relay anycast address that is specified in RFC 3068 and described in Chapter 7. The Mobile IPv6 specification also uses anycast addresses.

When using anycast addresses, we have to be aware that the sender has no control over which interface the packet will be delivered to. This decision is taken on the level of the routing protocol. When a sender sends multiple packets to an anycast address, the packets may arrive at different destinations due to routing table instability or changes during the requests. If there is a series of requests and replies or if the packet has to be fragmented, this may cause problems.

The *subnet-router anycast address*, which is defined in RFC 4291 and shown in Figure 2-9, is a required anycast address.

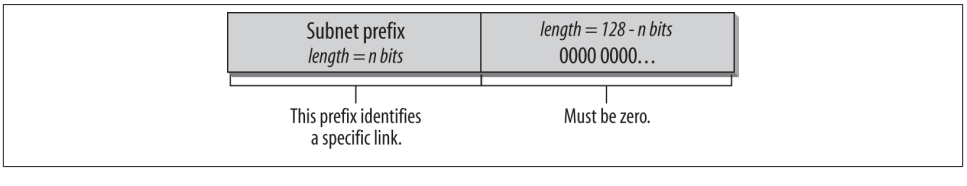

Figure 2-9. Format of the subnet-router anycast address

Basically, the address looks like a regular unicast address with a prefix specifying the subnet and an identifier set to all zeros. A packet sent to this address will be delivered to one router on that subnet. All routers are required to support the subnet-router anycast address for subnets to which they have interfaces.

A reserved subnet anycast address can have one of two formats, as shown in Figure 2-10.

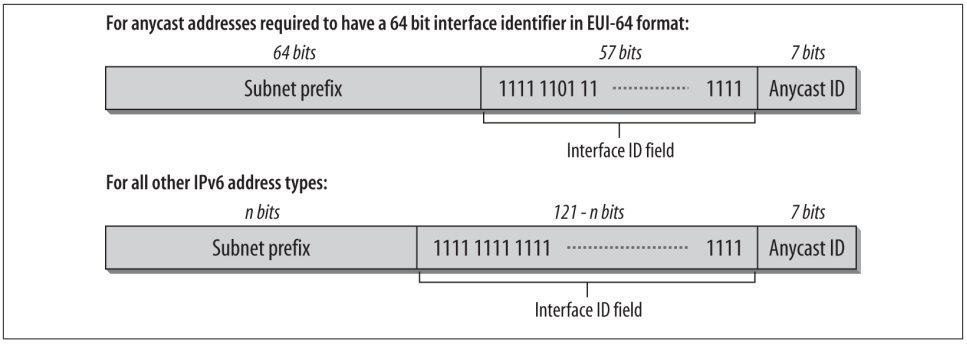

Figure 2-10. General format of anycast addresses

RFC 2526 specifies that within each subnet, the highest 128 interface identifier values are reserved for assignment as subnet anycast addresses. Currently, the anycast IDs listed in Table 2-4 have been reserved.

Table 2-4. Reserved anycast IDs

Decimal	Hexadecimal	Description
127	7F	Reserved
126	7E	Mobile IPv6 Home-Agents anycast
0–125	00–7D	Reserved

The main difference between this form of using anycast and the shared unicast address is that in the latter, the application needs to support anycast, while in the former, this support is avoided if possible. Guidelines of how to use this and modifications to existing stateful transport protocols are needed.

If you are interested in more information and background on anycast, refer to RFC 7094, "Architectural Considerations of IP Anycast." It provides an overview of the history of anycast, discusses different architectural models and principles, and covers anycast in IPv6 as well as deployment considerations.

Multicast Address

This section covers the multicast address format. For a description of multicast and Multicast Listener Discovery (MLD), also known as Multicast Group Management, refer to Chapter 4. For a general overview and summary of multicast topics, refer to Chapter 5.

A multicast address is an identifier for a group of nodes identified by the high-order byte `ff`, or `1111 1111` in binary notation (refer to Table 2-2 earlier in the chapter). The

multicast prefix is `ff00::/8`. A node can belong to more than one multicast group. When a packet is sent to a multicast address, all members of the multicast group process the packet. Multicast exists in IPv4, but it has been redefined and improved for IPv6. The multicast address format is shown in Figure 2-11.

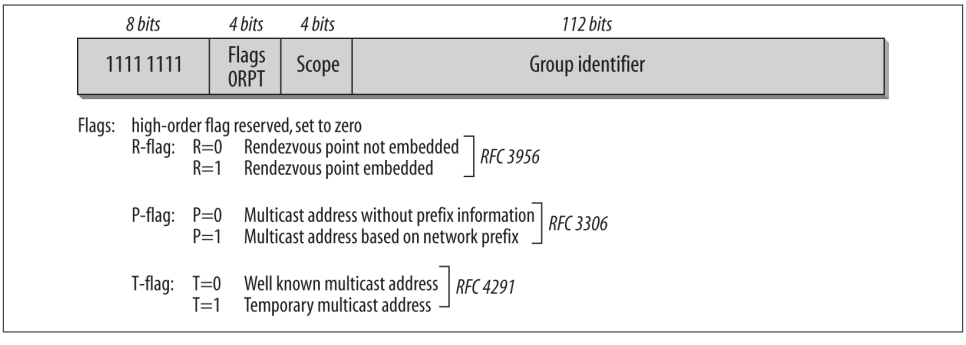

Figure 2-11. Format of the multicast address

The first byte identifies the address as a multicast address. The next four bits are used for Flags, defined as follows: the first bit of the Flag field must be zero; it is reserved for future use. The second bit indicates whether this multicast address embeds the *Rendezvous Point*. A Rendezvous Point is a point of distribution for a specific multicast stream in a multicast network (RFC 3956). The third bit indicates whether this multicast address embeds prefix information (discussed later in this chapter; see also RFC 3306). The last bit of the Flag field indicates whether this address is permanently assigned—i.e., one of the well-known multicast addresses assigned by the IANA—or a temporary multicast address. A value of zero for the last bit defines a well-known address; a value of one indicates a temporary address. The Scope field is used to limit the scope of a multicast address. The possible values are shown in Table 2-5.

Table 2-5. Values for the Scope field

Value	Description
0	Reserved
1	*Interface-local scope (used to be called node-local scope in earlier specs)*
2	*Link-local scope*
3	Reserved
4	Admin-local scope
5	*Site-local scope*
6, 7	Unassigned
8	Organization-local scope
9, A, B, C, D	Unassigned

Value	Description
E	*Global scope*
F	Reserved

The boundaries of zones of a scope other than interface-local, link-local, and global must be defined and configured by network administrators. The reserved scopes should not be used. RFC 4007, "IPv6 Scoped Address Architecture," specifies the architectural characteristics, expected behavior, textual representation, and usage of IPv6 addresses of different scopes.

Well-Known Multicast Addresses

According to RFC 4291, the last 112 bits of the address carry the multicast group ID. RFC 3307, "Allocation Guidelines for IPv6 Multicast Addresses," refers to a 32-bit group ID.

RFC 2375 defines the initial assignment of IPv6 multicast addresses that are permanently assigned. Some assignments are made for fixed scopes, and some assignments are valid over all scopes. Table 2-6 gives an overview of the addresses that have been assigned for fixed scopes. Note the scope values that are listed in Table 2-5 in the byte just following the multicast identifier of `ff` (first byte).

Table 2-6. Well-known multicast addresses

Address	Description
Interface-local scope	
ff01::1	All-nodes address
ff01::2	All-routers address
ff01::fb	mDNSv6
Link-local scope	
ff02::1	All-nodes address
ff02::2	All-routers address
ff02::4	DVMRP routers
ff02::5	OSPFIGP
ff02::6	OSPFIGP designated routers
ff02::7	ST routers
ff02::8	ST hosts
ff02::9	RIP routers
ff02::a	EIGRP routers
ff02::b	Mobile agents
ff02::d	All PIM routers

Address	Description
ff02::e	RSVP encapsulation
ff02::16	All MLDv2-capable routers
ff02::6a	All snoopers
ff02::fb	mDNSv6
ff02::1:1	Link name
ff02::1:2	All DHCP agents
ff02::1:3	Link-local Multicast Name Resolution (LLMNR)
ff02::1:4	DTCP Announcement
ff02::1:ffXX:XXXX	Solicited-node address
ff02::2:ff00::/104	Node Information Queries
Site-local scope	
ff05::2	All-routers address
ff05::fb	mDNSv6
ff05::1:3	All DHCP servers
ff05::1:1000 to ff05::1:13ff	Service location (SLP) version 2

The term *node-local scope* from RFC 2375 has been changed to *interface-local scope*, so you may encounter both terms. The list for the permanently assigned multicast addresses that are independent of scopes is long, and it is available in Appendix B and in RFC 2375. All those addresses are noted beginning with `ff0X`; `X` is the placeholder for a variable scope value.

The IPv4 broadcast address is replaced by the link-local all-nodes multicast address `ff02::1`.

Find the most updated list of multicast address assignments here: *http://www.iana.org/assignments/ipv6-multicast-addresses*.

As an example, let's look at the one described in RFC 2373. There is a multicast group ID defined for all NTP servers. The multicast group ID is `0x101`. This group ID can be used with different scope values as follows:

`ff01::101`
: All NTP servers on the same node as the sender.

`ff02::101`
: All NTP servers on the same link as the sender.

`ff05::101`
: All NTP servers on the same site as the sender.

`ff0e::101`
: All NTP servers in the Internet.

Temporarily assigned multicast addresses are meaningful only within a defined scope.

Multicast addresses should not be used as a Source address in IPv6 packets or appear in any routing header.

For the management of multicast, IPv6 uses Multicast Listener Discovery (MLD) based on ICMPv6.

To learn how multicast addresses are managed, refer to the section "Multicast Listener Discovery" on page 110 in Chapter 4. To get a general overview and summary of multicast, refer to Chapter 5.

Solicited-Node Multicast Address

The *solicited-node multicast address* is a multicast address that every node must join for every unicast and anycast address it is assigned. It is used in Neighbor Discovery, which is described in Chapter 4. RFC 4291 specifies the solicited-node multicast address.

In the IPv4 world, an ARP request (used to determine the MAC address of an interface) is sent to the MAC-layer broadcast address and therefore examined by every interface on the link. In the IPv6 world, resolving the MAC address of an interface is done by sending a Neighbor Solicitation message (discussed in Chapter 4) to the solicited-node multicast address, and not to the link-local all-nodes multicast address. This way only the node registered to this multicast address will examine the packet.

This address is formed by taking the low-order 24 bits of an IPv6 address (the last part of the host ID) and appending those bits to the well-known prefix `ff02:0:0:0:0:1:ff00::/104`. Thus, the range for solicited-node multicast addresses goes from `ff02:0:0:0:0:1:ff00:0000` to `ff02:0:0:0:0:1:ffff:ffff`.

For example, our host Marvin has the IPv6 address `fe80::202:b3ff:fe1e:8329`. The corresponding solicited-node multicast address is `ff02::1:ff1e:8329`. If this host has other IPv6 unicast or anycast addresses, each one will have a corresponding solicited-node multicast address for which the host must be registering.

Mapping Multicast Addresses to MAC Addresses

When a packet is sent to an IPv6 multicast address, the IPv6 address must be mapped to a MAC address on the link layer. The format of the Ethernet MAC multicast address is specified in RFC 2464. The first two bytes of an IPv6 MAC multicast address are `0x3333`. The following four bytes correspond to the last four bytes of the IPv6 multicast address.

Figure 2-12 shows how a multicast address is mapped to a MAC address.

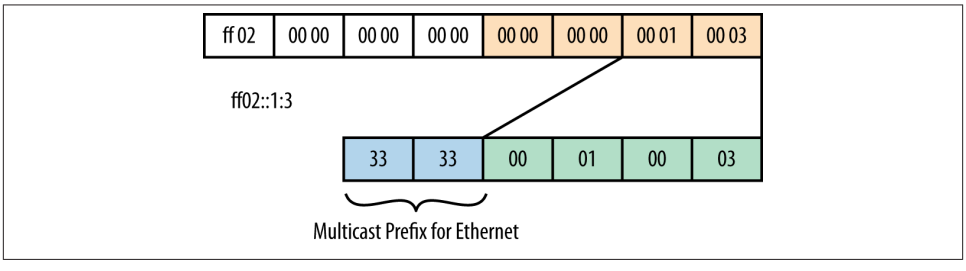

Figure 2-12. MAC representation of an IPv6 multicast address

The link-local scoped multicast address of `ff02::1:3` is mapped to the MAC address of `33:33:00:01:00:03`. The mapping for other media types is specified in separate RFCs. You can find more information about other media types in Chapter 5 or by searching the RFC database.

Dynamic Allocation of Multicast Addresses

The multicast address architecture has been extended in RFC 3306. It contains definitions that allow the allocation of unicast prefix-based addresses and of source-specific multicast addresses. It is based on a modified multicast address format that contains prefix information. The goal of this specification is to reduce the number of protocols needed for the allocation of multicast addresses.

Figure 2-13 shows the format of the extended multicast address.

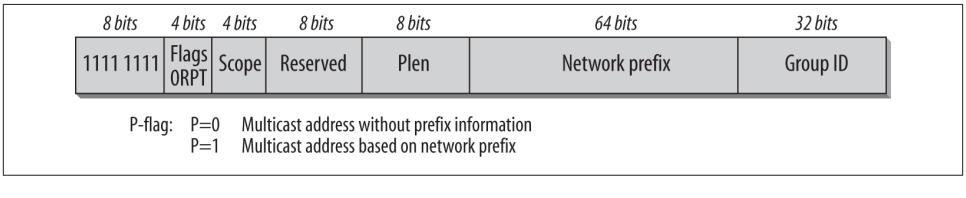

Figure 2-13. Format of the extended multicast address

In the original specification, the Flags field only uses the last bit (T) to specify whether the multicast address is a well-known or temporary one. The extended format shown here uses the second last bit (P) to indicate whether the multicast address assignment is based on the network prefix (value 1) or not (value 0). A P setting of 1 indicates that it is a multicast address following the extended format. The use of the scope field has not changed. If the P flag is set to 1, the eight bits following the Scope field are reserved and set to 0. The next eight bits (PLen) specify the length of the prefix in the prefix field. If the prefix length is smaller than 64 bits, the unused bits in the prefix field should be set to 0. The group ID uses 32 bits. Note that when P is set to 1 (extended multicast address), the T flag should also be set to 1 (temporary multicast address).

Multicast Listener Discovery is used for multicast management. There are two versions, MLDv1 and MLDv2. MLDv2 supports *source-specific multicast*. For an overview of source-specific multicast, refer to RFC 3569. In the traditional multicast model called any-source multicast (ASM), a multicast listener cannot control the source of the data it wants to receive. With source-specific multicast (SSM), an interface can register for a multicast group and specify the source(s) for the data. SSM can be implemented using MLDv2 and the extended multicast address format.

For a source-specific multicast address, the T and the P flag are set to 1. Prefix length and network prefix are both set to 0. This leads to a multicast prefix of `ff3x:/32`, where `x` is a scope value. The source address in the IPv6 header identifies the owner of the multicast address. All SSM addresses have the format `ff3X::/96`.

Refer to RFC 3307, "Allocation Guidelines for IPv6 Multicast Addresses," for more information.

RFC 4489, "Link-Scoped Multicast Address Format," defines an extension to the multicast addressing architecture of the IPv6 protocol. The extension allows for the use of interface identifiers to allocate link-local scoped multicast addresses. In this multicast address, the flags field is set to binary `0011`; the Scope field is set to 2 for link-local scope; the pLen field is set to `ff` (all ones in binary); and the 64 bits of the network ID field are used for the interface identifier. The group ID is generated to indicate a multicast application and needs to be unique only on this host. It is designed for environments in which link-local scope multicast addresses are used.

Required Addresses

The standard specifies that each host must assign the following addresses to identify itself:

- Its link-local address for each interface
- Any assigned unicast and anycast addresses
- The loopback address
- The all-nodes multicast address
- Solicited-node multicast address for each of its assigned unicast and anycast addresses
- Multicast addresses of all other groups to which the host belongs

A router needs to recognize all of the previous addresses, plus the following:

- The subnet-router anycast address for the interfaces for which it is configured to act as a router on each link
- All anycast addresses with which the router has been configured
- The all-routers multicast address
- Multicast addresses of all other groups to which the router belongs

Default Address Selection

The architecture of IPv6 allows an interface to have multiple addresses. The addresses may differ in scope (link-local, global) or state (preferred, deprecated); they may be part of mobility (home-address, care-of-address) or multihoming situation; or they may be permanent public addresses or virtual tunnel interfaces. Dual-stack hosts have IPv6 and IPv4 addresses. The result is that IPv6 implementations that need to initiate a connection are often faced with a choice between multiple Source and Destination addresses.

Imagine a situation where a client issues a DNS request for an external service and receives a global IPv6 and a public IPv4 address back. If this client has a private IPv4 address and a global IPv6 address, it might make sense to use IPv6 to access this external service. But if the client has a tunneled IPv6 address and a public IPv4 address, it should choose the IPv4 address for connecting to the service. These are types of situations and choices that will have to be dealt with in the future world of mixed networks, some IPv4-only, some IPv6-only, and some dual-stack. The way this is dealt with depends on the implementations. Application developers have to be aware of this and try to provide

mechanisms that will make their applications behave optimally in every possible environment.

RFC 6724, "Default Address Selection for IPv6," defines two general algorithms, one for Source address selection and the other for Destination address selection. All IPv6 nodes (host and router) have to implement RFC 6724. The algorithms specify default behavior for IPv6 nodes. The algorithms do not override choices made by applications, upper-layer protocols, or other policies. The RFC contains a policy table that, similar to a routing table, is a longest-matching-prefix lookup table. To each prefix listed, a *precedence* and a *label* are assigned. The precedence is used for sorting Destination addresses; the label value is used to define policies that associate a specific Source address to a given Destination address.

Here's a summary of the most important rules:

- Address pairs of the same scope or type (link-local, global) are preferred.
- A smaller scope for the Destination address is preferred (use the smallest scope possible).
- A preferred (nondeprecated) address is preferred.
- Transitional addresses (e.g., ISATAP or 6to4 addresses) are not used if native IPv6 addresses are available.
- If all criteria are similar, address pairs with the longest common prefix are preferred.
- Prefer outgoing interface.
- Prefer addresses in a prefix advertised by the next hop.
- Prefer matching label.
- For the Source address, temporary addresses are preferred over public addresses.
- Use longest matching prefix.
- In Mobile IP situations, home addresses are preferred over care-of addresses.

In short, the default table prefers native IPv6 over native IPv4 and NATed IPv4 over 6to4 and other tunnels.

The rules in RFC 6724 are to be used in all situations when nothing else is specified. Implementations should provide mechanisms to override the default policy and to configure address selection individually to adapt default address selection to the specifics of the network. RFC 7078 defines a DHCP option that allows the administrator to distribute an address selection policy which overrides the default policy from RFC 6724.

Now that you are familiar with the extended address space and the IPv6 address types, the next chapter discusses the IPv6 header and the Extension headers.

References

The following are lists of the most important RFCs and drafts mentioned in this chapter. Sometimes I include additional subject-related RFCs for your personal further study.

RFCs

- RFC 1546, "Host Anycasting Service," 1993
- RFC 1918, "Address Allocation for Private Internets," 1996
- RFC 2101, "IPv4 Address Behaviour Today," 1997
- RFC 2365, "Administratively Scoped IP Multicast," 1998
- RFC 2464, "Transmission of IPv6 Packets over Ethernet Networks," 1998
- RFC 2471, "IPv6 Testing Address Allocation (6Bone), " 1998
- RFC 2526, "Reserved IPv6 Subnet Anycast Addresses," 1999
- RFC 2710, "Multicast Listener Discovery (MLD)" for IPv6," 1999
- RFC 2908, "The Internet Multicast Address Allocation Architecture," 2000
- RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds" (6to4), 2001
- RFC 3068, "An Anycast Prefix for 6to4 Relay Routers," 2001
- RFC 3306, "Unicast-Prefix-based IPv6 Multicast," 2002
- RFC 3307, "Allocation Guidelines for IPv6 Multicast Addresses," 2002
- RFC 3569, "An Overview of Source-Specific Multicast (SSM)," 2003
- RFC 3587, "IPv6 Global Unicast Address Format," 2003
- RFC 3590, "Source Address Selection for the Multicast Listener Discovery (MLD) Protocol," 2003
- RFC 3810, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6," 2004
- RFC 3849, "IPv6 Documentation Address," 2004
- RFC 3879, "Deprecating Site Local Addresses," 2004
- RFC 3956, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address," 2004
- RFC 3972, "Cryptographically Generated Addresses (CGA)," 2005
- RFC 4007, "IPv6 Scoped Address Architecture," 2005

- RFC 4192, "Procedures for Renumbering an IPv6 Network without a Flag Day," 2005
- RFC 4193, "Unique Local IPv6 Unicast Addresses," 2005
- RFC 4291, "IPv6 Addressing Architecture," 2006
- RFC 4380, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," 2006
- RFC 4489, "A Method for Generating Link-Scoped IPv6 Multicast Addresses," 2006
- RFC 4581, "Cryptographically Generated Addresses (CGA) Extension Field Format," 2006
- RFC 4795, "Link-Local Multicast Name Resolution (LLMNR)," 2007
- RFC 4941, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," 2007
- RFC 4982, "Support for Multiple Hash Algorithms in Cryptographically Generated Addresses (CGAs)," 2007
- RFC 5156, "Special Use IPv6 Addresses," 2008
- RFC 5214, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)," 2008
- RFC 5375, "IPv6 Unicast Address Assignment Considerations," 2008
- RFC 5453, "Reserved IPv6 Interface Identifiers," 2009
- RFC 5569, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)," 2010
- RFC 5952, "A Recommendation for IPv6 Address Text Representation," 2010
- RFC 5991, "Teredo Security Updates," 2010
- RFC 6052, "IPv6 Addressing of IPv4/IPv6 Translators," 2010
- RFC 6081, "Teredo Extensions," 2011
- RFC 6085, "Address Mapping of IPv6 Multicast Packets on Ethernet," 2011
- RFC 6164, "Using 127-Bit IPv6 Prefixes on Inter-Router Links," 2011
- RFC 6177, "IPv6 Address Assignment to End Sites," 2011
- RFC 6724, "Default Address Selection for Internet Protocol version 6 (IPv6)," 2012
- RFC 7078, "Distributing Address Selection Policy Using DHCPv6," 2014
- RFC 7094, "Architectural Considerations of IP Anycast," 2014
- RFC 7108, "A Summary of Various Mechanisms Deployed at L-Root for the Identification of Anycast Nodes," 2014
- RFC 7136, "Significance of IPv6 Interface Identifiers," 2014

- RFC 7217, "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)," 2014

Drafts

Drafts can be found at *http://www.ietf.org/ID.html*. To locate the latest version of a draft, refer to *https://datatracker.ietf.org/public/pidtracker.cgi*. You can enter the draft name without a version number and the most current version will come up. If a draft does not show up, it was possibly deleted. If it was published as an RFC, the RFC number will be displayed. *http://tools.ietf.org/wg* is also a very useful site. More information on the process of standardization, RFCs, and drafts can be found in Appendix A.

Here's a list of drafts I refer to in this chapter, as well as interesting drafts that relate to the topics in this chapter:

"Privacy Considerations for IPv6 Address Generation Mechanisms"
draft-ietf-6man-ipv6-address-generation-privacy-01

"Recommendation on Stable IPv6 Interface Identifiers"
draft-ietf-6man-default-iids-00

CHAPTER 3

The Structure of the IPv6 Protocol

This chapter explains the structure of the IPv6 header and compares it to the IPv4 header. It also discusses Extension headers, which are new in IPv6.

Understanding the structure of a protocol header and the type of information that comes with it is the best foundation for working with a protocol. This understanding helps you to identify how the protocol can best be configured and what the options are. It also helps you to identify possible sources of problems and issues when troubleshooting.

The header structure of an IPv6 packet is specified in RFC 2460. The header has a fixed length of 40 bytes. The two fields for Source and Destination addresses each use 16 bytes (128 bits), so there are only 8 bytes for general header information. The base IPv6 header is therefore much simpler and leaner than the IPv4 header, allowing for more efficient processing and, as we will see, more flexibility in extending the protocol to meet future needs.

General Header Structure

In IPv6, five fields from the IPv4 header have been removed:

- Header Length
- Identification
- Flags
- Fragment Offset
- Header Checksum

The Header Length field was removed because it is not needed in a header with a fixed length. In IPv4, the minimum header length is 20 bytes, but if options are added, it can be extended in 4-byte increments up to 60 bytes. Therefore, with IPv4, the information

about the total length of the header is important. In IPv6, options are defined in Extension headers (covered later in this chapter).

The Identification, Flags, and Fragment Offset fields are the fields that are used for the fragmentation of a packet in the IPv4 header. *Fragmentation* happens if a large packet has to be sent over a network that supports only smaller packet sizes. In that case, the IPv4 router splits the packet into smaller slices and forwards multiple packets. The destination host collects the packets and reassembles them. If only one packet is missing or has an error, the whole transmission has to be redone; this is very inefficient. In IPv6, a host learns the Path Maximum Transmission Unit (MTU) size through a procedure called *Path MTU Discovery*, which has been defined in RFC 1981. In IPv4 the *Don't Fragment Bit* (DF Bit) was used for Path MTU Discovery. If a router could not forward a packet due to its size and could not fragment it because the DF Bit was set, it sent back an ICMP "Packet Too Big" message to the source node. If a sending IPv6 host wants to fragment a packet, it will use an Extension header to do so. IPv6 routers along the path of a packet do not provide fragmentation as they did with IPv4. So the router always sends back a "Packet Too Big" message to the source node. This is the reason that the Identification, Flags, and Fragment Offset fields were removed from the IPv6 header and will be inserted in an Extension header by the source host if needed. I explain Extension headers later in this chapter.

Path MTU Discovery is explained in Chapter 4.

The Header Checksum field was removed to improve processing speed. If routers do not have to check and update checksums, processing becomes much faster. At the time when IPv4 was developed, checksumming at the media access level wasn't common, so the checksum field in the IPv4 header made sense. Today, the risk for undetected errors and misrouted packets is minimal. There is also a checksum field at the transport layer (UDP and TCP). With IPv4, a UDP checksum is optional; with IPv6, a UDP checksum is mandatory. Since IP is a *best-effort delivery protocol*, it is the responsibility of upper layer protocols to ensure integrity.

The Traffic Class field replaces the "Type of Service" field in IPv4. IPv6 has a different mechanism to handle preferences. The Protocol Type field in IPv4 has been renamed to Next Header field and the Time-to-Live (TTL) field has been renamed to Hop Limit. A Flow Label field was added.

The Fields in the IPv6 Header

By becoming familiar with the fields of the IPv6 header, you will better understand how IPv6 works.

Figure 3-1 provides an overview of the IPv6 header. The fields are discussed in detail in the following list.

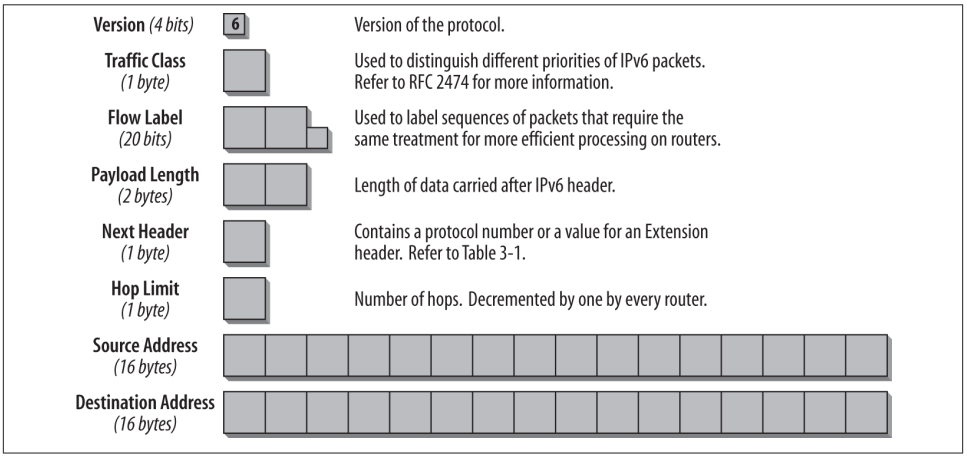

Figure 3-1. Fields in the IPv6 header

Figure 3-1 shows that even though the header has a total size of 40 bytes, which is twice as long as a default IPv4 header, it has actually been streamlined because most of the header is taken up by the two 16-byte IPv6 addresses. That leaves only 8 bytes for other header information.

Version (4 bits)
: This 4-bit field contains the version of the protocol. In the case of IPv6, the number is 6. Version number 5 could not be used because it was already assigned to the experimental stream protocol (RFC 1819).

Traffic class (1 byte)
: This field replaces the Type of Service field in IPv4. It facilitates the handling of real-time data and any other data that requires special handling, and sending nodes and forwarding routers can use it to identify and distinguish between different classes or priorities of IPv6 packets.

  RFC 2474, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," explains how the Traffic Class field in IPv6 can be used. RFC 2474 uses the term *DS Field* to refer to the Type of Service field in the IPv4 header, as well as to the Traffic Class field in the IPv6 header.

Refer to Chapter 5 for more information on the use of the Traffic Class field.

Flow label (20 bits)

This field distinguishes packets that require the same treatment in order to facilitate the handling of real-time traffic. A sending host can label sequences of packets with a set of options. Routers keep track of flows and can process packets belonging to the same flow more efficiently because they do not have to reprocess each packet's header. The flow label and address of the source node uniquely identify the flow. Nodes that do not support the functions of the Flow Label field are required to pass the field unchanged when forwarding a packet and to ignore the field when receiving a packet. All packets belonging to the same flow must have the same Source and Destination IP address.

The use of the Flow Label field is experimental and is currently still under discussion at the IETF at the time of writing. Refer to Chapter 5 for more information.

Payload length (2 bytes)

This field specifies the *payload*—i.e., the length of data carried after the IP header. The calculation in IPv6 is different from the one in IPv4. The Length field in IPv4 includes the length of the IPv4 header, whereas the Payload Length field in IPv6 contains only the data following the IPv6 header. Extension headers are considered part of the payload and are therefore included in the calculation.

The fact that the Payload Length field has 2 bytes limits the maximum packet payload size to 64 KB. IPv6 has a *Jumbogram Option*, which supports bigger packet sizes if needed. The Jumbogram Option is carried in a Hop-by-Hop Option header (discussed later in this chapter). Jumbograms are relevant only when IPv6 nodes are attached to links that have a link MTU greater than 64 KB; they are specified in RFC 2675.

Next Header (1 byte)

In IPv4, this field is called the Protocol Type field, but it was renamed in IPv6 to reflect the new organization of IP packets. If the next header is UDP or TCP, this field will contain the same protocol numbers as in IPv4—for example, protocol number 6 for TCP or 17 for UDP. But if Extension headers are used with IPv6, this field contains the type of the next Extension header. Extension headers are located

between the IP header and the TCP or UDP header. Table 3-1 lists possible values in the Next Header field. The new IPv6-related headers are bold.

Table 3-1. Values in the Next Header field

Value	Description
0	In an IPv4 header: reserved and not used **In an IPv6 header: Hop-by-Hop Option header following**
1	Internet Control Message Protocol (ICMPv4)—IPv4 support
2	Internet Group Management Protocol (IGMPv4)—IPv4 support
4	IPv4
5	Stream Protocol (RFC 1819)
6	TCP
8	Exterior Gateway Protocol (EGP)
9	IGP—any private interior gateway (used by Cisco for their IGRP)
17	UDP
41	**IPv6**
43	**Routing header**
44	**Fragmentation header**
45	Interdomain Routing Protocol (IDRP)
46	Resource Reservation Protocol (RSVP)
47	General Routing Encapsulation (GRE)
50	**Encapsulating Security Payload header**
51	**Authentication header**
58	**ICMPv6**
59	**No Next Header for IPv6**
60	**Destination Options header**
88	EIGRP
89	OSPF
103	PIM
108	IP Payload Compression Protocol
115	Layer 2 Tunneling Protocol (L2TP)
132	Stream Control Transmission Protocol (SCTP)
135	*Mobility Header (Mobile IPv6)*
140	Shim6 (RFC 5533)
143–252	Unassigned
253, 254	Used for experimentation and testing (RFC 3692)
255	Reserved

Header type numbers derive from the same range of numbers as protocol type numbers, and therefore should not conflict with them.

Go to *http://www.iana.org/assignments/protocol-numbers* for the current list.

Hop limit (1 byte)
: This field is analogous to the TTL field in IPv4. Originally, the IPv4 TTL field contained a number of seconds, indicating how long a packet can remain in the network before being destroyed. In fact, IPv4 routers simply decrement this value by one at each hop. This field has been renamed to Hop Limit in IPv6 to reflect the purpose. The value in this field expresses a number of hops. Every forwarding node decrements the number by one. If a router receives a packet with a Hop Limit of 1, it decrements it to 0, discards the packet, and sends the ICMPv6 message "Hop Limit exceeded in transit" back to the sender.

Source address (16 bytes)
: This field contains the IP address of the originator of the packet.

Destination address (16 bytes)
: This field contains the IP address of the intended recipient of the packet. This can be the ultimate destination or if, for example, a Routing header is present, the address of the next hop router.

Figure 3-2 shows the IPv6 header in the trace file.

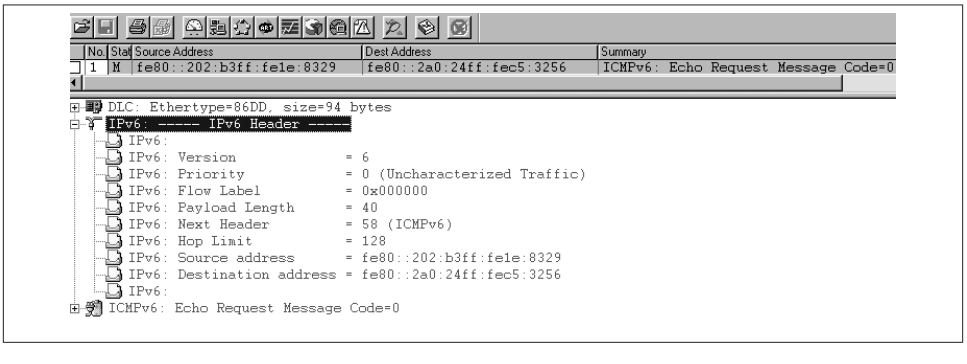

Figure 3-2. The IPv6 header in a trace file

This trace file shows all of the header fields discussed and how they can be presented in a trace file. The Version field is set to 6 for IPv6. The Traffic Class (Priority) and Flow

Label fields are not used in this packet and are set to 0. The Payload Length is 40, and the Next Header value is set to 58 for ICMPv6. The Hop Limit is set to 128, and the Source and Destination addresses contain the link-local addresses of my IPv6 nodes. The first line in the detail window shows `Ethertype 0x86DD`. This value indicates that this is an IPv6 packet. For IPv4, the value would be `0x0800`. This field can be used to set an analyzer filter for all native IPv6 packets.

Analyzer tools can decode packets in different ways. If you use another version or another type of analyzer, your decode may look slightly different. The difference is not in the packet, but in the way the packet is presented in the analyzer interface.

Extension Headers

The IPv4 header can be extended from a minimum of 20 bytes to a maximum of 60 bytes in order to specify options such as Security Options, Source Routing, or Timestamping. This capacity has rarely been used because it causes a performance hit. For example, IPv4 hardware forwarding implementations have to pass the packet containing options to the main processor (software handling).

The simpler a packet header, the faster the processing is. IPv6 has a new way to deal with options that has substantially improved processing: it handles options in additional headers called *Extension headers*. Extension headers are inserted into a packet only if the options are needed. And in most cases, the Extension headers are only processed by the final destination, not by intermediate devices.

The current IPv6 specification defines six Extension headers, which must be supported by all IPv6 nodes:

- Hop-by-Hop Options header
- Routing header
- Fragment header
- Destination Options header
- Authentication header
- Encapsulating Security Payload header

There can be zero, one, or more than one Extension header in an IPv6 packet. Extension headers are placed between the IPv6 header and the upper-layer protocol header. Each Extension header is identified by the Next Header field in the preceding header. The Extension headers are examined or processed only by the node identified in the Destination address field of the IPv6 header. If the address in the Destination address field

is a multicast address, the Extension headers are examined and processed by all the nodes belonging to that multicast group. Extension headers must be strictly processed in the order in which they appear in the packet header.

There is one exception to the rule that only the destination node will process an Extension header. If the Extension header is a Hop-by-Hop Options header, the information it carries must be examined and processed by every node along the path of the packet. The Hop-by-Hop Options header, if present, must immediately follow the IPv6 header. It is indicated by the value 0 in the Next Header field of the IPv6 header (see Table 3-1 earlier in this chapter).

The first four Extension headers are described in RFC 2460. The Authentication header is described in RFC 4302, and the Encapsulating Security Payload header in RFC 4303. An update to the format of future Extension headers has been defined in RFC 6564.

This architecture is very flexible for developing additional Extension headers for future uses as needed. New Extension headers can be defined and used without changing the IPv6 header. A good example is the Mobility header defined for Mobile IPv6 (RFC 6275), which is discussed in Chapter 8.

Figure 3-3 shows how Extension headers are used.

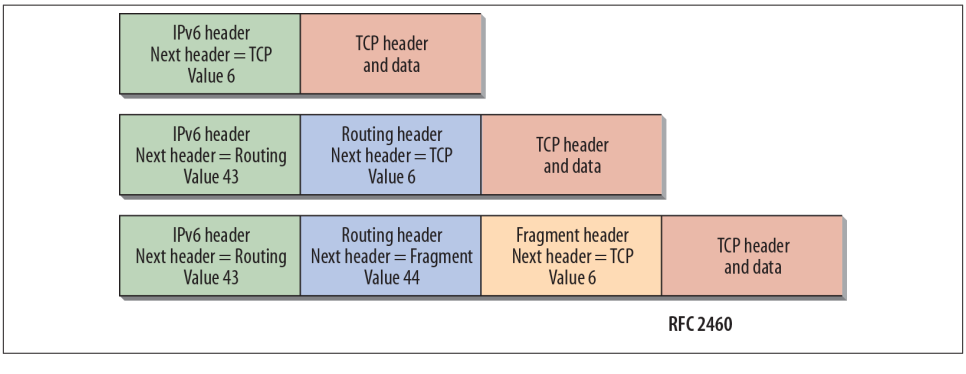

Figure 3-3. The use of Extension headers

Each Extension header's length is a multiple of eight bytes so that subsequent headers can always be aligned. If a node is required to process the next header but cannot identify the value in the Next Header field, it is required to discard the packet and send an ICMPv6 Parameter Problem message back to the source of the packet.

For details on ICMPv6 messages, refer to Chapter 4.

If more than one Extension header is used in a single packet, the following header order should be used (RFC 2460):

1. IPv6 header
2. Hop-by-Hop Options header
3. Destination Options header (for options to be processed by the first destination that appears in the IPv6 Destination address field, plus subsequent destinations listed in the Routing header)
4. Routing header
5. Fragment header
6. Authentication header
7. Encapsulating Security Payload header
8. Destination Options header (for options to be processed only by the final destination of the packet)
9. Upper-Layer header

RFC 2460 leaves some space for interpretation. Although this is the recommended order, IPv6 nodes must attempt to process Extension headers in any order. But it is still strongly advised that sources of IPv6 packets use the recommended order unless newer specifications revise it.

In cases when IPv6 is encapsulated in IPv4, the Upper-Layer header can be another IPv6 header and can contain Extension headers that have to follow the same rules.

Hop-by-Hop Options Header

The Hop-by-Hop Options header carries optional information that must be examined by every node along the path of the packet. It must immediately follow the IPv6 header and is indicated by a Next Header value of 0. For example, the Router Alert (RFC 2711) uses the Hop-by-Hop Options header for protocols such as Resource Reservation Protocol (RSVP), Multicast Listener Discovery (MLD) messages, or the Jumbogram Option. With IPv4, the only way for a router to determine whether it needs to examine a datagram is to at least partially parse upper-layer data in all datagrams. This process slows down the routing process substantially. With IPv6, in the absence of a Hop-by-Hop Options header, a router knows that it does not need to process router-specific

information and can route the packet immediately to the final destination. If there is a Hop-by-Hop Options header, the router needs only to examine this header and does not have to look further into the packet.

The format of the Hop-by-Hop Options header is shown in Figure 3-4.

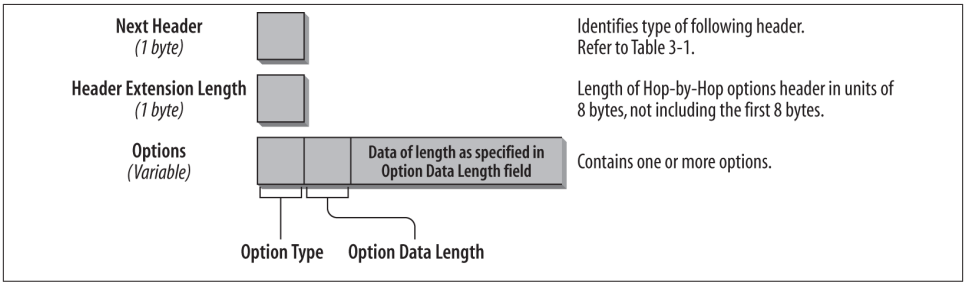

Figure 3-4. Format of the Hop-by-Hop Options header

The following list describes each field:

Next Header (1 byte)
: The Next Header field identifies the type of header that follows the Hop-by-Hop Options header. The Next Header field uses the values listed in Table 3-1, shown earlier in this chapter.

Extension Header Length (1 byte)
: This field identifies the length of the Hop-by-Hop Options header in eight-byte units. The length calculation does not include the first eight bytes. So if the header is shorter than eight bytes, this field contains the value 0.

Options (variable size)
: There can be one or more options. The length of the options is variable and is determined in the Header Extension Length field.

The Option Type field, the first byte of the Options fields, contains information about how this option must be treated in case the processing node does not recognize the option. The value of the first two bits specifies the actions to be taken:

- 00: Skip and continue processing.
- 01: Discard the packet.
- 10: Discard the packet and send ICMP Parameter Problem, Code 2, message to the packet's Source address pointing to the unrecognized option type.
- 11: Discard the packet and send ICMP Parameter Problem, Code 2, message to the packet's Source address only if the destination is not a multicast address.

The third bit of the Options Type field specifies whether the option information can change en route (value 1) or does not change en route (value 0).

Option Type Jumbogram

This Hop-by-Hop Option Type supports the sending of IPv6 Jumbograms. The IPv6 Payload Length field supports a maximum packet size of 65,535 bytes. The Jumbo Payload Option (RFC 2675) allows for larger packets to be sent.

In the IPv6 header of a packet with the Jumbo Payload option, the Payload Length field is set to 0. The Next Header field contains the value 0, which indicates a Hop-by-Hop Options header. The Option Type value of 194 indicates the Jumbo Payload option. The Jumbo Payload Length field has 32 bits and therefore supports the transmission of packets that are between 65,536 and 4,294,967,295 bytes. RFC 2675 also defines extensions to UDP and TCP that have to be implemented on hosts that need to support the sending of Jumbograms. All devices on the path of a Jumbogram must support the option.

Option Router Alert

This Option Type indicates to the router that the packet contains important information to be processed when forwarding the packet. The option is currently used mostly for MLD (Multicast Listener Discovery) and RSVP (Resource Reservation Protocol). It is specified in RFC 2711, which has been updated by RFC 6398.

RSVP uses control packets containing information that needs to be interpreted or updated by routers along the path. These control packets use a Hop-by-Hop Options header, so only routers process the packet. Regular data packets do not have this Extension header and are therefore forwarded immediately without further inspection by the router.

The first three bits of the Option Type field are set to 0. A router that doesn't know this option ignores it and forwards the packet. In the remaining five bits of the first byte, the option type 5 is specified. The Option Data Length field contains the value 2, which indicates that the following value field has a length of two bytes (refer to Figure 3-4).

The list of Router Alert values can be found at the following link: *http://www.iana.org/assignments/ipv6-routeralert-values*.

Figure 3-5 show the Hop-by-Hop Options header in the trace file.

```
No.   Source                  Destination             Protocol  Info
   10 fe80::d4e4:d1e6:c310:aef ff02::16               ICMPv6 Multicast Listener Report Message v2
⊞ Frame 10: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0
⊞ Ethernet II, Src: CadmusCo_dd:ed:d3 (08:00:27:dd:ed:d3), Dst: IPv6mcast_00:00:00:16 (33:33:00:00:00:16)
⊟ Internet Protocol Version 6, Src: fe80::d4e4:d1e6:c310:aef (fe80::d4e4:d1e6:c310:aef), Dst: ff02::16 (ff02::16)
  ⊞ 0110 .... = Version: 6
  ⊞ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 36
    Next header: IPv6 hop-by-hop option (0)
    Hop limit: 1
    Source: fe80::d4e4:d1e6:c310:aef (fe80::d4e4:d1e6:c310:aef)
    Destination: ff02::16 (ff02::16)
  ⊟ Hop-by-Hop Option
      Next header: ICMPv6 (58)
      Length: 0 (8 bytes)
    ⊟ IPv6 Option (Router Alert)
        Type: Router Alert (5)
        Length: 2
        Router Alert: MLD (0)
    ⊞ IPv6 Option (PadN)
⊞ Internet Control Message Protocol v6
```

Figure 3-5. The Hop-by-Hop Options header in a trace file

The screenshot shows the details of packet number 10. It is an MLDv2 Multicast Listener Report Message. As mentioned before, these multicast registration messages always have a Hop-by-Hop Options header (Next Header value zero), because this is a packet that the router doesn't have to forward, but that contains information that it must process. You can see the Hop Limit set to 1 for MLD messages; the Destination address of `ff02::16` is the multicast address for MLDv2 routers; the Hop-by-Hop Options header contains the next header field with the value 58 for ICMPv6; the Length field and the Router Alert option type 5 with the value field zero for MLD.

Routing Header

The *Routing header* is used to give a list of one or more intermediate nodes that should be visited on the packet's path to its destination. In the IPv4 world, this is called the *Loose Source Route* option. The Routing header is identified by a Next Header value of 43 in the preceding header. Figure 3-6 shows the format of the Routing header.

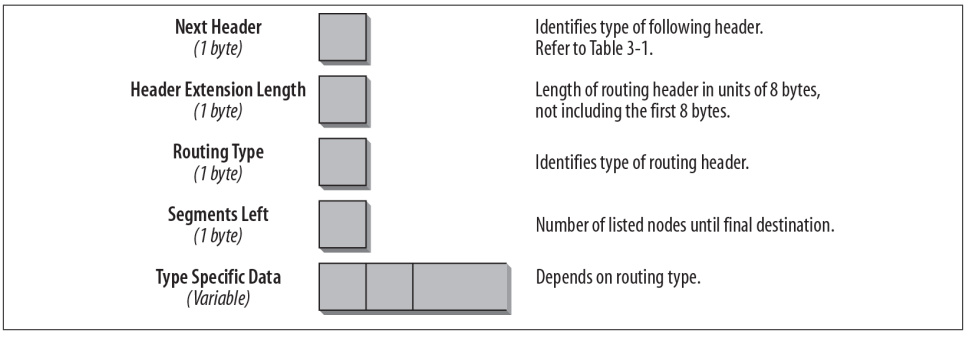

Figure 3-6. Format of the Routing header

The following list describes each field:

Next Header (1 byte)
: The Next Header field identifies the type of header that follows the Routing header. It uses the same values as the IPv4 Protocol Type field (see Table 3-1 earlier in this chapter).

Extension Header Length (1 byte)
: This field identifies the length of the Routing header in 8-byte units. The length calculation does not include the first 8 bytes.

Routing Type (1 byte)
: This field identifies the type of Routing header. RFC 2460 describes Routing Type 0, which has been deprecated by RFC 5095 for security reasons. The Mobile IPv6 specification defines a Routing Type 2. (This specification is discussed in Chapter 8.) At the time of writing there are some drafts in progress, which define a new segment routing architecture and a new routing header type called Segment Routing header. Find the links to these drafts in the draft section at the end of this chapter. Whether this specification is going to see the light of day you may know by the time you read these lines.

Segments Left (1 byte)
: This field identifies how many nodes are left to be visited before the packet reaches its final destination.

Type-Specific Data (variable length)
: The length of this field depends on the Routing Type. The complete header is always a multiple of 8 bytes.

If a node processing a Routing header cannot identify a Routing Type value, the action taken depends on the content of the Segments Left field. If the Segments Left field does not contain any nodes to be visited, the node must ignore the Routing header and process the next header in the packet, which is determined by the Next Header field's value. If the Segments Left field is not zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0 message to the packet's Source address pointing to the unrecognized Routing Type. If a forwarding node cannot process the packet because the next link MTU size is too small, it discards the packet and sends an ICMP Packet Too Big message back to the source of the packet.

Figure 3-7 shows the Type 2 Routing header in a trace file.

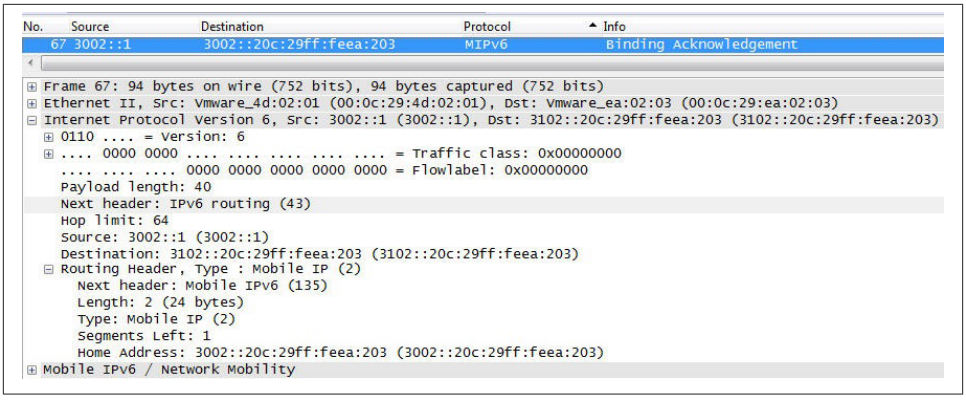

Figure 3-7. Routing header Type 2 in a trace file

To show the Type 2 Routing header we must take a Mobile IPv6 trace, the specification that defines this type of Routing header. The Next Header field within the IPv6 header shows the value 43 for the Routing header. The Routing header contains the fields discussed earlier in this section. Next Header is a Mobility header indicated by a Next Header value of 135 in the routing header. The Header Length contains two 8-byte units, which add up to a total length of 16 bytes (one address). The Segments Left field contains the value 1 because there is one address entry in the Options field. Finally, the Options field lists the Home Address option with the home address.

Refer to Chapter 8 to find out how the Routing header is used for Mobility.

For an example of a new Routing header option, refer to RFC 6554, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)." In Low-power and Lossy Networks (LLNs), routers typically have very constrained memory that only allows for a small number of default routes and no other destinations. This RFC defines the *Source Routing Header* (SRH), which is strictly to be used between RPL routers.

Fragment Header

An IPv6 host that wants to send a packet to an IPv6 destination uses Path MTU discovery to determine the maximum packet size that can be used on the path to that destination. If the packet to be sent is larger than the supported MTU, the source host fragments the packet. Unlike in IPv4, with IPv6 a router along the path does not fragment packets. Fragmentation occurs only at the source host sending the packet. The destination host

handles reassembly. A Fragment header is identified by a Next Header value of 44 in the preceding header. The format of the Fragment header is shown in Figure 3-8.

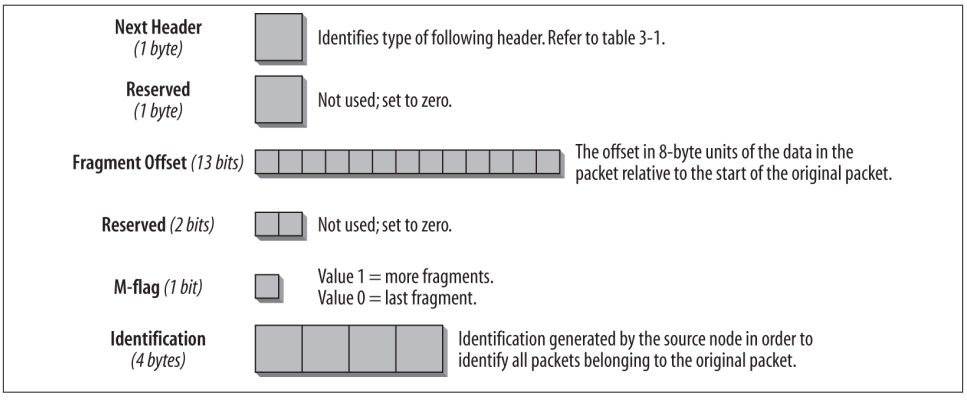

Figure 3-8. Format of the Fragment header

The following list describes each field:

Next Header (1 byte)
: The Next Header field identifies the type of header that follows the Fragment header. It uses the same values as the IPv4 Protocol Type field. (See Table 3-1.)

Reserved (1 byte)
: Not used; set to 0.

Fragment Offset (13 bits)
: The offset in 8-byte units of the data in this packet relative to the start of the data in the original packet.

Reserved (2 bits)
: Not used; set to 0.

M-Flag (1 bit)
: Value 1 indicates more fragments; a value of 0 indicates the last fragment.

Identification (4 Bytes)
: Generated by the source host in order to identify all packets belonging to the original packet. This field is usually implemented as a counter, increasing by one for every packet that needs to be fragmented by the source host.

The Fragment header does not contain a Don't Fragment field as in IPv4. It is not necessary, because routers no longer fragment in IPv6. Only the source host can fragment a packet.

The initial unfragmented packet is referred to as the *original packet*. It has an unfragmentable part that consists of the IPv6 header plus any Extension headers that must be processed by nodes along the path to the destination (i.e., Hop-by-Hop Options header). The fragmentable part of the original packet consists of any Extension headers that need only to be processed by the final destination, plus the Upper-Layer headers and any data. Figure 3-9 (RFC 2460) illustrates the fragmenting process.

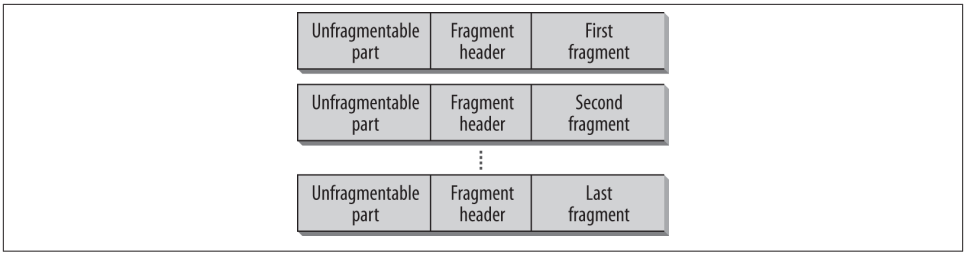

Figure 3-9. Fragmentation with IPv6

The unfragmentable part of the original packet appears in every fragment, followed by the Fragmentation header and then the fragmentable data. The IPv6 header of the original packet has to be slightly modified. The Length field reflects the length of the fragment (excluding the IPv6 header) and not the length of the original packet.

The destination node collects all the fragments and reassembles them. The fragments must have identical Source and Destination addresses and the same identification value in order to be reassembled. If all fragments do not arrive at the destination within 60 seconds after the first fragment, the destination will discard all packets. If the destination has received the first fragment (Offset = zero), it sends back an ICMPv6 Fragment Reassembly Time Exceeded message to the source.

Figure 3-10 shows a Fragment header.

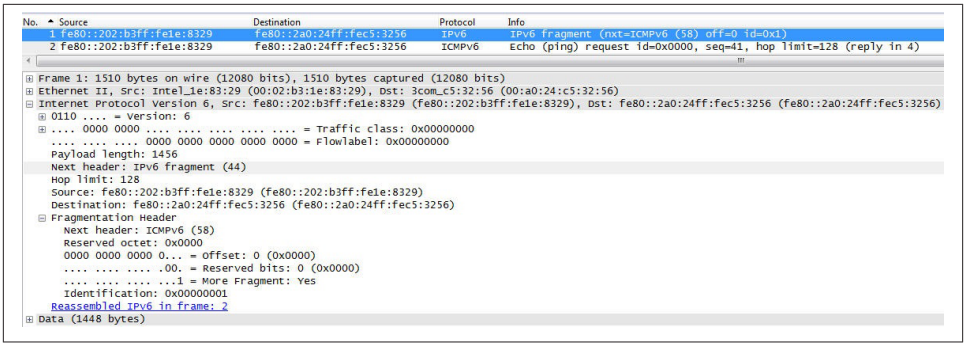

Figure 3-10. Fragment header in a trace file

The whole fragment set consists of two packets, the first of which is shown in Figure 3-10. In the IPv6 header, the Payload Length field has a value of 1,456, which is the length of the fragmentation header and this one fragment, not the length of the whole original packet. The Next Header field specifies the value 44, which is the value for the Fragment header. This field is followed by the Hop Limit field and the Source and Destination IP addresses. The first field in the Fragment header is the Next Header field. Because this is a ping, it contains the value 58 for ICMPv6. And because this is the first packet in the fragment set, the value in the Offset field is 0 and the M-Flag is set to 1, which means there are more fragments to come. The Identification field is set to 1 and has to be identical in all packets belonging to this fragment set. Figure 3-11 shows the second packet of the fragment set.

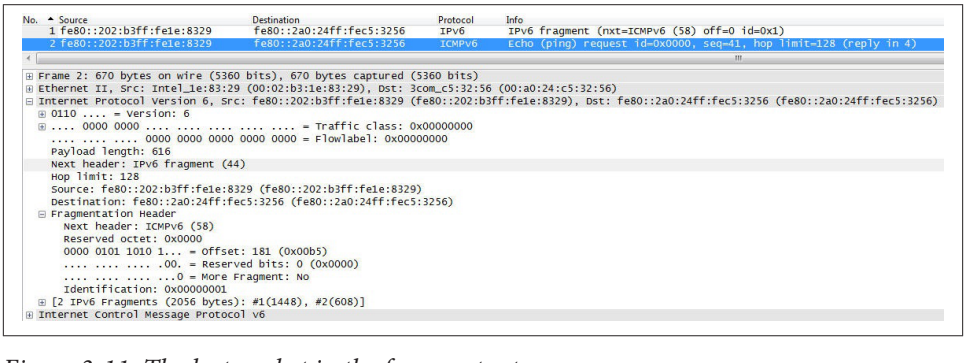

Figure 3-11. The last packet in the fragment set

The second and last packet of this fragment set has an Offset value of `0x00b5`, which translates to 181 in decimal notation, the length of the first fragment. The M-Flag is set to 0, which indicates that it is the last packet and tells the receiving host that it is time to reassemble the fragments. The Identification field is set to 1 in both packets.

The specification in RFC 2460 allows for overlapping fragments, which creates a security issue. RFC 5722, "Handling of Overlapping IPv6 Fragments," explains the security issue, updates RFC 2460 and forbids overlapping fragments. RFC 6980 describes how IPv6 fragmentation can become a security issue by eliminating the effectiveness of securing mechanisms such as RA Guard and forbids the use of IPv6 Fragmentation for traditional Neighbor Discovery messages.

Refer to Chapter 4 for a description of Neighbor Discovery and to Chapter 6 for a discussion of RA Guard and the security implications of the Fragment header.

Destination Options Header

A *Destination Options* header carries optional information that is examined by the destination node only (the Destination address in the IPv6 header). A Next Header value of 60 identifies this type of header. As mentioned previously, the Destination Options header can appear twice in an IPv6 packet. When inserted before a Routing header, it contains information to be processed by the routers listed in the Routing header. When inserted before the upper-layer protocol headers, it contains information for the final destination of the packet. Figure 3-12 shows the format of the Destination Options header.

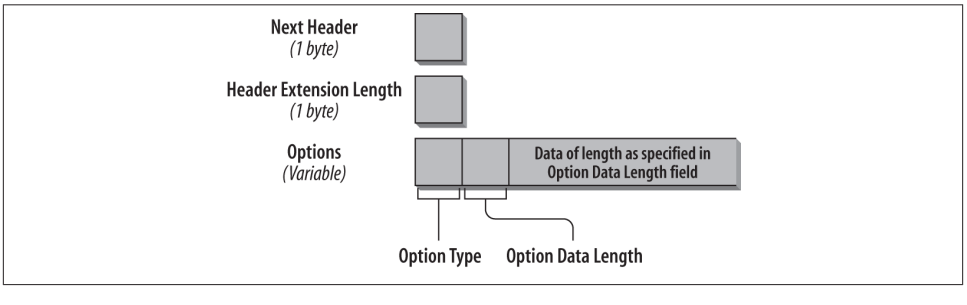

Figure 3-12. Format of the Destination Options header

As you can see, the format is similar to the format of the Hop-by-Hop Options header. The following list describes each field:

Next Header (1 byte)
: The Next Header field identifies the type of header that follows the Destination Options header. It uses the same values listed in Table 3-1, shown earlier in this chapter.

Extension Header Length (1 byte)
: This field identifies the length of the Destination Options header in 8-byte units. The length calculation does not include the first 8 bytes.

Options (variable size)
: There can be one or more options. The length of the options is variable and is determined in the Header Extension Length field.

The Options field is used in a similar way as with the Hop-by-Hop Options header, which I discussed earlier in this chapter. An example of the use of the Destination Options header is Mobile IPv6. You can find a detailed description of Mobile IPv6 in Chapter 8. Another defined Destination Option Header option is the *Tunnel Encapsulation Limit Option* in RFC 2473, "Generic Packet Tunneling in IPv6 Specification," which is used to limit the number of times that a packet can be further encapsulated.

Find the most current list of defined options for the Routing and the Destination Options header at *http://www.iana.org/assignments/ipv6-parameters/*.

Figure 3-13 shows the Destination Options header in the trace file.

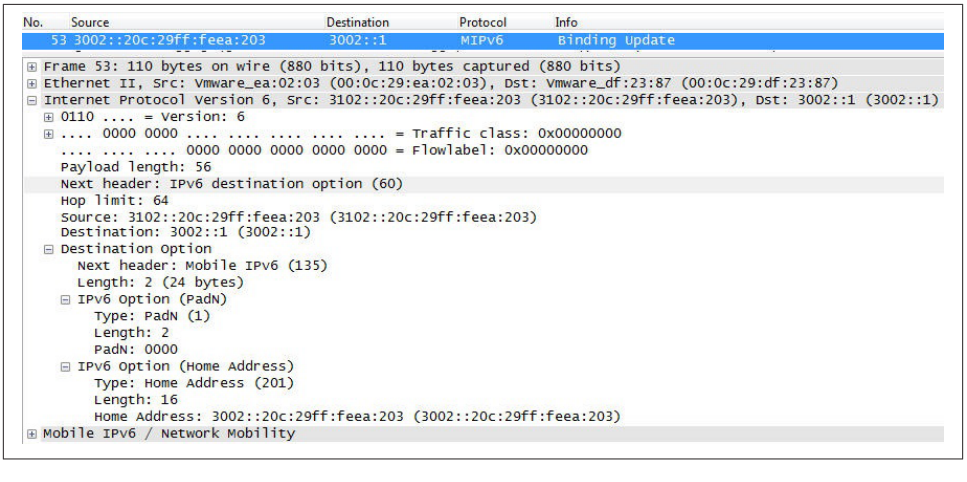

Figure 3-13. Destination Options header in the trace file

To show the Destination Options header, we refer to the Mobile IPv6 trace again. This is a Binding Update message. It uses the Destination Options header with value 60 in the Next Header field of the IP header. The Destination Options header has a Next

Header field with the value 135 for a Mobile IPv6 message and contains the Home Address option and the home address for the mobile node.

Refer to Chapter 8 to find out how the Destination Options header is used for Mobility.

New Extension Header Format

With the exception of the Hop-by-Hop and Routing header, Extension headers are usually only processed by the final destination of a packet. In practice there are devices on the path of a packet, such as routers and firewalls, which are capable of parsing past or ignoring Extension headers at wire speed. In order to accommodate real-world implementations and to optimize Extension header processing and inspection of Extension headers, a new format for Extension headers has been defined in RFC 6564, "A Uniform Format for IPv6 Extension Headers." Figure 3-14 shows the new format.

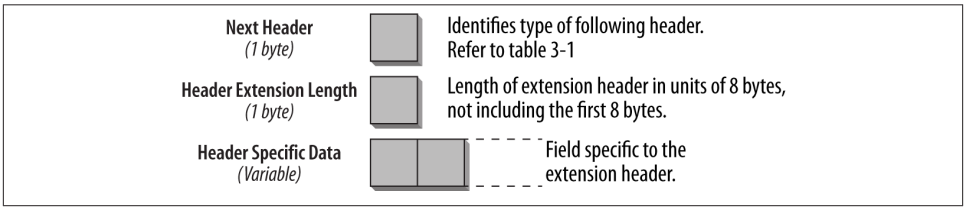

Figure 3-14. The new Extension header format

The following list describes each field:

Next Header (1 byte)
: The Next Header field identifies the type of header that follows the Extension header. It uses the same values listed in Table 3-1, shown earlier in this chapter.

Extension Header Length (1 byte)
: This field identifies the length of the Extension header in 8-byte units. The length calculation does not include the first 8 bytes.

Options (variable size)
: The length of the options is variable and is determined in the Header Extension Length field.

The format of the basic Extension headers described in this chapter will not change. But if new Extension headers are defined in the future, they must follow this format. This means that any device that deals with Extension headers, such as firewalls, must be

capable of properly processing the basic Extension headers, but also new Extension headers using the new format.

Several rules are defined in RFC 6564 and summarized below:

- If possible, no new Extension headers should be defined, but rather new options for the Destination Options header. Only if that is not possible can a new Extension header be defined.
- No new header with hop-by-hop behavior must be created, and new options for the existing Hop-by-Hop Options header should only be created under limited circumstances.

Processing of Extension Headers and Header Chain Length

The base specification in RFC 2460 says that Extension headers are only processed by end nodes (with the exception of the Hop-by-Hop Options header). The goal of this architecture was that new Extension headers can be introduced and only end nodes need to be updated. This process would be transparent to forwarding nodes along the path of the packet. Practice has shown that this is not always applicable. Some routers and a variety of intermediate boxes such as firewalls, load balancers, and packet classifiers, also called middleware, might inspect other parts of the IP header beyond the IPv6 base header. Very often, if they do not recognize an Extension header, they simply drop the packet, which leads to connectivity failures. Also the Hop-by-Hop Extension header is often not handled by high-speed routers or is processed on a slow path.

RFC 7045, "Transmission and Processing of Extension Headers," discusses these issues. While according to the base specification, end nodes should discard Extension headers that they don't recognize, this should not be done by forwarding devices on the path of a packet. Otherwise, these forwarding devices may discard packets with newly defined Extension headers that they don't recognize yet. The RFC says that there should be a policy on these devices to be individually configurable. The default configuration should allow all standard Extension headers. For firewalls, the RFC requires that, in particular, packets containing standard Extension headers are only to be discarded as a result of an intentionally configured policy. For the Hop-by-Hop Extension header the requirement is that all forwarding devices should process it, but implementers have to be aware that this usually happens on a slow path.

Another problem was that there was not one single place where all Extension headers can be found and the number may increase regularly as new specifications come out. So it is difficult for vendors to identify what Extension headers they have to support in their implementations. The RFC therefore defines that there must be a new section in the IANA (Internet Assigned Numbers Authority) IPv6 Parameters section to list all IPv6 Extension header types.

The new IANA registry section for "IPv6 Extension header" on page 153 can be found at *http://bit.ly/1na7H1Q*.

With regard to the header chain (which includes the IPv6 header, any Extension headers, plus the upper protocol header), note the following: in IPv4 we had a fixed upper limit for the size of all IPv4 options in an IPv4 packet. In the IPv6 base specification there is no limit to the number of Extension headers in a packet. So it is possible that when a packet is fragmented, the header chain may span multiple fragments. This causes problems, specifically if firewalls cannot apply rules to fragments, because the information they need is missing in the first fragment. RFC 7112, "Implications of Oversized IPv6 Header Chains," describes the issue and updates RFC 2460 such that the first fragment of a fragmented datagram is required to contain the entire IPv6 header chain.

Now that you are familiar with the IPv6 header and the Extension headers, the next chapter introduces the advanced features of ICMPv6, which offer management functionality not known with ICMPv4.

References

The following is a list of the most important RFCs and drafts mentioned in this chapter. Sometimes I include additional subject-related RFCs for your personal further study.

RFCs

- RFC 791, "Internet Protocol," 1981
- RFC 1812, "Requirements for IP Version 4 Routers," 1995
- RFC 1819, "Internet Stream Protocol Version 2," 1995
- RFC 1981, "Path MTU Discovery for IP version 6," 1996
- RFC 2460, "Internet Protocol, Version 6 (IPv6) Specification," 1998
- RFC 2473, "Generic Packet Tunneling in IPv6 Specification," 1998
- RFC 2474, "Definition of the Differentiated Services Field (DS Field)," 1998
- RFC 2475, "An Architecture for Differentiated Services," 1998
- RFC 2507, "IP Header Compression," 1999
- RFC 2675, "IPv6 Jumbograms," 1999
- RFC 2711, "IPv6 Router Alert Option," 1999
- RFC 3168, "The Addition of Explicit Congestion Notification (ECN) to IP," 2001

- RFC 3175, "Aggregation of RSVP for IPv4 and IPv6 Reservations," 2001
- RFC 3514, "The Security Flag in the IPv4 Header," April 1, 2003
- RFC 4301, "Security Architecture for the Internet Protocol," 2005
- RFC 4302, "IP Authentication Header," 2005
- RFC 4303, "IP Encapsulating Security Payload (ESP)," 2005
- RFC 4305, "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)," 2005
- RFC 4727, "Experimental Values in IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers," 2006
- RFC 5095, "Deprecation of Type 0 Routing Headers in IPv6," 2007
- RFC 5350, "IANA Considerations for the IPv4 and IPv6 Router Alert Options," 2008
- RFC 5722, "Handling of Overlapping IPv6 Fragments," 2009
- RFC 5871, "IANA Allocation Guidelines for the IPv6 Routing Header," 2010
- RFC 6105, "IPv6 Router Advertisement Guard," 2011
- RFC 6275, "Mobility Support in IPv6," 2011
- RFC 6398, "IP Router Alert Considerations and Usage," 2011
- RFC 6434, "IPv6 Node Requirements," 2011
- RFC 6437, "IPv6 Flow Label Specification," 2011
- RFC 6553, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams," 2012
- RFC 6554, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)," 2012
- RFC 6564, "A Uniform Format for IPv6 Extension Headers," 2012
- RFC 6621, "Simplified Multicast Forwarding," 2012
- RFC 6946, "Processing of IPv6 'Atomic' Fragments," 2013
- RFC 6980, "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery," 2013
- RFC 7045, "Transmission and Processing of IPv6 Extension Headers," 2014
- RFC 7112, "Implications of Oversized IPv6 Header Chains," 2014
- RFC 7113, "Implementation Advice for IPv6 Router Advertisement Guard (RA Guard)," 2014
- RFC 7136, "Significance of IPv6 Interface Identifiers," 2014

Drafts

Drafts can be found at *http://www.ietf.org/ID.html*. To locate the latest version of a draft, refer to *https://datatracker.ietf.org/public/pidtracker.cgi*. You can enter the draft name without a version number and the most current version will come up. If a draft does not show up, it was possibly deleted. If it was published as an RFC, the RFC number will be displayed. *http://tools.ietf.org/wg* is also a very useful site. More information on the process of standardization, RFCs, and drafts can be found in Appendix A.

Here's a list of drafts I refer to in this chapter, as well as interesting drafts that relate to the topics in this chapter:

"Segment Routing Architecture"
draft-filsfils-spring-segment-routing-02

"Segment Routing Use Cases"
draft-filsfils-spring-segment-routing-use-cases-00

"IPv6 Segment Routing Header (SRH)"
draft-previdi-6man-segment-routing-header-00

CHAPTER 4
ICMPv6

If you are familiar with IPv4, the Internet Control Message Protocol (ICMP) for IPv4 is probably a good friend of yours: it gives important information about the health of the network. ICMPv6 is the version that works with IPv6. It reports errors if packets cannot be processed properly and sends informational messages about the status of the network. For example, if a router cannot forward a packet because it is too large to be sent out on another network, it sends an ICMP message back to the originating host. The source host can use this ICMP message to determine a better packet size and then resend the data. ICMP also performs diagnostic functions, such as the well-known ping, which uses ICMP Echo Request and Echo Reply messages to test availability of a node.

ICMPv6 is much more powerful than ICMPv4 and contains new functionality, as described in this chapter. For instance, the Internet Group Management Protocol (IGMP) function that manages multicast group memberships with IPv4 has been incorporated into ICMPv6. The same is true for ARP/RARP, the Address Resolution Protocol/Reverse Address Resolution Protocol function used in IPv4 to map Layer 2 addresses to IP addresses (and vice versa). Neighbor Discovery (ND) is introduced; it uses ICMPv6 messages to determine link-layer addresses for neighbors attached to the same link, find routers, keep track of which neighbors are reachable, and detect changed link-layer addresses. New message types have been defined to allow for simpler renumbering of networks and updating of address information between hosts and routers. ICMPv6 also supports Mobile IPv6, which is described in Chapter 8. ICMPv6 is part of IPv6, and it must be implemented fully by every IPv6 node. The protocol is defined in RFC 4443. Neighbor Discovery is defined in RFC 4861.

General Message Format

All ICMPv6 messages have the same general header structure, as shown in Figure 4-1.

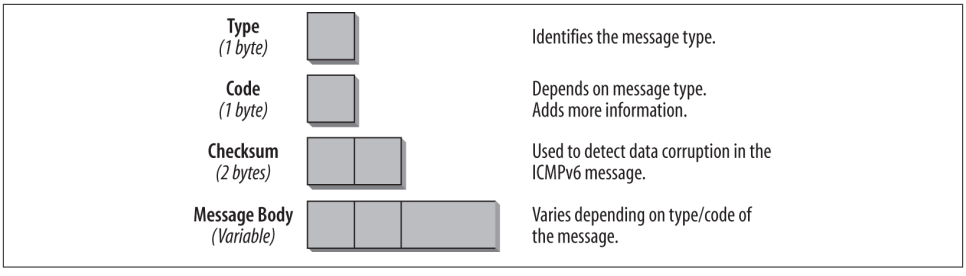

Figure 4-1. General ICMPv6 header format

Type (1 byte)
: This field specifies the type of message, which determines the format of the remainder of the message. Tables 4-1 and 4-2 list ICMPv6 message types and numbers.

Code (1 byte)
: The Code field depends on the message type and allows for more granular information in certain cases. Refer to Tables 4-1 and 4-2 for a detailed list.

Checksum (2 bytes)
: The Checksum field is used to detect data corruption in the ICMPv6 header and in parts of the IPv6 header. In order to calculate the checksum, a node must determine the Source and Destination address in the IPv6 header. There is a pseudoheader included in the checksum calculation, which is new with ICMPv6. Chapter 5 discusses the checksum and the pseudoheader.

Message body (variable size)
: Depending on the type and code, the message body will hold different data. In the case of an error message, to assist in troubleshooting, it will contain as much as possible of the packet that invoked the message. The total size of the ICMPv6 packet should not exceed the minimum IPv6 MTU, which is 1,280 bytes. Tables 4-1 and 4-2 provide an overview of the different message types, along with the additional code information, which depends on the message type.

There are two classes of ICMP messages:

ICMP error messages
: Error messages have a 0 in the high-order bit of their message Type field. ICMP error message types are, therefore, in the range from 0 to 127.

ICMP informational messages
: Informational messages have a 1 in the high-order bit of their message Type field. ICMP informational message types are, therefore, in the range from 128 to 255.

An IPv6 header and zero or more Extension headers precede every ICMPv6 message. The header just preceding the ICMP header has a Next Header value of 58. This value is different from the value for ICMPv4 (which has the value 1).

The values for the Next Header field are discussed in Chapter 3.

The following message types are described in RFC 4443:

ICMPv6 error messages

- Destination Unreachable (message type 1)
- Packet Too Big (message type 2)
- Time Exceeded (message type 3)
- Parameter Problem (message type 4)

ICMPv6 informational messages

- Echo Request (message type 128)
- Echo Reply (message type 129)

For the most current list of ICMPv6 message types, refer to the Internet Assigned Number Authority (IANA) at *http://www.iana.org/assignments/icmpv6-parameters*. All IPv4 ICMP parameters can be found at *http://www.iana.org/assignments/icmp-parameters*.

Table 4-1. ICMPv6 error messages and code type

Message number	Message type	Code field
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 7 = error in source routing header
2	Packet Too Big	Code field set to 0 by the sender and ignored by the receiver

Message number	Message type	Code field
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 The pointer field identifies the octet offset within the invoking packet where the error was detected. The pointer points beyond the end of the ICMPv6 packet if the field in error is beyond what can fit in the maximum size of an ICMPv6 error message.
100 and 101	Private experimentation	RFC 4443
127	Reserved for expansion of ICMPv6 error messages	RFC 4443

Note that the message numbers and types have substantially changed compared to ICMPv4. ICMP for IPv6 is a different protocol, and the two versions of ICMP are not compatible. Your analyzer, such as Wireshark, should properly decode all this information, so you do not have to worry about memorizing it.

Table 4-2. ICMPv6 informational messages

Message number	Message type	Description
128	Echo Request	RFC 4443. Used for the ping command.
129	Echo Reply	
130	Multicast Listener Query	RFC 2710. Used for multicast goup management.
131	Multicast Listener Report	
132	Multicast Listener Done	
133	Router Solicitation	RFC 4861. Used for Neighbor Discovery and Autoconfiguration.
134	Router Advertisement	
135	Neighbor Solicitation	
136	Neighbor Advertisement	
137	Redirect Message	
138	Router Renumbering	RFC 2894 Codes: 0 = Router renumbering command 1 = Router renumbering result 255 = Sequence number reset
139	ICMP Node Information Query	RFC 4620
140	ICMP Node Information Response	
141	Inverse ND Solicitation	RFC 3122

Message number	Message type	Description
142	Inverse ND Adv Message	RFC 3122
143	Version 2 Multicast Listener Report	RFC 3810
144	ICMP Home Agent Address Discovery Request Message	RFC 6275, "ICMPv6 Messages for Mobile IPv6"
145	ICMP Home Agent Address Discovery Reply Message	
146	ICMP Mobile Prefix Solicitation Message	
147	ICMP Mobile Prefix Advertisement Message	
148	Certification Path Solicitation Message	RFC 3971 "ICMPv6 Messages for SEcure Neighbor Discovery"
149	Certification Path Advertisement Message	
151	Multicast Router Advertisement	RFC 4286
152	Multicast Router Solicitation	
153	Multicast Router Termination	
154	Fast Mobile IPv6 Messages	RFC 5568
155	Routing Protocol for Low-Power Network Messages	RFC 6550
156	ILNPv6 Locator Update Message	RFC 6743
157	Duplicate Address Request	RFC 6775 (6LoWPANs)
158	Duplicate Address Confirmation	
200 and 201	Private experimentation	RFC 4443
255	Reserved for expansion of ICMPv6 informational messages	RFC 4443

With the exception of the router renumbering message (138), the ICMPv6 informational messages do not use the Code field. It is, therefore, set to zero.

As you will learn in the coming paragraphs, ICMPv6 is very powerful and is used for many processes that are critical to the proper working of IPv6, such as Path MTU Discovery. Therefore it is not a good idea to completely filter all ICMP messages at firewalls, as has been the practice in many IPv4 networks. With ICMPv6 you have to carefully evaluate which ICMPv6 messages are important. RFC 4890, "Recommendations for Filtering ICMPv6 Messages in Firewalls," discusses this and makes recommendations.

ICMP Error Messages

Every ICMP message can have a slightly different header depending on the kind of error report or information it carries. The following sections outline the structure of each type of ICMPv6 message.

Destination Unreachable

A *Destination Unreachable* message is generated if an IP datagram cannot be delivered. A Type field with the value 1 identifies this message. The ICMP message is sent to the Source address of the invoking packet. The format of the Destination Unreachable message is shown in Figure 4-2.

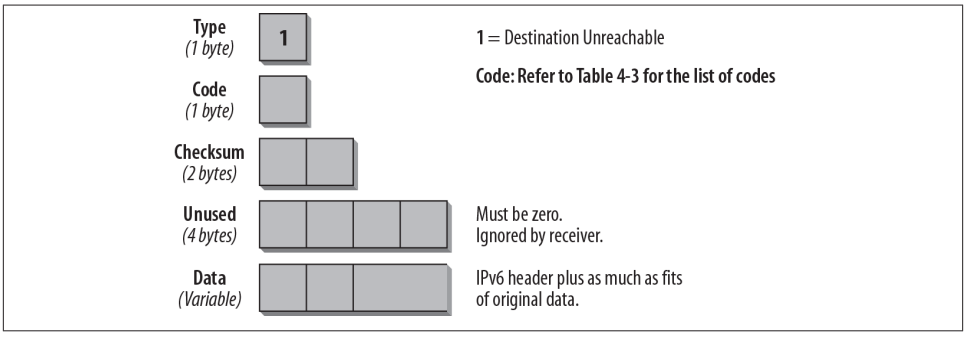

Figure 4-2. Format of the Destination Unreachable message

The Type field is set to 1, which is the value for the Destination Unreachable message. The Code field supplies more information about the reason why the datagram was not delivered. The possible codes are listed in Table 4-3. The data portion of the ICMP message contains as much of the original message as will fit into the ICMP message.

Table 4-3. Code values of the Destination Unreachable message (type 1)

Code	Description
0	"No route to destination." This code is used if a router cannot forward a packet because it does not have a route in its table for a destination network. This can happen only if the router does not have an entry for a default route.
1	"Communication with destination administratively prohibited." This type of message can, for example, be sent by a firewall that cannot forward a packet to a host inside the firewall because of a packet filter. It might also be sent if a node is configured not to accept unauthenticated Echo Requests.
2	"Beyond scope of Source address." This code is used if the Destination address is beyond the scope of the Source address, e.g., if a packet has a link-local Source address and a global Destination address.
3	"Address unreachable." This code is used if a Destination address cannot be resolved into a corresponding network address or if there is a data-link layer problem preventing the node from reaching the destination network.
4	"Port unreachable." This code is used if the transport protocol (e.g., UDP) has no listener and there is no other means to inform the sender. For example, if a Domain Name System (DNS) query is sent to a host and the DNS server is not running, this type of message is generated.

Code	Description
5	"Source address failed ingress/egress policy." This code is used if a packet with this Source address is not allowed due to ingress or egress filtering policies.
6	"Reject route to destination." This code is used if the route to the destination is a reject route.

If the destination is unreachable due to congestion, no ICMP message is generated. A host that receives a Destination Unreachable message must inform the upper-layer process.

Packet Too Big

If a router cannot forward a packet because it is larger than the MTU of the outgoing link, it will generate a Packet Too Big message (shown in Figure 4-3). This ICMPv6 message type is used as part of the Path MTU Discovery process that I discuss later in this chapter. The ICMP message is sent to the Source address of the invoking packet.

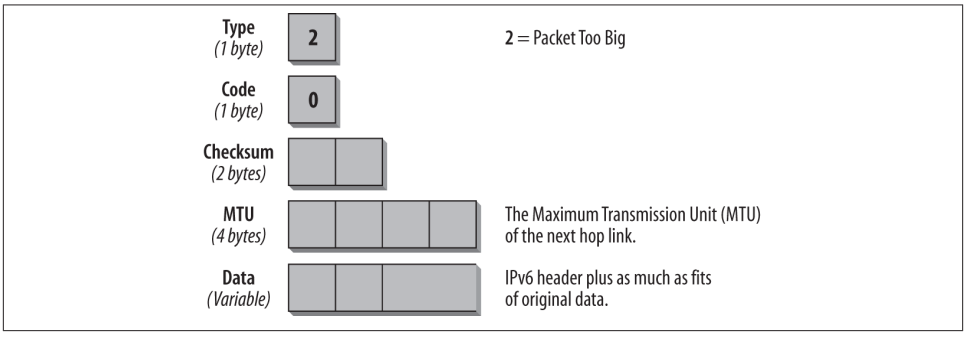

Figure 4-3. Format of the Packet Too Big message

The Type field has the value 2, which identifies the Packet Too Big message. In this case, the Code field is not used and is set to 0. The important information for this type of message is the MTU field, which contains the MTU size of the next hop link.

RFC 4443 states that an ICMPv6 message should not be generated as a response to a packet with an IPv6 multicast Destination address, a link-layer multicast address, or a link-layer broadcast address. The Packet Too Big message is an exception to this rule. Because the ICMP message contains the supported MTU of the next hop link, the source host can determine the MTU that it should use for further communication. A host that receives a Packet Too Big message must inform the upper-layer process.

Time Exceeded

When a router forwards a packet, it always decrements the hop limit by one. The hop limit makes sure that a packet does not endlessly travel through a network. If a router receives a packet with a hop limit of 1 and decrements the limit to 0, it discards the packet, generates a Time Exceeded message with a code value of 0, and sends this message back to the source host. This error can indicate a routing loop or the fact that the sender's initial hop limit is too low. It can also tell you that someone used the *traceroute* utility, which is described later in this chapter. Figure 4-4 shows the format of the Time Exceeded message.

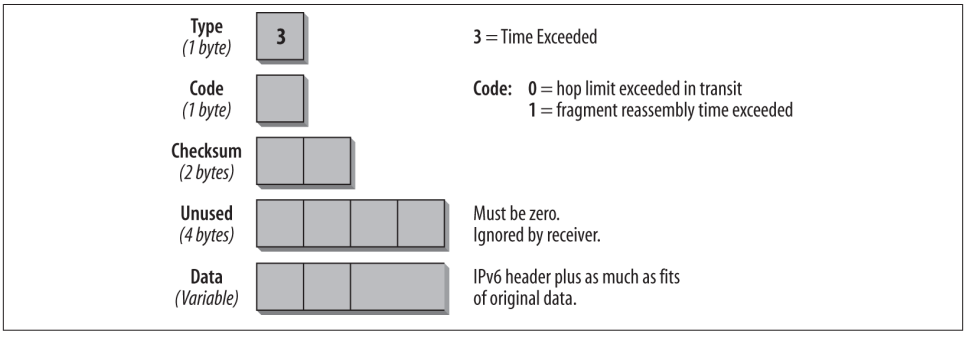

Figure 4-4. Format of the Time Exceeded message

The Type field carries the value 3, specifying the Time Exceeded message. The Code field can be set to 0, which means the hop limit was exceeded in transit, or to 1, which means that the fragment reassembly time is exceeded. The data portion of the ICMP message contains as much of the original message as will fit into the ICMP message, depending on the MTU used.

An incoming Time Exceeded message must be passed to the upper-layer process. Table 4-4 shows the Code fields for the Time Exceeded message.

Table 4-4. Code values for Time Exceeded message (type 3)

Code	Description
0	"Hop limit exceeded in transit." Possible causes: the initial hop limit value is too low; there are routing loops; or use of the traceroute utility.
1	"Fragment reassembly time exceeded." If a fragmented packet is sent by using a fragment header (refer to Chapter 2 for more details) and the receiving host cannot reassemble all packets within a certain time, it notifies the sender by issuing this ICMP message.

The "Hop limit exceeded in transit" message type is commonly used to do the traceroute function. *Traceroute* is helpful in determining the path that a packet takes when traveling through the network. In order to do this, a first packet is sent out with a hop limit of 1.

The first router in the path decrements the hop limit to 0, discards the packet, and sends back an ICMP message type 3, code 0. The source host now knows the address of the first hop router. Next, it sends out a second packet with a hop limit of 2. This packet is forwarded by the first router, which decrements the hop limit to 1. The second router in the path decrements the hop limit to 0, discards the packet, and sends back an ICMP message type 3, code 0. Now the source knows about the second router in the path. Raising the hop limit by one (with every packet sent until the packet reaches the final destination) continues this process. Every router in the path to the final destination sends an ICMP message back to the source host, thereby providing its IP address. It is important to know that if there are redundant paths to the destination, *traceroute* does not necessarily show the same route for all tests because it might choose different paths.

Parameter Problem

If an IPv6 node cannot complete the processing of a packet because it has a problem identifying a field in the IPv6 header or in an Extension header, it must discard the packet, and it should send an ICMP Parameter Problem message back to the source of the problem packet. This type of message is often used when an error that does not fit into any of the other categories is encountered. The format of this ICMP message is shown in Figure 4-5.

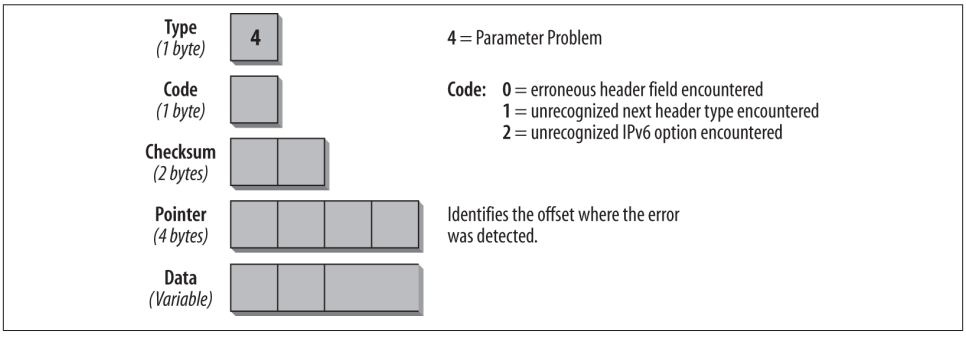

Figure 4-5. Format of the Parameter Problem message

The Type field has the value 4, which specifies the Parameter Problem message. The Code field can contain any of the three values described in Table 4-5. The Pointer field identifies at which byte in the original packet the error was detected. The ICMP message includes as much of the original data as fits, up to the minimum IPv6 MTU. It is possible that the pointer points beyond the ICMPv6 message. This would be the case if the field in error was beyond what can fit in the maximum size of an ICMPv6 error message.

Table 4-5 shows the Code fields for the Parameter Problem message.

Table 4-5. Code values for Parameter Problem (type 4)

Code	Description
0	Erroneous header field encountered
1	Unrecognized next header type encountered
2	Unrecognized IPv6 option encountered

For example, an ICMPv6 message of type 4 with a code value of 1 and a pointer set to 40 indicates that the Next Header type in the header following the IPv6 header was unrecognized.

An incoming Parameter Problem message must be passed to the upper-layer process.

ICMP Informational Messages

In RFC 4443, two types of informational messages are defined: the Echo Request and the Echo Reply messages. Other ICMP informational messages are used for Path MTU Discovery and Neighbor Discovery. These messages are discussed at the end of this chapter and defined in RFC 4861, "Neighbor Discovery for IP Version 6," and RFC 1981, "'Path MTU Discovery' for IP Version 6."

The Echo Request and Echo Reply messages are used for one of the most common TCP/IP utilities: Packet InterNet Groper (*ping*). Ping is used to determine whether a specified host is available on the network and ready to communicate. The source host issues an Echo Request message to the specified destination. The destination host, if available, responds with an Echo Reply message. Figures 4-8 and 4-9 (later in the chapter) show what a ping looks like in the trace file.

Echo Request Message

The format of the Echo Request message is shown in Figure 4-6.

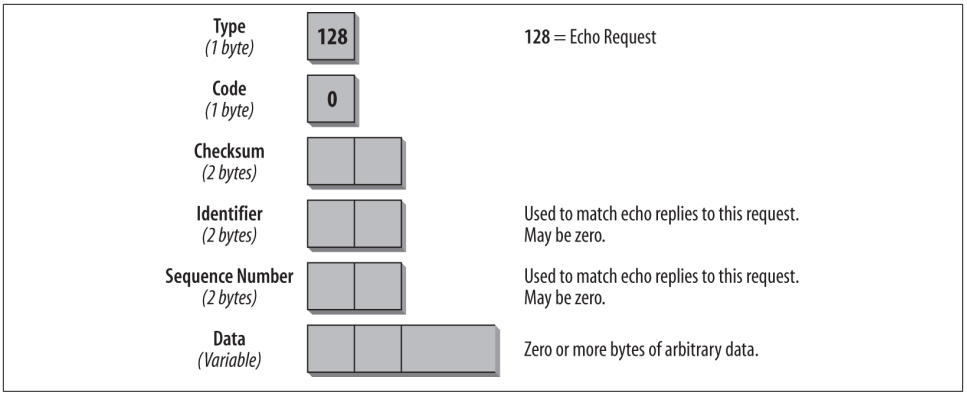

Figure 4-6. Format of the Echo Request message

The Type field is set to 128, the value for the Echo Request. The Code Field is not used for this message and is therefore set to 0. The Identifier and Sequence Number fields are used to match requests with replies. The reply must always contain the same numbers as the request. Whether an identifier and a sequence number are used and what kind of arbitrary data is included in the Echo Request depends on the TCP/IP stack you are using. When you analyze trace files with Echo Request and Echo Reply messages and you are familiar with some stacks, you can determine the TCP/IP stack of the sender by looking at the arbitrary data.

Echo Reply

The format of the Echo Reply message is very similar to that of the Echo Request, as shown in Figure 4-7.

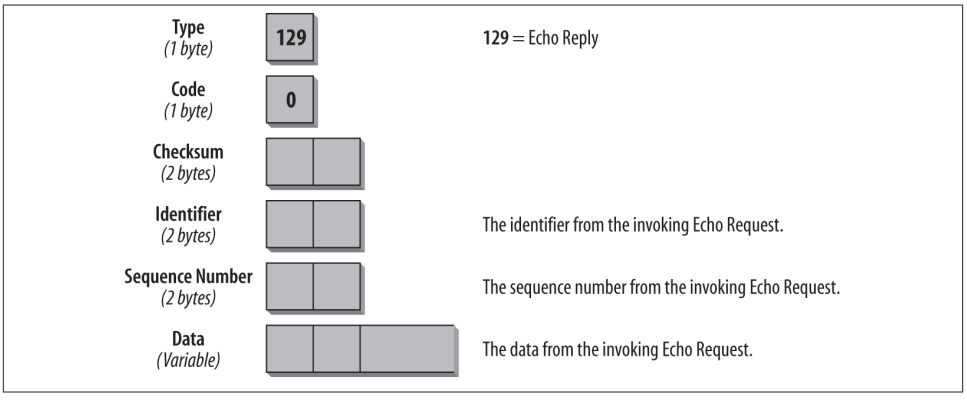

Figure 4-7. Format of the Echo Reply message

The Type field contains the value 129 for Echo Reply. The Code field is unused and set to 0. The Identifier and Sequence Number fields must match the fields in the request. The data of the Echo Request message must be copied into the reply entirely and unmodified. If an upper-layer process initiated the Echo Request, the reply must be passed to that process. If the Echo Request message was sent to a unicast address, the Source address of the Echo Reply message must be the same as the Destination address of the Echo Request message. If the Echo Request was sent to an IPv6 multicast address, the Source address of the Echo Reply must be a unicast address of the interface on which the multicast Echo Request was received.

According to the specification, ICMPv6 Echo Request and Reply messages can be authenticated, using an IPv6 authentication header. This means that a node can be configured to ignore nonauthenticated ICMPv6 pings and provide protection against different ICMPv6 attacks. I don't know of any implementation, though, that supports this feature.

Processing Rules

There are several rules that govern processing of ICMP packets. They can be found in RFC 4443 and are summarized as follows:

- If a node receives an ICMPv6 error message of unknown type, it must pass it to the upper layer.
- If a node receives an ICMPv6 informational message of unknown type, it must be silently discarded.
- As much as possible of the packet that caused the ICMP error message will be included in the ICMP message body. The ICMP packet should not exceed the minimum IPv6 MTU.
- If the error message has to be passed to the upper-layer protocol, the protocol type is determined by extracting it from the original packet (present in the body of the ICMPv6 error message). In case the protocol type cannot be found in the body of the ICMPv6 message (because there were too many Extension headers present in the original packet, and the part of the header that contained the upper-layer protocol type was truncated), the ICMPv6 message is silently discarded.

An ICMPv6 error message must not be sent in the following cases:

- As a result of an ICMPv6 error message.
- As a result of an ICMPv6 redirect message.
- As a result of a packet sent to an IPv6 multicast address. There are two exceptions to this rule: the Packet Too Big message that is used for Path MTU Discovery, and the Parameter Problem with the code value 2 for an unrecognized IPv6 option.

- As a result of a packet sent as a link-layer multicast (exceptions just described apply).
- As a result of a packet sent as a link-layer broadcast (exceptions just described apply).
- As a result of a packet whose Source address does not uniquely identify a single node. This could be an IPv6 unspecified address, an IPv6 multicast address, or an IPv6 address known to be an anycast address.

Every IPv6 node must implement a rate-limiting function that limits the rate of ICMPv6 messages it sends, and it should be configurable. If this function is implemented properly, it protects against Denial of Service attacks.

The ICMPv6 Header in a Trace File

After reading through all that dry information, you deserve something different. The following screenshot (Figure 4-8) shows what a ping looks like in the trace file and provides details of many of the fields discussed so far.

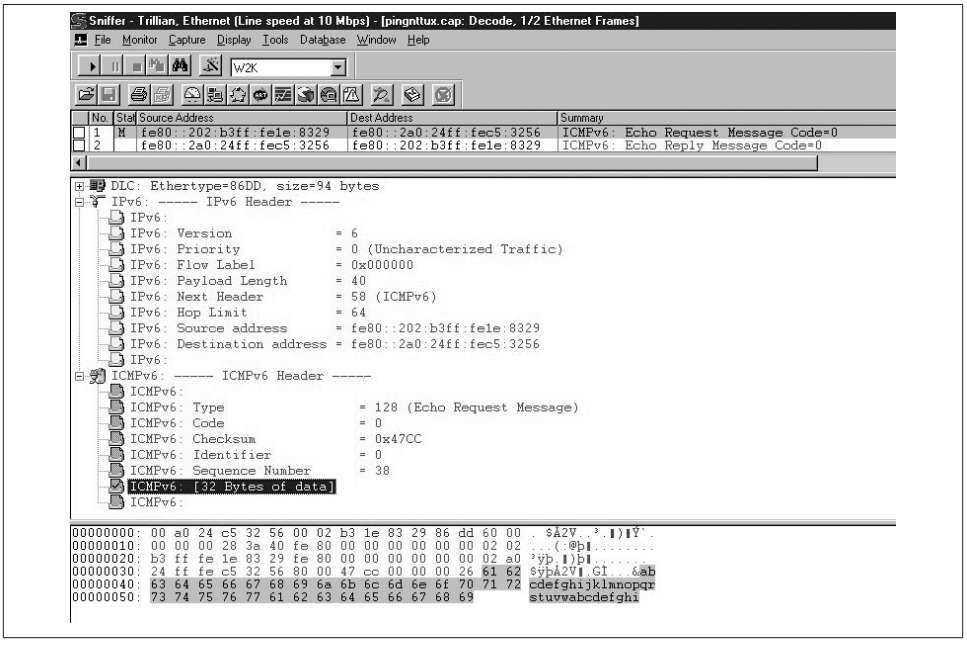

Figure 4-8. Echo Request in a trace file

As shown in Chapter 3, the IPv6 header provides the following information: the Version field is set to 6 and indicates that this is an IPv6 packet. Priority and Flow Label are not configured and set to zero. The Payload Length field is set to 40 bytes. The Next Header

field has the value 58, which is the value for ICMPv6. The Hop Limit is set to 64. We can also see source and destination IP address. The prefix `fe80:` indicates that these two addresses are link-local addresses.

Note the first three fields of the ICMPv6 header. They are the fields that are common for every ICMPv6 message: the Type, Code, and Checksum fields. The Type field contains the value 128, which is the value for an Echo Request. The Identifier and Sequence Number fields are unique to the Echo Request and Echo Reply message. The Identifier is not used in this case, and the sender has set the sequence to 38. It has to be identical in the matching reply shown in the following screenshot. The Data field contains arbitrary data that doesn't need to make sense to anyone.

Oh, I almost forgot that earlier I promised to show vendor stack-related data in the Echo Request message. What you see here—the alphabet up to the letter "w"—is what Microsoft uses. Whenever you see this in a trace file, a Microsoft stack is sending the request. Figure 4-9 shows the Echo Reply in detail.

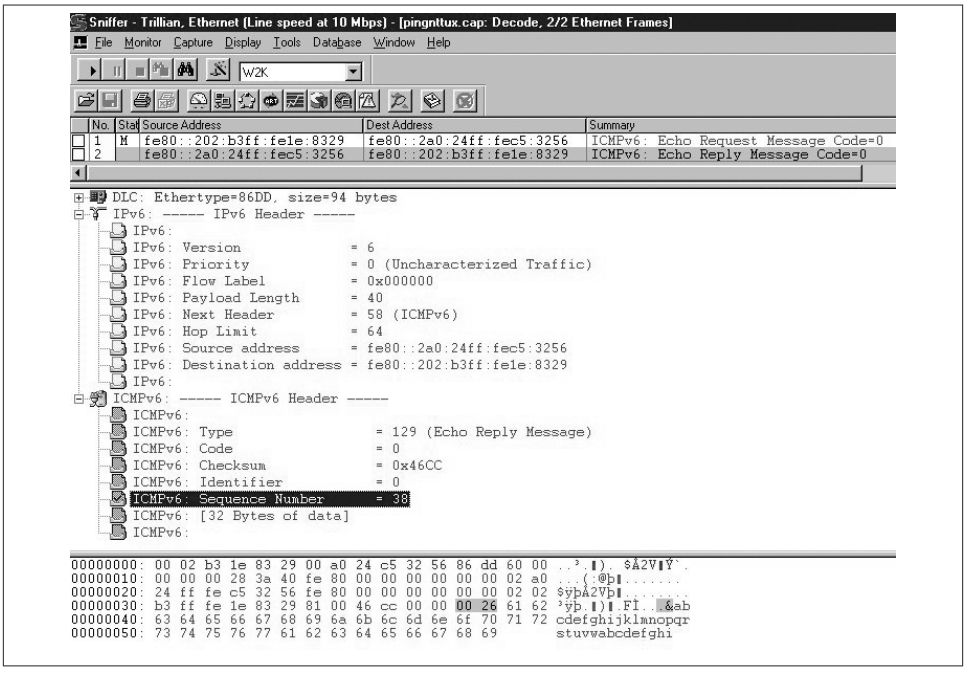

Figure 4-9. Echo Reply in a trace file

Again, the IPv6 header shows a value of 6 for the IP version and a Next Header value of 58 for ICMPv6. The Destination address of the previous frame is now the Source address, and the previous Source address is now the Destination address. The Type field in the ICMPv6 header shows a value of 129, which is the value for an Echo Reply. The

Identifier and Sequence Number fields, as well as the Data field, match the ones in the Echo Request.

Neighbor Discovery

Neighbor Discovery (ND) is specified in RFC 4861. The specifications in this RFC relate to different protocols and processes known from IPv4 that have been modified and improved. New functionality has also been added. It combines Address Resolution Protocol (ARP) and ICMP Router Discovery and Redirect. With IPv4, we have no means to detect whether a neighbor is reachable. With the Neighbor Discovery protocol, a Neighbor Unreachability Detection (NUD) mechanism has been defined. Duplicate IP address detection (DAD) has also been implemented. IPv6 nodes use Neighbor Discovery for the following purposes:

- For Stateless Autoconfiguration of IPv6 addresses
- To determine network prefixes, routes, and other configuration information
- For Duplicate IP Address Detection (DAD)
- To determine Layer 2 addresses of nodes on the same link
- To find neighboring routers that can forward their packets
- To keep track of which neighbors are reachable and which are not (NUD)
- To detect changed link-layer addresses

The following improvements over the IPv4 set of protocols can be noted:

- Router Discovery is now part of the base protocol set. With IPv4, the mechanism needs to get the information from the routing table or DHCP.
- Router Advertisement packets contain link-layer addresses for the router. There is no need for the node receiving a Router Advertisement to send out an additional ARP request (as an IPv4 node would have to do) to get the link-layer address for the router interface. The same is true for ICMPv6 Redirect messages; they contain the link-layer address of the new next-hop router interface.
- Router Advertisement packets contain the prefix for a link (subnet information). There is no longer a need to configure subnet masks; they can be learned from the Router Advertisement.
- Neighbor Discovery (ND) provides mechanisms to renumber networks more easily. New prefixes and addresses can be introduced while the old ones are still in use, and the old ones can be deprecated and removed gradually.
- Router Advertisements enable Stateless Address Autoconfiguration and can tell hosts when to use Stateful Address Configuration (e.g., DHCP).

- Routers can advertise an MTU to be used on a link.
- Multiple prefixes can be assigned to one link. By default, hosts learn all prefixes from the router, but the router can be configured not to advertise some or all of the prefixes. In that case, hosts assume that a nonadvertised prefix destination is remote and send the packets to the router. The router can then issue ICMP Redirect messages as needed.
- Neighbor Unreachability Detection (NUD) is part of the base protocol. It substantially improves packet delivery in case of failed routers or link interfaces that changed their link-layer address. It solves the issues with outdated ARP caches. NUD detects failed connectivity, and traffic is not sent to unreachable neighbors. The Neighbor Unreachability Detection also detects failed routers and switches to live ones.
- Router Advertisements and ICMP redirects use link-local addresses to identify routers. This allows hosts to maintain their router associations even in the case of renumbering or use of new global prefixes.
- Neighbor Discovery messages have a hop limit value of 255, and requests with a lower hop limit are not answered. This makes Neighbor Discovery immune to remote hosts that try to sneak into your link, because their packets have a decremented hop limit and are thus ignored.
- Standard IP authentication and security mechanisms can be applied to Neighbor Discovery.

This summary gives an idea of what can be expected from this part of the specification. Now let's discuss the different processes in detail. The Neighbor Discovery protocol consists of five ICMP messages: a pair of Router Solicitation/Router Advertisement messages, a pair of Neighbor Solicitation/Neighbor Advertisement messages, and an ICMP Redirect message (refer to Table 4-2 earlier in this chapter for a summary of ICMP informational message types).

To summarize, the Neighbor Discovery Protocol (NDP) specification is used by both hosts and routers. Its functions include Neighbor Discovery (ND), Router Discovery (RD), Stateless Address Autoconfiguration (SLAAC), Address Resolution, Neighbor Unreachability Detection (NUD), Duplicate Address Detection (DAD), and Redirection.

There are operational issues with Neighbor Discovery, which can result in vulnerabilities when a network is scanned. RFC 6583, "Operational Neighbor Discovery Problems," describes these issues and presents possible mitigation techniques.

Router Solicitation and Router Advertisement

Routers send out Router Advertisement messages at regular intervals. Hosts can request Router Advertisements by issuing a Router Solicitation message. This will trigger routers to issue Router Advertisements immediately, outside of the regular interval. The format is shown in Figure 4-10.

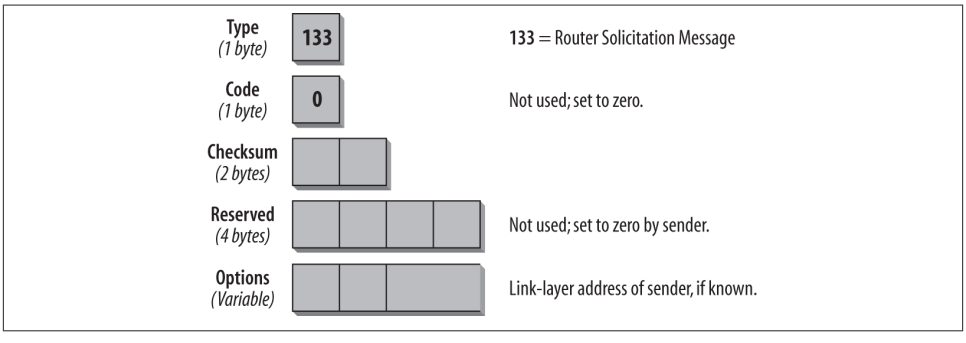

Figure 4-10. Router Solicitation message

In the IP header of a Router Solicitation message, you will usually see the all-routers multicast address of `ff02::2` as a Destination address. The hop limit is set to 255. The ICMP Type field is set to 133, which is the value for the Router Solicitation message. The Code field is unused and set to 0. The following two bytes are used for the Checksum. The next four bytes are unused and reserved for future use. The sender sets them to 0, and the receiver ignores those fields. For a Router Solicitation message, a valid option is the link-layer address of the sending host, if the address of the sending host is known. If the Source address on the IP layer is the unspecified (all-zeros) address, this field is not used. More options may be defined in future versions of ND. If a host cannot recognize an option, it should ignore the option and process the packet.

Routers that receive this Solicitation message reply with a Router Advertisement message. Routers also issue those messages periodically. The format of the Router Advertisement message is shown in Figure 4-11.

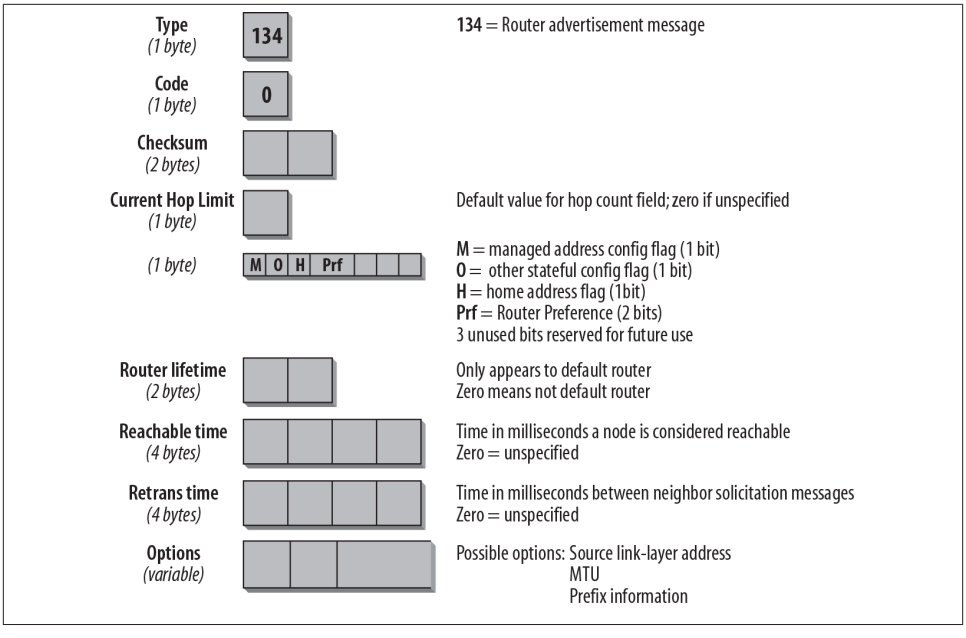

Figure 4-11. Router Advertisement message

By inspecting the IP header of the Router Advertisement message, you can determine whether this Router Advertisement is periodic or was sent in reply to a Solicitation message. A periodic advertisement's Destination address will be the all-nodes multicast address `ff02::1`. A solicited advertisement's Destination address will be the address of the interface that originated the solicitation message. Again, the hop limit is set to 255.

The ICMP Type field is set to 134, the value for a Router Advertisement message; the Code field is unused and set to 0. The Current Hop Limit field can be used to configure all nodes on a link for a default hop limit. The value entered in this field will be used as a default hop limit value in outgoing packets by all nodes on the link. A value of 0 in this field means that this option is unspecified by this router—in which case the default hop limit values of the source hosts are used.

The next 1-bit field, the M-Flag, specifies whether *Stateful Configuration* is to be used. Stateful Configuration refers to DHCPv6. If this bit is 0, the nodes on this link use Stateless Address Autoconfiguration (SLAAC). If the bit is set to 1, it specifies Stateful Autoconfiguration (DHCPv6). The O-Flag configures whether nodes on this link use DHCPv6 for other than IP address information. A value of 1 means the nodes on this link use DHCPv6 for nonaddress-related information. In this case a DHCPv6 client will send out a DHCPv6 information request message to obtain additional configuration options. The Mobile IPv6 specification (RFC 6275) defines the third bit, the Home

Address flag (H-Flag). When a router sets the H-Flag to 1, it means that it is a home agent for this link.

Note that the DHCPv6 RFC leaves room for interpretation. It is possible that different operating systems have slightly different behavior in response to these two flags. There is a draft called "DHCPv6/SLAAC Address Configuration Interaction Problem Statement," which describes these issues and discusses the results found with testing several desktop operating systems in a lab.

For a discussion of Mobile IPv6 and Home Agent, refer to Chapter 8.

The next two bits are used by an optional extension to the Router Advertisement message, which allows routers to advertise preferences and more specific routes. This makes it possible for hosts to choose the best router in situations where they receive more than one router advertisement. It is also important for multihomed routers, which will be an increasingly important scenario in IPv6 networks. This extension uses the two bits after the H-Flag in the router advertisement as a *Preference flag* and defines a *Route Information option*. It is specified in RFC 4191. The remaining three bits of this byte are reserved for future use and must be 0.

The Router Lifetime field is important only if this router is to be used as a default router by the nodes on the link. A value of 0 indicates that this router is not a default router and will therefore not appear on the default router list of receiving nodes. Any other value in this field specifies the lifetime, in seconds, associated with this router as a default router. The maximum value is 18.2 hours.

The Reachable Time field indicates the time in which a host assumes that neighbors are reachable after having received a reachability confirmation. A value of 0 means that it is unspecified. The Neighbor Unreachability Detection algorithm uses this field.

The Retrans Timer field is used by the address resolution and Neighbor Unreachability Detection mechanisms; it states the time in milliseconds between retransmitted Neighbor Solicitation messages. A value of 0 indicates that this router is not configured with a retransmission timer.

For the Options field, there are currently three possible values:

- Source link-layer address.
- MTU size to be used on links with variable MTU sizes (Token Ring, for example).

- Prefix information, important for Stateless Address Autoconfiguration. The router inserts all its prefixes for the link that the nodes on the link need to know.

More options may be defined in future versions of ND. A trace file later in this chapter shows what the options look like.

Router Advertisement spoofing is a security concern also known as rogue RA. The problem and mitigation techniques are discussed in the section "Neighbor Discovery issues" on page 204 in Chapter 6 on Security.

Neighbor Solicitation and Neighbor Advertisement

This pair of messages fulfills two functions: the link-layer address resolution that is handled by ARP in IPv4, and the Neighbor Unreachability Detection mechanism. If the Destination address is a multicast address (usually the solicited node multicast address), the source is resolving a link-layer address. If the source is verifying the reachability of a neighbor, the Destination address is a unicast address. This message type is also used for Duplicate IP Address Detection (DAD).

The format of the Neighbor Solicitation message is shown in Figure 4-12.

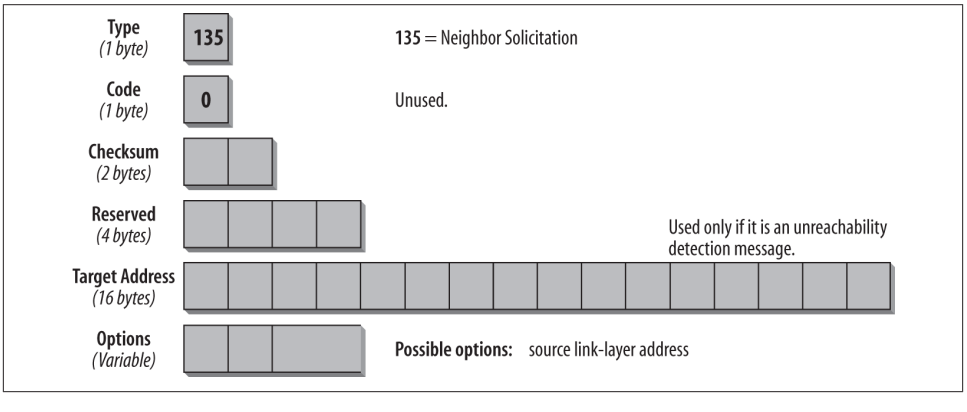

Figure 4-12. Format of the Neighbor Solicitation message

In the IP header of this message type, the Source address can be either the interface address of the originating host or, in the case of SLAAC and DAD, the unspecified (all-zeros) address. The hop limit is set to 255. The Type field in the ICMP header is set to 135, and the Code field is unused and set to 0. After the two checksum bytes, four unused bytes are reserved and must be set to 0. The target address is used in Neighbor Advertisement and Redirect messages. It must not be a multicast address.

The Options field can contain the link-layer Source address, but only if it is not sent from the all-zeros address. During Stateless Address Autoconfiguration, in a message that uses the unspecified address as a Source address, the Options field is set to 0. The link-layer option must be used in multicast solicitations (link-layer address detection) and can be used in unicast solicitations (Unreachability Detection).

Neighbor Advertisement messages are sent as a reply to Neighbor Solicitation messages or to propagate new information quickly. The format of the message is shown in Figure 4-13.

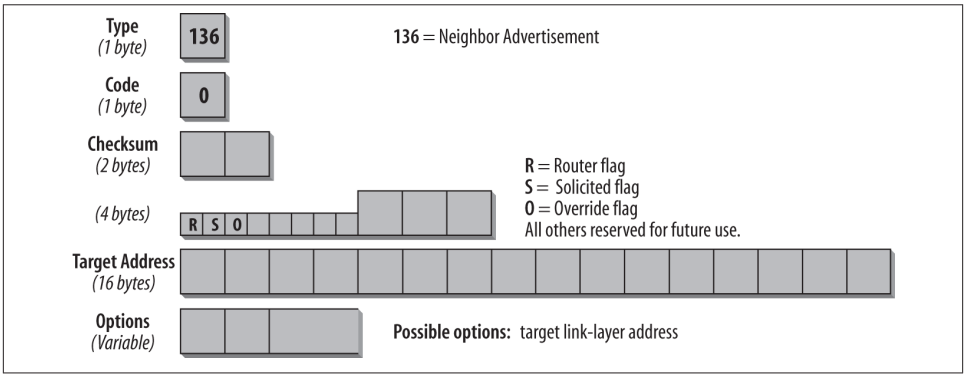

Figure 4-13. Format of the Neighbor Advertisement message

The type of address in the IP header indicates whether the message is the answer to a solicitation or an unsolicited message. In the case of a solicited advertisement, the destination IP address is the Source address of the interface that sent the solicitation. If the message is the reply to a DAD message that originated from an unspecified Source address, the reply will go to the all-nodes multicast address of `ff02::1`. The same is true for all unsolicited periodic advertisements.

The Type field in the ICMP header is set to 136, the value for Neighbor Advertisement messages. The Code field is unused and set to 0. When the Router flag is set, the sender is a router.

When the Solicited flag is set, the message is sent in response to a Neighbor Solicitation. For instance, if a host confirms its reachability in answer to an unreachability detection message, the S-Bit is set. The S-Bit is not set in multicast advertisements. The Override flag indicates that the information in the Advertisement message should override existing Neighbor Cache entries and update any cached link-layer addresses. If the O-Bit is not set, the advertisement will not update a cached link-layer address, but it will update an existing Neighbor Cache entry for which no link-layer address exists. The O-Bit should not be set in an advertisement for an anycast address. I discuss the cache entries later in this chapter. The remaining 29 bits are reserved for future use and set to 0.

In solicited advertisements, the Target Address contains the address of the interface that sent the solicitation. In unsolicited advertisements, this field contains the address of the interface whose link-layer address has changed. A possible option for the Options field is the target link-layer address.

Table 4-6 helps you to identify what you are looking at and summarizes the different processes.

Table 4-6. Identification of ND messages

Source address	Destination address	Message type
All-zero (`::`)	All-routers multicast	SLAAC
	Solicited node multicast	DAD
Unicast	Solicited node multicast	Resolve link-layer address
	Unicast	Unreachability detection

The ICMP Redirect Message

Routers issue ICMP Redirect messages to inform a node of a better first-hop node on the path to a given destination. A Redirect message can also inform a node that the destination used is in fact a neighbor on the same link and not a node on a remote subnet. The format of the ICMPv6 Redirect message is shown in Figure 4-14.

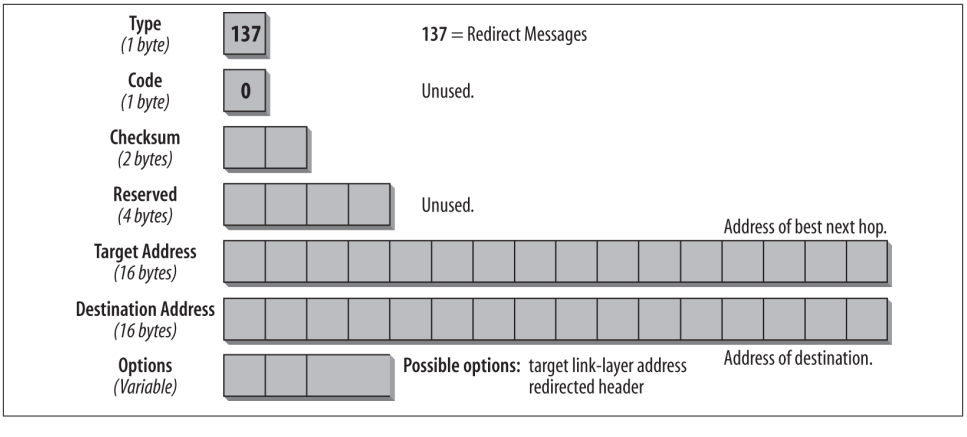

Figure 4-14. Format of the ICMP Redirect message

The Source address in the IP header must be the link-local address of the interface from which the message is sent. The Destination address in the IP header is the Source address from the packet that triggered the Redirect message. The hop limit is set to 255.

The Target Address field contains the link-local address of the interface that is a better next hop to use for the given Destination address. The Destination Address field contains the address of the destination that is redirected. If the address in the Target Address

field is the same as the address in the Destination Address field, the destination is a neighbor and not a remote node. The Options field contains the link-layer address for the target (the best next-hop router) if it is known. This is an improvement on the IPv4 version, in which the host needed to issue a separate ARP request to determine the link-layer address of the next-hop router. The remaining bits in the Options field contain as much of the redirected header as fits into the minimum IPv6 MTU of 1,280 bytes.

Inverse Neighbor Discovery

Inverse Neighbor Discovery (IND) is an extension to ND. It was originally designed for Frame Relay networks, but it can be used in other networks with similar requirements. IND is specified in RFC 3122. It consists of two messages: the IND Solicitation and the IND Advertisement message. The messages are used to determine the IPv6 address of hosts for which the link layer address is known. IND corresponds to the Reverse Address Resolution Protocol (RARP) used with IPv4. The messages have the same format as the ND messages. The IND Solicitation has a message type of 141 and the IND Advertisement of 142. The code field is always set to 0.

The Options field has the same format as in the ND messages and contains the same options. Two new IND-specific options have been defined. Option type 9 defines the Source Address list; option type 10 the Target Address list (see the overview in Table 4-7).

Source Address list—option type 9
: List of one or more IPv6 addresses of the interface, which is specified in the Source link-layer address option.

Target Address list—option type 10
: List of all IPv6 addresses of the interface, which is specified in the Target link-layer address list.

When a host wants to determine the IPv6 address of an interface for which it knows the link-layer address, it sends an IND solicitation to the all-nodes multicast address. On the link layer, the message is sent directly to the interface in question. The destination replies with an IND advertisement containing the Target Address list. If the interface has more IPv6 addresses than fit into a single Advertisement message, it must send multiple IND advertisements. Like in all other ND messages, the hop limit is set to 255, and messages with a hop limit lower than 255 must be ignored.

Neighbor Discovery Options

Neighbor Discovery messages contain a variable-size Options field that has the format shown in Figure 4-15.

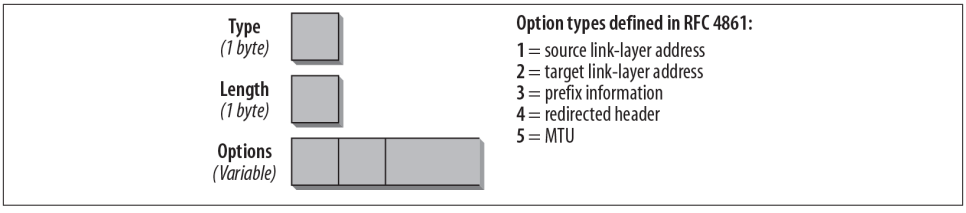

Figure 4-15. Format of the Options field

The Type field indicates what type of option follows. Table 4-7 shows an overview of the different options and the message types in which they are used.

The Length field indicates the length of the option. Value 0 is invalid for this field, and packets with this value must be discarded. The calculation of the length includes the Type and Length fields.

Table 4-7. Overview of ND options

Option type	RFC	Used in
Type 1 Source link-layer address	RFC 4861	Neighbor Solicitation Router Solicitation Router Advertisement IND Solicitation/Advertisement
Type 2 Target link-layer address	RFC 4861	Neighbor Advertisement Redirect IND Solicitation/Advertisement
Type 3 Prefix	RFC 4861	Router Advertisement
Type 4 Redirected header	RFC 4861	Redirect
Type 5 MTU	RFC 4861	Router Advertisement IND Solicitation/Advertisement
Type 7 Advertisement interval	RFC 6275	Router Advertisement (Mobile IPv6)
Type 8 Home Agent information	RFC 6275	Router Advertisement (Mobile IPv6)
Type 9 Source address list	RFC 3122	IND Solicitation
Type 10 Target address list	RFC 3122	IND Advertisement

More options have been defined in other specifications, for instance a CGA option for Secure Neighbor Discovery (see below).

For a full list of available options, refer to *http://bit.ly/1na84cG*.

Secure Neighbor Discovery

ND can be used for a number of attacks and should therefore be protected. An example of a Denial of Service attack is when a node on the link can both advertise itself as a default router and also send "forged" Router Advertisement messages that immediately time out all other default routers as well as all on-link prefixes.

The first protection is that packets coming from off-link (with a hop limit lower than 255) must be ignored. Further, the original ND specification suggests using IPsec to secure ND messages. However, this requires manual setup of security associations or the use of a key management protocol. The number of security associations to be configured for protecting ND can be very large, so this approach may be impractical.

The Secure Neighbor Discovery (SEND) working group was chartered with the goal to define the protocol support needed to secure ND. Three different trust models were outlined, roughly corresponding to secured corporate intranets, public wireless access networks, and pure ad hoc networks. A number of possible threats are discussed relating to these trust models. Refer to RFC 3756 for more details.

The SEND protocol, defined in RFC 3971 (updated by RFC 6494, RFC 6495, RFC 6980), is designed to counter the threats to ND. SEND can be used in environments where physical security on the link is not assured (such as over wireless) and attacks on ND are a concern. The following components are specified in RFC 3971:

- The authority of routers must be certified before it can be used as default router. A host must be configured with a trust anchor to which the router has a certification path. *Certification Path Solicitation and Advertisement* messages are used to discover a certification path to the trust anchor.
- *Cryptographically Generated Addresses* (CGAs) are used to make sure that the sender of a Neighbor Discovery message is the owner of the claimed address. A public-private key pair is generated by all nodes before they can claim an address. A new ND option, the CGA option, is used to carry the public key and associated parameters.
- Another new ND option, the *RSA Signature* option, is used to protect all messages relating to Neighbor and Router Discovery.
- Two new ND options, the *Timestamp* and *Nonce* options, have been introduced to prevent replay attacks.

The SEND protocol uses Cryptographically Generated Addresses. SEND currently does not support the protection of ND messages for nodes configured with a static address or with addresses configured through IPv6 Stateless Address Autoconfiguration mechanisms. All new option types and messages are specified in RFC 3971.

Cryptographically Generated Addresses (CGAs) are specified in RFC 3972 (updated by RFC 4581, RFC 4982), which defines a method for binding a public signature key to an IPv6 address in the SEND protocol. CGA are IPv6 addresses for which the interface identifier is generated by computing a cryptographic one-way hash function from a public key and auxiliary parameters.

Due to a lack of implementations by vendors, we don't expect SEND to be used widely. It does not make sense in a dual-stack environment, because IPv4 is not secured either. SEND would only make sense in an IPv6-only network.

Router Advertisement in the Trace File

At this point, you all deserve a refreshment. The following trace file shows what ND looks like in the real world and illustrates what we have been talking about.

The screenshot in Figure 4-16 shows the details of a Router Advertisement with three Option fields. Besides initializing the IPv6 stack and configuring it for the prefix, we have not changed any of the default configuration parameters on the router. The options used in this case are options 1 (Source link-layer address), 5 (MTU size), and 3 (prefix information).

The Type field is set to 134, the value for a Router Advertisement. The Current Hop Limit has a value of 64. All nodes on this link will use this value for their Hop Count field. The M-Flag is set to 0, which means this interface will not use a DHCPv6 server to receive an IPv6 address. The O-Flag is also set to 0, which means the interface will not send out a DHCP information request to look for additional DHCP information (such as DNS or NTP servers). The Router Lifetime is set to 1,800, which indicates that this router is a default router. The first Option listed is type 1. The link-layer address in the detail screen contains the link-layer address of the router interface. The second option is of type 5, which can be used to change the MTU size from the default of 1,500 bytes (can be useful in some tunneling scenarios to prevent fragmentation of the tunneled packet). The third Option is of type 3 for prefix information. Note all the additional information that can be given with a prefix. The Prefix Length field specifies the number of bits valid for the prefix (i.e., the length of the subnet mask). The L-Bit is the on-link flag. If set, it indicates that this prefix can be used for on-link determination. If it is not set, the advertisement does not make a statement, and the prefix can be used for on- and off-link configuration. The A-Bit is the autonomous address configuration flag. If set, it indicates that the prefix can be used for autonomous address configuration. In this case, the host will generate an address by adding the interface identifier to the prefix or, if the privacy options are used, by adding a random number as an interface ID. The Valid Lifetime field specifies how long this prefix is valid (in milliseconds). The Preferred Lifetime specifies how long the address being configured with this prefix can remain in the preferred state (in milliseconds). The last field shows the prefix of `2001:db8:cafe:b0::` advertised by this router.

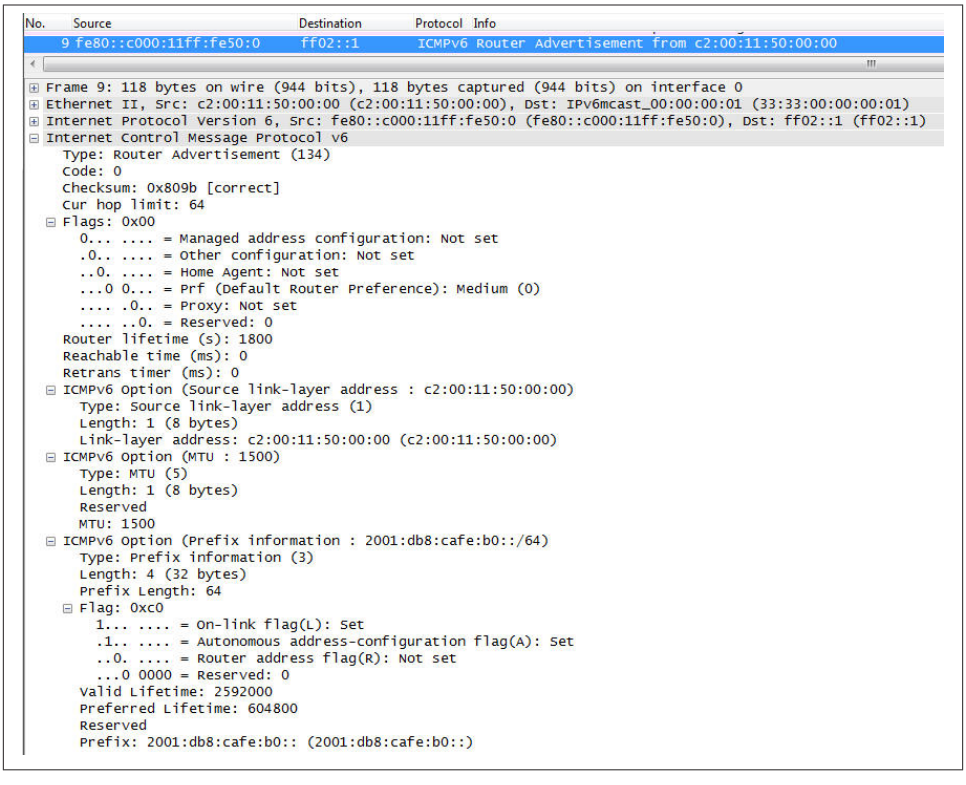

Figure 4-16. The Router Advertisement in a trace file

Draft-ietf-v6ops-dhcpv6-slaac-problem-00 describes how different operating systems have slightly different interpretations of the M-, O- and A-Flag. The reason for the ambiguity comes from the fact that the specification defines these flags not as prescriptive, but rather as advisory. Therefore different operating systems take different approaches of how to interpret these flags. The RFC reviews the standard definition of the flags, and provides a test result of several major desktop operating systems' behavior.

Link-Layer Address Resolution

Link-Layer Address Resolution is the process that happens when a node wants to determine the link-layer address of an interface for which it knows the IP address. With IPv4, this is the functionality of ARP. Link-Layer Address Resolution is performed only for nodes that are known to be on the same link (neighbors) and is never performed for multicast addresses.

With IPv6, this is a functionality accomplished with ND messages. A node wanting to resolve a link-layer address sends a Neighbor Solicitation message to the solicited node

multicast address of the neighbor. This solicitation message contains the link-layer address of the sender in the ND option field. If the destination is reachable, it replies with a Neighbor Advertisement message containing its link-layer address. If the resolving node does not receive an answer within a preconfigured number of attempts, the address resolution has failed.

Neighbor Unreachability Detection

A neighbor is considered reachable if the node has recently received a confirmation that packets sent to the neighbor have been received by its IP layer. This confirmation can come in one of two ways: it can be a Neighbor Advertisement in response to a Neighbor Solicitation, or it can be an upper-layer process that indicates the successful connection (e.g., an active TCP connection). In this case, the receipt of TCP acknowledgments implies the reachability of the neighbor.

To keep track of active and reachable connections, IPv6 nodes use different tables. Two important tables relating to ND are the Neighbor and Destination caches, which I discuss in the next section.

Neighbor Cache and Destination Cache

IPv6 nodes need to maintain different tables of information. Among these tables, the Neighbor Cache and Destination Cache are particularly important. Depending on the IPv6 stack you are working with, the implementation and the troubleshooting utilities will be different. But the information must be available on every IPv6 node.

Neighbor Cache
: The Neighbor Cache maintains a list of neighbors to which traffic has been sent recently. They are listed by their unicast IP addresses, and each entry contains information about the neighbor's link-layer address and a flag that indicates whether the neighbor is a router or host. This can be compared to the ARP cache in an IPv4 node. The entry also contains information about whether there are packets queued to be sent to that destination, information about the neighbor's reachability, and the time the next neighbor unreachability detection event is scheduled to take place.

Destination Cache
: This table maintains information about destinations to which traffic has been sent recently, including local and remote destinations. The Neighbor Cache can be seen as a subset of Destination Cache information. In case of remote destinations, the entry lists the link-layer address of the next-hop router. The Destination Cache is updated with information received by ICMP Redirect messages. It can also contain additional information about MTU sizes and round-trip timers.

The Neighbor and Destination Caches have been mentioned in regard to the Override flag that can be set in a Neighbor Advertisement message. If the Override flag is set, the

information in the Advertisement message should override existing Neighbor Cache entries and update any cached link-layer addresses in the cache of the host that receives the advertisement. If the O-Bit is not set, the advertisement will not update a cached link-layer address, but it will update an existing Neighbor Cache entry for which no link-layer address exists.

The screenshot in Figure 4-17 shows the Neighbor Cache entries of our router. There were two hosts on the link at the time the screenshot was taken.

```
ASW1#show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
2001:DB8:CAFE:B0:701:B16C:D114:7697          0 0800.27dd.edd3  REACH Vl10
FD85:87:69:2:C803:10FF:FED8:1C              17 ca03.10d8.001c  STALE Fa0/1
FD85:87:69:1:C802:10FF:FED8:1C              23 ca02.10d8.001c  STALE Fa0/0
FE80::6140:586:901D:6335                    21 0800.2777.0369  STALE Vl20
FE80::D4E4:D1E6:C310:AEF                    31 0800.27dd.edd3  STALE Vl10
2001:DB8:CAFE:B1:5EAB:F2A2:1ADF:1C8D         0 0800.2777.0369  REACH Vl20
FE80::C803:10FF:FED8:1C                     16 ca03.10d8.001c  STALE Fa0/1
FE80::C802:10FF:FED8:1C                     23 ca02.10d8.001c  STALE Fa0/0
```

Figure 4-17. Neighbor Cache entries on a router

According to RFC 4861, a Neighbor Cache entry can be in one of five states. The five states are explained in Table 4-8.

Table 4-8. States of Neighbor Cache entries

State	Description
Incomplete	Address resolution is currently being performed and awaiting either a response or a timeout. Specifically, a Neighbor Solicitation has been sent to the solicited-node multicast address of the target, but the corresponding Neighbor Advertisement has not yet been received.
Reachable	This neighbor is currently reachable, which means positive confirmation was received within the last `ReachableTime` milliseconds that the neighbor was functioning properly.
Stale	More than `ReachableTime` milliseconds have elapsed since the last positive confirmation that the forward path was functioning properly was received. No action will take place regarding this neighbor until a packet is sent.
Delay	This neighbor's Reachable Time has expired, and a packet was sent within the last `DelayFirstProbeTime` seconds. If no confirmation is received within the `DelayFirstProbeTime` seconds, send a Neighbor Solicitation and change the neighbor state to Probe state. The use of `Delay` allows upper-layer protocols additional time to provide reachability confirmation. Without this extra time, possible redundant traffic would be generated.
Probe	A reachability confirmation is being actively attempted by sending Neighbor Solicitations every `RetransTimer` milliseconds until reachability is confirmed.

If you are interested in the details about the timers, default values, and configuration options, refer to RFC 4861.

Neighbor Discovery and Fragmentation

As will be discussed in Chapter 6 on Security, ND-based attacks can be mitigated by implementing *RA Guard*. With RA Guard you can filter ND messages based on Layer 2 source and filtering rules. It has been found that fragmentation on Neighbor Discovery messages can pose a security risk, due to the fact that the fragmentation header can make it impossible for Layer 2 devices to properly filter ND messages. RFC 6980, "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery," explains the problem and specifies that all ND and SEND messages must not use fragmentation.

Refer to Chapter 6 for more information on RA Guard and *First Hop Security*.

Stateless Address Autoconfiguration (SLAAC)

The autoconfiguration capability of IPv6 saves network administrators a lot of work. It has been designed to ensure that manually configuring hosts before connecting them to the network is not required. Even larger sites with multiple networks and routers should not need a DHCP server to configure hosts. The autoconfiguration features of IPv6 will be a key feature of the protocol when all sorts of devices—such as televisions, refrigerators, DVD players, and mobile phones—use IP addresses. You don't want to depend on a DHCP server to use your home devices. It is also useful in public ad hoc networks to minimize administration. In an enterprise scenario we rather expect DHCPv6 to be used, mainly for traceability purposes.

DHCPv6 is discussed in Chapter 5.

IPv6 knows Stateless and Stateful Address Autoconfiguration. Stateful Address Autoconfiguration is DHCPv6. What's really new with IPv6 is that hosts can autoconfigure their IPv6 addresses without any manual configuration of the host. Some configuration might be done on the routers, but no DHCP servers are required for this configuration mechanism. To generate its IP address, a host uses a combination of local information, such as an interface ID based on its MAC address or a randomly chosen interface ID, and information received from routers. Routers can advertise one or multiple prefixes, and hosts determine prefix information from these advertisements. This also allows for simpler renumbering of a site: only the prefix information on the router has to be

changed. For instance, if you change your ISP and the new ISP assigns a new IPv6 prefix, you can configure your routers to advertise this new prefix, keeping the subnet IDs that you used with the old prefix. All hosts attached to those routers will renumber themselves through the autoconfiguration mechanism. You can find more on renumbering networks later in this chapter. If there is no router present, a host can generate only a link-local address with the prefix `fe80`. With this address only nodes on the link can be reached.

Stateless and Stateful Address Autoconfiguration can also be combined. For instance, a host can use Stateless Address Autoconfiguration to generate an IPv6 address but then use DHCPv6 for additional parameters. To configure hosts that use SLAAC for additional information (e.g., DNS servers), a Stateless DHCP server has been specified.

An IPv6 address is leased to a node for a certain lifetime. When the lifetime expires, the address becomes invalid. To make sure an address is unique on a link, a node runs the Duplicate Address Detection (DAD) process. The DAD algorithm is defined in RFC 4862. An IPv6 address can have different states:

Tentative address
: This is an address that has not yet been assigned. It is the state prior to the assignment, when uniqueness is being verified (during DAD). A node cannot communicate in the network using a tentative address. The only messages that can be processed with a tentative address are ND messages relating to the autoconfiguration process.

Preferred address
: This is an address that has been assigned to an interface and can be used without any restrictions for the lifetime assigned.

Deprecated address
: The use of this address is discouraged but not forbidden. A deprecated address might be one whose lifetime is about to expire. It can still be used to continue a communication that would disrupt a service if the address changed. It is no longer used as a Source address for newly established communications.

Valid address
: This term is used for both the Preferred and Deprecated address.

Invalid address
: An invalid address is not assigned to an interface. A valid address becomes invalid when its lifetime expires.

Optimistic address
: An address that is assigned to an interface and available for use, subject to restrictions, while the DAD process has not completed yet. This address state is introduced in RFC 4429, "Optimistic Duplicate Address Detection (DAD) for IPv6." Optimistic

Duplicate Address Detection (DAD) is a modification of the existing ND and SLAAC processes. The intention is to minimize address configuration delays in the successful case, and to reduce disruption as far as possible in the failure case.

Autoconfiguration as defined in RFC 4862 only applies to hosts, not to routers. Routers should be configured in a different way. A router can use Stateless Autoconfiguration for the generation of its link-local addresses, and it must use the DAD process for each of its addresses.

When a node is autoconfigured, the following steps are performed:

1. A link-local address is generated by using the link-local prefix of `fe80` and appending the interface identifier. This address is a *tentative* address.
2. The node joins the following multicast groups: the all-nodes multicast group (`ff02::1`) and the solicited-node multicast group for the tentative address (from step 1).
3. A Neighbor Solicitation message is sent out with the tentative address as the target address. The IP Source address of this message is the all-zeros address; the IP Destination address is the solicited-node multicast address. This is the DAD process to detect whether another node on the link already uses this address. If there is such a node, it replies with a Neighbor Advertisement message, and the autoconfiguration mechanism stops if the interface ID is an EUI-64 ID. If the interface ID is a randomized number, new combinations will be tested. How this is done in detail depends on the operating system used. If the process stops, manual reconfiguration of the host is required. If there is no answer to the Neighbor Solicitation, it is safe to use the address; the address is assigned to the interface and the state of the address changes to *preferred*. IP connectivity on the local link is now established. So far, the process is the same for hosts and routers, but only hosts perform the next step.
4. In order to determine if there are IPv6 routers out there, to find a default gateway, and if there is a prefix to use for SLAAC, the host sends a Router Solicitation message to the all-routers multicast group of `ff02::2`.
5. All routers on the link reply with a Router Advertisement. For each prefix in Router Advertisements with the Autonomous Configuration flag set, the node generates an address, combining the prefix with the interface identifier. These addresses are added to the list of assigned addresses for the interface.

All addresses must be verified with a Neighbor Solicitation message (DAD) before they are assigned. If the link-local address was generated through the autoconfiguration mechanism using the interface identifier, uniqueness has been verified in step 3 and may not need to be repeated for additional addresses that use the same interface

identifier. All other addresses configured manually or through Stateful configuration need to be verified individually. Multihomed hosts perform autoconfiguration for each interface.

So we have two types of addresses based on how the interface ID is generated:

Permanent IPv6 address
: This is a Global Unicast or ULA address assigned to the interface either via autoconfiguration (SLAAC or DHCPv6) or manually. It does not change as long as the interface is enabled and active.

Temporary IPv6 address
: This is a Global Unicast or ULA address assigned to the interface that uses a random interface ID that has a limited lifetime and changes in regular and configurable intervals.

All the different address types mentioned in these descriptions are discussed in Chapter 2.

The trace shown in the screenshot in Figure 4-18 was taken during the autoconfiguration process of our Windows 8 host. The figure shows some of the processes and message types discussed earlier, and the discussion of the trace summarizes the concepts in this section.

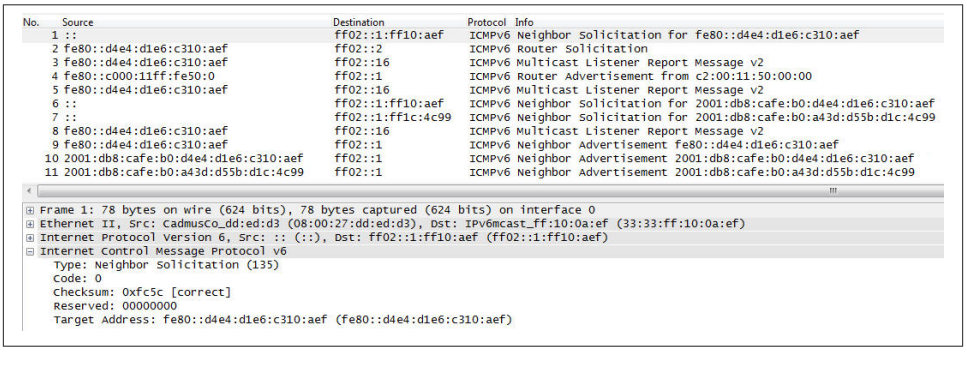

Figure 4-18. Autoconfiguration in the trace file

As long as the booting host does not have a valid IPv6 address, it sends all packets from its unspecified all-zeros address, represented as `::` in the trace file.

Packet 1
: In this first packet, the booting host (interface) sends a Neighbor Solicitation from the all-zeros address to the solicited node address in order to perform DAD. The Target Address field of the ICMPv6 header contains the link-local address of `fe80::d4e4:d1e6:c310:aef`. If another node on the link already used this address, it would reply with a Neighbor Advertisement message.

Packet 2
: With the second packet, it sends a Router Solicitation message (message type 133) to the all-routers multicast address `ff02::2`.

Packet 3
: With this packet, the host registers for the solicited node multicast address of `ff02::1:ff10:aef.` The packet contains a Hop-by-Hop Options header with option type 5 for MLD messages. The hop limit is set to 1.

Packet 4
: The router sends a Router Advertisement (RA) to the all-nodes multicast address of `ff02::1`, because the host does not have a valid preferred IPv6 address yet. With this RA the router advertises itself as default gateway (router lifetime set to 1,800 seconds). This RA has the M- and O-Bit set to zero (no DHCPv6) and the Autonomous Address Configuration flag set to one. The prefix option contains the prefix `2001:db8:cafe:b0::/64`. The details of this RA can be seen in Figure 4-16.

Packet 5
: With this packet, the host registers for the solicited node multicast address of `ff02::1:ff1c:4c99`. The packet contains a Hop-by-Hop Options header with option type 5 for MLD messages. The hop limit is set to 1.

Packet 6 and 7
: The host performs the DAD test for the addresses formed by combining the global prefix received in the Router Advertisement (packet four) with the two interface IDs. The address with the interface ID ending in `aef` is the permanent address. The address with the interface ID ending in `4c99` is the address using the privacy option, or as Microsoft calls it, the temporary address. The packet is sent from the all-zeros address to the respective solicited node multicast address. The IPv6 address can be seen in the summary line of packet 6 and 7.

Packet 8
: With this packet, the host registers for its two solicited node multicast addresses. The packet contains the Hop-by-Hop Options header with option type 5 for MLD messages.

Packet 9, 10, and 11

With these three packets the host advertises itself on the link for the following addresses: the link-local of `fe80::d4e4:d1e6:c310:aef`, for the global address `2001:db8:cafe:b0:d4e4:d1e6:c310:aef`, and also for the global address with the temporary IID `2001:db8:cafe:b0:a43d:d55b:d1c:4c99`. The override flag is set in all three packets, which tells the receivers to update their neighbor cache. For communication going out to the Internet, this node should use the temporary global address ending in `4c99`.

When you take your own traces to analyze a boot process and use different operating systems, you will also find differences in the process shown in the trace file. The differences come from the fact that the RFCs leave room for interpretation of the standard, which can be implemented slightly differently by different vendors. As long as the "MUST" rules in the RFCs are followed, compliance should not be an issue. Note the "should"; this is where your trace file analysis expertise will be helpful in case of failures. Another reason for differences is that the implementation behavior depends on what level and state of the standardization has been implemented. If one operating system has an implementation of the privacy option, this stack will obviously behave differently from a stack that has not implemented the privacy option. So when you analyze processes of a specific operating system, get enough information from the vendor about which RFCs have been implemented. And if the vendor states that it implemented something, you can use your trace files to verify whether the stack works as it is supposed to.

So for instance, with regard to DAD and how Microsoft has implemented it, Windows skips the DAD process for temporary addresses since the interface ID is randomly generated. It will start using the temporary address right away and do a DAD process later to confirm the address is not in use. This is the *Opportunistic DAD* process mentioned earlier. Also note that on Microsoft operating systems, the default behavior is different on a client OS such as Windows 7 or 8 and a Windows server OS such as Windows 2008 or higher. The server operating systems do not enable the temporary addresses by default. Refer to Microsoft's documentation for more information.

Until recently, the choice for interface IDs with SLAAC was either hardware-based (EUI-64) or randomized and temporary (Privacy addresses). RFC 7217, "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)," defines stable randomized interface IDs. As the name implies they are stable but not based on the Layer 2 address.

For those of you who use and manage Microsoft operating systems, there is a great book which I recommend highly. It is written by Ed Horley and will save you a lot of sleepless nights and troubleshooting hours. The title is *Practical IPv6 for Windows Administrators* (Apress).

Network Renumbering

Renumbering a network means replacing an old prefix with a new prefix. This can become necessary for a number of reasons, a common one being a change of provider, which usually implies a change of prefix. Moving from the IPv4 world to the IPv6 world, prefixes and addresses become more and more flexible and are not uniquely identifying a network or hosts. The IPv6 addressing architecture adds the concept of interfaces having multiple addresses. This challenges us to develop new addressing concepts.

With the mechanisms given by ICMPv6, renumbering a network in an IPv6 world may become a lot easier in the future. It is a procedure that should be considered during creation of an IPv6 address plan. Currently there is not much operational experience.

Renumbering a network may encompass the following steps:

1. Each link in the network must be assigned a subprefix from the new prefix before beginning the procedure. This is important for the overall process, in order to ensure proper configuration of all relevant devices and services such as routers, switches, interfaces, DNS, DHCP, etc.
2. The DNS database must be updated with the addresses for interfaces from the new prefix, and addresses for interfaces from the old prefix must be removed. Obviously the changes to DNS must be coordinated with the changes to addresses assigned to interfaces. The propagation of this new information can be controlled by parameters such as the Time to Live (TTL) for DNS records and the update interval between primary and secondary DNS servers.
3. Switches and routers are prepared for the new prefix. All necessary changes in the routing infrastructure for the new prefix are added in parallel to the old prefix, while the old prefix is still used for datagram services. This includes not only routers and switches, but also firewalls, ingress and egress filters, and all other filtering functions. For propagating subnet prefix information to routers, the IPv6 Prefix option for DHCPv6 (RFC 3633) may be used. In the case where hosts use Stateless Address Autoconfiguration, the routers are not configured to advertise the prefix for autoconfiguration yet (meaning that the Autonomous Address Configuration flag is not set). This will be done once stable routing for the new prefix has been verified.
4. All access lists, route maps, and other network configuration options (e.g., name services other than DNS) that use IP addresses should be checked to ensure that hosts and services that use the new prefix will behave as they did with the old one.
5. Test and verify network infrastructure and routing for the new prefix.
6. Advertise the new prefix outside of the corporate network. Configure all border defense systems accordingly to protect the new prefix from outside attacks.
7. Assign addresses from the new prefix to interfaces on hosts while still retaining the addresses from the old prefix. If Stateless Autoconfiguration is used, the

autonomous address-config flag is set for the new prefix, so hosts configure addresses for the new prefix in addition to the old addresses. DHCP now assigns addresses from both prefixes if it is used. The new information can be propagated by using the DHCP Reconfigure message, which will cause every DHCP client to contact the DHCP server. The addresses from the new prefix will not be used until they are inserted into DNS.

8. When the new prefix has been fully integrated into the network infrastructure and tested for stable operation, hosts, switches, and routers can begin using the new prefix. Once the transition has completed, the old prefix will not be in use in the network and can be removed step by step from DNS.

Special attention has to be given to applications and devices that do not get their IP addresses from DHCP or DNS, or that cache or store IP address information locally.

This is a high-level view of a renumbering process and obviously—as all network administrators know well—there are many details and possible pitfalls to be considered. Thorough and careful planning of this process is a must. RFC 4192 describes this process. RFC 7010, "IPv6 Site Renumbering Gap Analysis," which provides a good overview and an update to RFC 4192, discusses currently available mechanisms and components to support renumbering and analyzes gaps and mechanisms that should be developed in order to make renumbering an easier process.

There is a specification for IPv6-to-IPv6 Network Prefix Translation (NPTv6, RFC 6296) that can also be used in the scenario of changing prefixes due to ISP changes. This sort of introduces NAT for IPv6, although not with the issues of port translation, because in IPv6 this would be a one-to-one mapping from an address perspective.

Refer to Chapter 7 for a discussion of NPTv6.

Path MTU Discovery

With IPv4, every router can fragment packets if needed. If a router cannot forward a packet because the MTU of the next link is smaller than the packet it has to send, the router fragments the packet. It cuts it into slices that fit the smaller MTU and sends it out as a set of fragments. The packet is then reassembled at the final destination. Depending on the network design, an IPv4 packet may be fragmented more than once during its travel through the network.

With IPv6, routers do not fragment packets anymore; the sender takes care of it. Path MTU Discovery tries to ensure that a packet is sent using the largest possible size that

is supported on a certain route. The Path MTU is the smallest link MTU of all links between a source and a destination. The discovery of the Path MTU is described in RFC 1981.

The discovery process works like this. First, a host assumes that the Path MTU is the same as the MTU of the first hop link and it uses that size. If the packet is too big for a certain router along the path to deliver the packet to the next link, the router discards the packet and sends back an ICMPv6 Packet Too Big message. Recall that this message type includes the MTU size of the next hop link. The host now uses this MTU for sending further packets to the same destination. The host will never go below the IPv6 minimum MTU size of 1,280 bytes, however. The process of receiving a Packet Too Big message and reducing the size of the packets can happen more than once before the packet reaches its destination. The discovery process ends when the packets arrive at the final destination.

Note that ICMPv6 Packet Too Big messages should not be blocked, as this will make fragmentation fail.

The path from a given source to a given destination can change, and so can the Path MTU. Smaller MTU sizes are discovered by getting Packet Too Big messages. An IPv6 host will try to increase the MTU size from time to time in order to be able to detect a larger Path MTU. Path MTU Discovery also supports multicast destinations. If the destination is multicast, there are many paths that copies of the packets may travel, and each path can have a different Path MTU. Packet Too Big messages will be generated just as with a unicast destination, and the packet size used by the sender is the smallest Path MTU of the whole set of destinations.

Multicast Listener Discovery

Multicast has been available in IPv4 since 1988 and is used to address a certain group of hosts at the same time. Instead of sending out a broadcast, which is not routable and has to be processed by every node on the subnet, the multicast packet is addressed to a multicast group address out of the Class D address range. Only the hosts that are members of that multicast group will process the packet. Multicast messages can be forwarded over routers. In order to make this routing efficient, a multicast group management protocol ensures that routers only forward multicast packets over interfaces to links with members of the multicast group.

In IPv6, multicast is an integral part of the protocol and is available on all IPv6 nodes. A new multicast address format has been defined with a prefix of `ff` and with added functionality by using scopes in addition to the group address. For example, a multicast

group address can be in a link-local scope (`ff02`), a site-local scope (`ff05`), or a global scope (`ff0e`).

For an explanation of the multicast address format and a list of scope identifiers, refer to Chapter 2.

Multicast group management in IPv4 is done through Internet Group Management Protocol (IGMP). Version 2 of IGMP is defined in RFC 2236. IPv6 uses ICMPv6 messages for the same functionality; initial development was based on the IGMPv2 specification. It is now called *Multicast Listener Discovery* (MLD). Version 1 of MLD is defined in RFC 2710 (updated by RFC 3590 and RFC 3810). In 2004, MLD version 2 was defined. It extends MLD version 1 to support the use of Source-Specific Multicast (SSM). It is based on IGMPv3 (RFC 3376) and is specified in RFC 3810 and RFC 4604. MLDv2 is compatible with MLD version 1.

MLD is the protocol that allows multicast listeners to register for multicast addresses they want to use, to ensure efficient routing. Therefore, a routing mechanism is needed to manage the forwarding of multicast messages. PIM (Protocol Independent Multicast, RFC 4601) can be used with IPv6 with minimal changes.

To find information about the status of PIM, go to the IETF working group at *http://www.ietf.org/html.charters/pim-charter.html*.

MLD is an asymmetric protocol. The behavior of *listeners*, i.e., nodes that want to receive messages destined for a specific multicast group, differs from the behavior of routers. For multicast addresses where a router is a listener, it uses both parts of the protocol. Routers use MLD to discover which multicast addresses have listeners on each of their links. For each attached link, the router keeps a list of listener addresses. It does not keep track of how many members are listening to an address. A multicast address is listed as long as there is at least one member on the link. A listener sends member reports for its multicast addresses. With these messages, it registers with routers on the link to receive the messages addressed to the respective multicast group. If the multicast address is not in the router's list for this link, the router adds the address to its list of multicast addresses to be forwarded over this interface. With a Done message, a listener deregisters for a multicast address. When the last member of a group deregisters for a multicast address, the router removes the address from its list for this link.

All MLD messages are sent with a link-local IPv6 Source address and a hop limit of one to make sure they remain on the local link. The packet has a Hop-by-Hop Options header with the Router Alert flag set. Thus, routers will not ignore the packet, even if they are not listening to the multicast group address in question. Refer to Chapter 2 for more information on Extension headers.

MLDv1

The following message types have been specified for MLDv1 (RFC 2710):

Multicast Listener Query—type 130
: Used by an IPv6 router to query the multicast addresses on a link. There are two types of queries: the general query used to determine the multicast addresses that have listeners on the link (in the general query, the multicast address field is set to zero) and the address-specific query used to find out whether there are members for a specific multicast address on a link (the multicast address field is set to the multicast address for which the query is done).

Multicast Listener Report—type 131
: Used by a listener to register for a multicast group. This can be an unsolicited registration or the answer to a multicast listener query from the router.

Multicast Listener Done—type 132
: Sent by a listener to deregister for a multicast address. When a router receives a Multicast Listener Done message from the last member of the multicast address on a link, it will delete the multicast address from its list of multicast addresses to be forwarded over this interface.

All three message types of MLDv1 have the same format, which is shown in Figure 4-19.

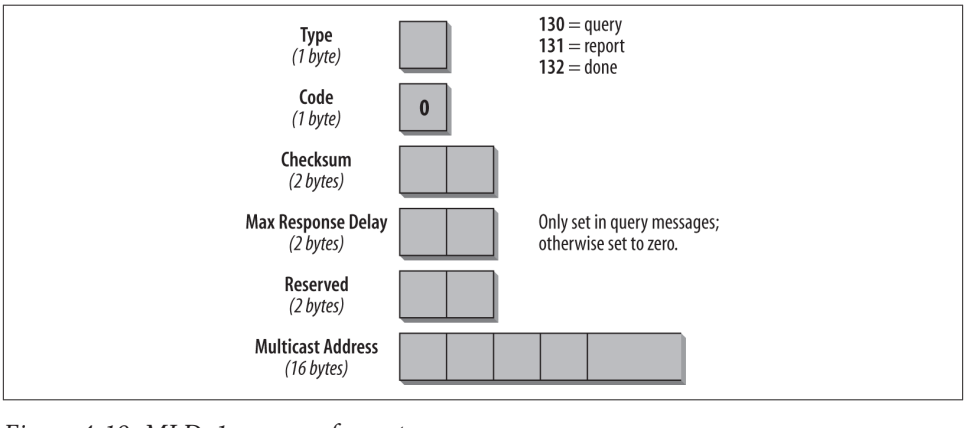

Figure 4-19. MLDv1 message format

The Type field is 130 for Multicast Listener Queries, 131 for Multicast Listener Reports, or 132 for Multicast Listener Done messages. The Code field is set to 0 by the sender and ignored by the receiver. The Maximum Response Delay field is used only in query messages. This is the maximum allowed delay (in milliseconds) in which a node has to send a report if it has a listener. In all other messages, this field is set to 0. The Multicast Address field is set to 0 in a general query. In an address-specific query, it contains the multicast group address to be queried. In Report and Done messages, this field contains either the multicast group to which a member listens (Report message) or the group it is leaving (Done message).

Routers send general queries to the link-local scope all-nodes multicast address `ff02::1`. Any station that wants to send a report in answer to a query starts a timer when it receives the query and is supposed to wait some random delay before sending the report. The maximum delay is the one specified in the Maximum Response Delay field in the query. If the station sees another station sending a report within that delay, it stops the process. Thus, multiple reports for the same address can be avoided. Group membership join reports and terminations are sent to the address in question.

The link-local scope all-nodes address (`ff02::1`) is a special address. It never sends a membership Report or a Done message. If an address has a scope of 1 (interface-local), MLD messages are never sent. Table 4-9 summarizes the message types and their Destination address.

Table 4-9. Message types and their destinations

Message type	IPv6 Destination address
General Query	Link-local scope all-nodes (`ff02::1`)
Multicast Address-Specific Query	The multicast address being queried
Report	The multicast address being reported
Done	Link-local scope all-routers (`ff02::2`)

RFC 2710 contains a lot of interesting and detailed information. It discusses various states that nodes can go through and includes state transition diagrams. There is also much detailed information on timers: how they are used, their default values, and how they can be configured.

MLDv2

MLD version 2 has been specified in RFC 3810 and RFC 4604. It is based on IGMPv3 (RFC 3376). MLDv2 adds the ability for a node to do *source filtering*, which means to report interest in listening to packets with a particular multicast address from only specific Source addresses or from all sources except for specific Source addresses. This is also referred to as Source-Specific Multicast (SSM) as opposed to Any Source Multicast (ASM), which is the name for the previous version, based on IGMPv2 (MLDv1).

The filter mode to support SSM may be either *INCLUDE* or *EXCLUDE*. In INCLUDE mode, reception of packets sent to the specified multicast address is enabled only from the source addresses listed in the source list. In EXCLUDE mode, reception of packets sent to the given multicast address is enabled from all source addresses except those listed in the source list.

There are two message types for MLDv2:

- Multicast Listener Query—type 130
- Version 2 Multicast Listener Report—type 143 (RFC 3810)

To be interoperable with MLDv1, MLDv2 implementations also need to support Version 1 Multicast Listener Report (type 131) and Version 1 Multicast Listener Done (type 132) messages.

For MLDv2 query messages, the MLD header is extended with the following fields, which are appended after the Multicast Address field shown in Figure 4-19:

Reserved—4 bits
: Must be set to zero.

S-Flag (Suppress Router-Side Processing)—1 bit
: When set to one, the S-Flag indicates to any receiving multicast routers that they have to suppress the normal timer updates they perform upon hearing a Query.

QRV (Querier's Robustness Variable)—3 bits
: If nonzero, the QRV field contains the Robustness Variable value used by the Querier. This value affects timers and the number of retries. Included in queries in order to synchronize all MLDv2 routers connected to the same link.

QQIC (Querier's Query Interval Code)—8 bits
: The Querier's Query Interval Code field specifies the Query Interval used by the Querier. Included in queries in order to synchronize all MLDv2 routers connected to the same link.

Number of Sources—16 bits
: The Number of Sources (N) field specifies how many Source addresses are present in the Query. This number is zero in a General Query or a Multicast Address Specific Query, and nonzero in a Multicast Address and Source Specific Query.

Source Address—variable length
: Contains the Source addresses. Length is defined by the number of addresses specified in the N-Field.

There are three different kinds of queries that can be sent:

General Query
: Sent by the Querier to learn which multicast addresses have listeners on an attached link. In a General Query, both the multicast address field and the number of sources field are set to zero.

Multicast Address Specific Query
: Sent by the Querier to learn if a particular multicast address has any listeners on an attached link. In a Multicast Address Specific Query, the Multicast Address field contains the multicast address of interest, while the Number of Sources field is set to zero.

Multicast Address and Source Specific Query
: Sent by the Querier to learn if any of the sources from the specified list for the particular multicast address has any listeners on an attached link or not. In a Multicast Address and Source Specific Query, the Multicast Address field contains the multicast address of interest, while the Source Address field(s) contain(s) the source address(es) of interest.

And Figure 4-20 shows what these fields look like in a trace file.

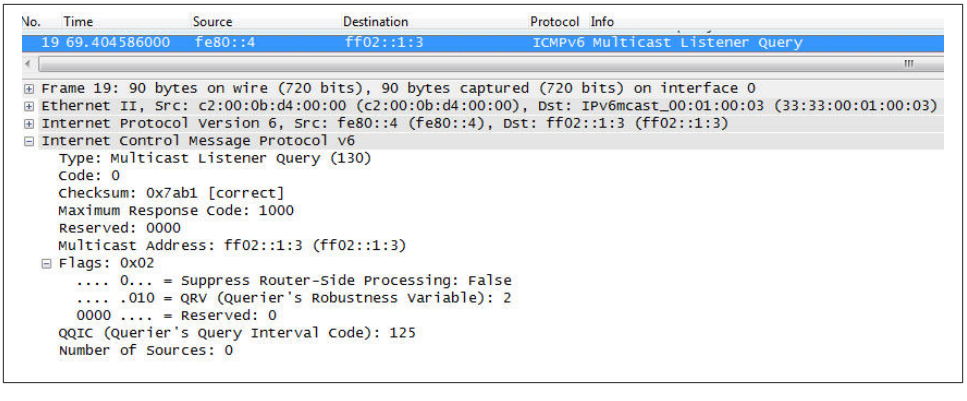

Figure 4-20. MLDv2 Query

This is a Multicast Address Specific Query, as it contains the multicast address to be checked (`ff02::1:3`) in the Multicast Address field. The Number of Sources field is set to zero. What you can't see in this screenshot is that the message has a Hop-by-Hop Options header with a Router Alert Option.

MLDv2 Listener Report messages have the following format:

- Type field—1 byte, set to 143
- Reserved—1 byte
- Checksum—2 bytes

- Reserved—2 bytes
- M-Field (Number of Multicast Address Records)—2 bytes
- Multicast Address Records—variable length, depending on number of addresses

Figure 4-21 shows the MLDv2 Listener Report in the trace file.

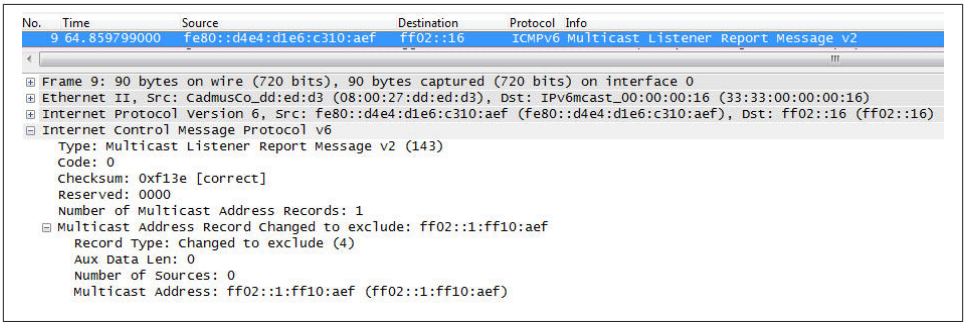

Figure 4-21. MLDv2 Listener Report in the trace file

In this case, there is only one multicast address record, so the value is set to one. Each Multicast Address Record is a block of fields that contain information on the MLD sender listening to a single multicast address on the interface from which the Report is sent. It includes a field specifying the number of sources and the list of sources for a particular multicast address. The *Record Type field* specifies the type of the Multicast Address Record described next.

A node that receives a Query on an interface responds with a *Current State Record* to report the state of the interface regarding the multicast address in question. The Current State Record can have one of two values:

Value 1—MODE_IS_INCLUDE
: Indicates that the interface has a filter mode of INCLUDE for the specified multicast address.

Value 2—MODE_IS_EXCLUDE
: Indicates that the interface has a filter mode of EXCLUDE for the specified multicast address.

If there is a change of the filter mode, a node sends a *Filter Mode Change Record*. This record is included in a report sent from the interface where the change occurred. The Filter Mode Change Record can have one of two values:

Value 3—CHANGE_TO_INCLUDE_MODE
: Indicates that the interface has changed to INCLUDE filter mode for the specified multicast address. The Source Address fields in this Multicast Address Record

contain the interface's new source list for the specified multicast address, if it is nonempty.

Value 4—CHANGE_TO_EXCLUDE_MODE
: Indicates that the interface has changed to EXCLUDE filter mode for the specified multicast address. The Source Address fields in this Multicast Address Record contain the interface's new source list for the specified multicast address, if it is nonempty.

If there is a change of source list, a node includes a *Source List Change Record* in a report sent from the interface where the change occurred. The Source List Change Record can have one of two values:

Value 5—ALLOW_NEW_SOURCES
: Indicates that the Source Address fields in this Multicast Address Record contain a list of the additional sources that the node wishes to listen to for packets sent to the specified multicast address. If the change was to an INCLUDE source list, these are the addresses that were added to the list; if the change was to an EXCLUDE source list, these are the addresses that were deleted from the list.

Value 6—BLOCK_OLD_SOURCES
: Indicates that the Source Address fields in this Multicast Address Record contain a list of the sources that the node no longer wishes to listen to for packets sent to the specified multicast address. If the change was to an INCLUDE source list, these are the addresses that were deleted from the list; if the change was to an EXCLUDE source list, these are the addresses that were added to the list.

Version 2 Multicast Listener Reports are sent with an IP Destination address of `ff02::16`. All MLDv2-capable multicast routers listen to this address.

For more information about Multicast and Anycast Group Membership, refer to the IETF Magma Group at *http://bit.ly/1rIgD3q*.

Multicast Router Discovery

Multicast Router Discovery (MRD) is a general mechanism that allows for the discovery of multicast routers. It does not depend on a specific multicast routing protocol. It is specified in RFC 4286 and introduces three new message types:

Multicast Router Advertisement (message type 151)
: This message is sent by routers to advertise that IP multicast forwarding is enabled. It is sent from a link-local Source address to the all-snoopers multicast address (`ff02::6a`).

Multicast Router Solicitation (message type 152)
: This message is sent by devices in order to solicit Advertisement messages from multicast routers. It is sent from a link-local Source address to the all-routers multicast address (`ff02::2`).

Multicast Router Termination (message type 153)
: This message is sent by routers to advertise that it stops IP multicast routing functions on an interface. It is sent from a link-local Source address to the all-snoopers multicast address (`ff02::6a`).

All MRD messages are sent with an IPv6 Hop Limit of 1 and contain the Router Alert Option.

Now that you know how all the cool functionality of IPv6 works, the next chapter discusses networking aspects such as Layer 2, Routing, Multicast, QoS, DHCPv6, and DNS. So read on.

References

Here's a list of the most important RFCs and drafts mentioned in this chapter. Sometimes I include additional subject-related RFCs for your personal further study.

RFCs

- RFC 1191, "Path MTU Discovery," 1991
- RFC 1981, "Path MTU Discovery" for IP version 6," 1996
- RFC 2236, "Internet Group Management Protocol, Version 2,"1997
- RFC 2362, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," 1998
- RFC 2365, "Administratively Scoped IP Multicast,"1998
- RFC 2463, "Internet Control Message Protocol (ICMPv6)," 1998
- RFC 2710, "Multicast Listener Discovery (MLD) for IPv6," 1999
- RFC 2715, "Interoperability Rules for Multicast Routing Protocols," 1999
- RFC 2894, "Router Renumbering for IPv6," 2000
- RFC 3041, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," 2001
- RFC 3122, "Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification," 2001
- RFC 3168, "The Addition of Explicit Congestion Notification (ECN) to IP," 2001
- RFC 3306, "Unicast-Prefix-based IPv6 Multicast Addresses," 2002

- RFC 3353, "Overview of IP Multicast in a Multi-Protocol Label Switching (MPLS) Environment," 2002
- RFC 3376, "Internet Group Management Protocol, Version 3," 2002
- RFC 3569, "An Overview of Source-Specific Multicast (SSM)," 2003
- RFC 3590, "Source Address Selection for the Multicast Listener Discovery (MLD) Protocol," 2003
- RFC 3756, "IPv6 Neighbor Discovery (ND) Trust Models and Threats," 2004
- RFC 3810, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6," 2004
- RFC 3971, ""Secure Neighbor Discovery," 2005
- RFC 3972, "Cryptographically Generated Addresses (CGA)," 2005
- RFC 3973, "Protocol Independent Multicast—Dense Mode (PIM-DM): Protocol Specification (Revised)," 2005
- RFC 4065, "Instructions for Seamoby and Experimental Mobility Protocol IANA Allocations," 2005
- RFC 4191, "Default Router Preferences and More-Specific Routes," 2005
- RFC 4192, "Procedures for Renumbering an IPv6 Network without a Flag Day," 2005
- RFC 4213, "Basic Transition Mechanisms for IPv6 Hosts and Routers," 2005
- RFC 4286, "Multicast Router Discovery (MRD)," 2005
- RFC 4429, "Optimistic Duplicate Address Detection (DAD) for IPv6," 2006
- RFC 4443, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," 2006
- RFC 4581, "Cryptographically Generated Addresses (CGA) Extension Field Format," 2006
- RFC 4604, "Using MLDv2 for Source Specific Multicast," 2006
- RFC 4620, "IPv6 Node Information Queries," 2006
- RFC 4861, "Neighbor Discovery for IP Version 6," 2007
- RFC 4862, "IPv6 Stateless Address Autoconfiguration," 2007
- RFC 4884, "Extended ICMP to Support Multi-Part Messages," 2007
- RFC 4890, "Recommendations for Filtering ICMPv6 Messages in Firewalls," 2007
- RFC 4982, "Support for Multiple Hash Algorithms in Cryptographically Generated Addresses (CGAs)," 2007
- RFC 5059, "Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM)," 2008

- RFC 5722, "Handling of Overlapping IPv6 Fragments," 2009
- RFC 5796, "Authentication and Confidentiality in PIM-SM Link-Local Messages," 2010
- RFC 5871, "IANA Allocation Guidelines for the IPv6 Routing Header," 2010
- RFC 5887, "Renumbering Still Needs Work," 2010
- RFC 5942, "IPv6 Subnet Model: the Relationship between Links and Subnet Prefixes," 2012
- RFC 6104, "Rogue IPv6 Router Advertisement Problem Statement," 2011
- RFC 6105, "IPv6 Router Advertisement Guard," 2011
- RFC 6226, "PIM Group-to-Rendezvous-Point Mapping," 2011
- RFC 6275, "Mobility Support in IPv6," 2011
- RFC 6434, "IPv6 Node Requirements," 2011
- RFC 6494, "Certificate Profile and Certificate Management for SEcure Neighbor Discovery (SEND)," 2012
- RFC 6495, "Subject Key Identifier (SKI) SEcure Neighbor Discovery (SEND) Name Type Fields," 2012
- RFC 6553, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams," 2012
- RFC 6554, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)," 2012
- RFC 6564, "A Uniform Format for IPv6 Extension Headers," 2012
- RFC 6583, "Operational Neighbor Discovery Problems," 2012
- RFC 6603, "Prefix Exclude Option for DHCPv6-based Prefix Delegation," 2012
- RFC 6724, "Default Address Selection for Internet Protocol version 6 (IPv6)," 2012
- RFC 6775, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," 2012
- RFC 6791, "Stateless Source Address Mapping for ICMPv6 Packets," 2012
- RFC 6866, "Problem Statement for Renumbering IPv6 Hosts with Static Addresses in Enterprise Networks," 2013
- RFC 6946, "Processing of IPv6 'Atomic' Fragments," 2013
- RFC 6980, "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery," 2013
- RFC 7048, "Neighbor Unreachability Detection Is Too Impatient," 2014
- RFC 7010, "IPv6 Site Renumbering Gap Analysis," 2013

- RFC 7113, "Implementation Advice for IPv6 Router Advertisement Guard (RA Guard)," 2014
- RFC 7136, "Significance of IPv6 Interface Identifiers," 2014
- RFC 7217, "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)," 2014
- RFC 7279, "An Acceptable Use Policy for New ICMP Types and Codes," 2014

Drafts

Drafts can be found at *http://www.ietf.org/ID.html*. To locate the latest version of a draft, refer to *https://datatracker.ietf.org/public/pidtracker.cgi*. You can enter the draft name without a version number and the most current version will come up. If a draft does not show up, it was possibly deleted. If it was published as an RFC, the RFC number will be displayed. *http://tools.ietf.org/wg* is also a very useful site. More information on the process of standardization, RFCs, and drafts can be found in Appendix A.

Here's a list of drafts I refer to in this chapter, as well as interesting drafts that relate to the topics in this chapter:

"DHCPv6/SLAAC Address Configuration Interaction Problem Statement"
draft-ietf-v6ops-dhcpv6-slaac-problem-00

"Reducing Multicast in IPv6 Neighbor Discovery"
draft-yourtchenko-colitti-nd-reduce-multicast-00

"A comparison between the DHCPv6 and RA based host configuration"
draft-yourtchenko-ra-dhcpv6-comparison-00

CHAPTER 5
Networking

If you have read the previous chapters, you now know about the IPv6 basics, the new addressing architecture, the header format, and the Extension header architecture, as well as all about the new ICMPv6-based processes such as Neighbor Discovery (ND), Stateless Address Autoconfiguration (SLAAC), Path MTU Discovery, and Multicast Listener Discovery (MLD).

Before we dive into transition mechanisms and integration of IPv6 into the IPv4 network, this chapter covers several topics that are important in the network, such as Layer 2 support for IPv6, checksumming, multicast and how it has been extended for IPv6, available routing protocols, and quality of service. Last but not least, I discuss DHCPv6 and DNS. Even though IPv6 supports SLAAC, we expect DHCPv6 to be used in the enterprise networks, mainly due to the fact that organizations want to be able to log address use, which is not easily done if using SLAAC. And with the IPv6 address space coming into our networks in addition to still using IPv4, DNS becomes even more important than before.

Layer 2 Support for IPv6

IP sits between the Data Link layer and the Transport layer. One of the goals in the development of IPv6 was to be able to support as many different physical networks as possible and to require no changes in the Transport layer. This approach is called "IP over Everything." To make IP as independent as possible from the Data Link layer, it needs an interface to this layer, which can be Ethernet, ATM, Token Ring, or any other media. The interface needs to be flexible and must be able to adapt to different requirements. For this purpose, features such as Path MTU Discovery and Fragmentation have been optimized. For UDP and TCP, it should not matter whether IPv4 or IPv6 is used. Obviously, changes are needed whenever IP addresses are used because of the difference in the address format. All these requirements lead to changes within the IP layer itself. This section discusses the interface to the Data Link layer.

Different terms are used when the Data Link layer is discussed. The TCP/IP model has four layers, the first of which is called the Link layer. The OSI model has seven layers. It subdivides the Link layer of the TCP/IP model into two layers: the Physical layer and the Data Link layer. Thus, the term *Layer 2 Support for IPv6* refers to the second layer of the OSI model.

IPv6's independence from the physical network media is important. When a packet is sent from one network to another, we do not usually know in advance the kind of physical networks through which the packet will travel. IP cares only about the Destination address and finding a way to get there regardless of the network hardware used. IP then passes the packet to the Data Link layer. In 802 networks, the interface driver on the Data Link layer applies a Media Access Control (MAC) header to the datagram and sends it out to the physical network. The interface driver needs to be aware of the physical requirements for transmission. Each network's hardware technology defines a specific addressing mechanism. Neighbor Discovery, as described in Chapter 4, is used to map between IPv6 addresses and MAC addresses.

The rules and packet sizes for the transport of IPv6 datagrams differ depending on the Link layer technology. There is an RFC covering each technology in detail. This chapter summarizes the main points to consider; a list of the RFCs can be found in Appendix B.

In the previous edition we covered some of the RFCs in more detail, some that aren't even much used anymore today, such as Token Ring. I have changed the format of this chapter. I still cover Ethernet in more detail, as this is the probably most used and serves as an example. The others I will summarize shortly so you have a reference.

Ethernet (RFC 2464)

Ethernet is a widely used LAN technology developed in the early 1970s at Xerox. There are many different variants used: in the early days, Twisted Pair Ethernet, also known as 10Base-T and operating at 10 Mbps, and Fast Ethernet, also known as 100Base-T and operating at 100 Mbps, were common; today Gigabit Ethernet, also known as 1,000Base-T and operating at 1 Gbps, and 10 Gigabit Ethernet, also known as 10GE and operating at 10 Gbps, are common. And now even 40 and soon 100 Gigabit Ethernet are showing up. The race will go on. The Institute of Electrical and Electronic Engineers (IEEE) together with a number of IT and telecom companies have defined a new standard called "Ethernet for the First Mile" (EFM, IEEE 802.3ah), which could allow usage of the Ethernet standard for first-mile connections to homes and companies.

RFC 2464 describes the format of IPv6 datagrams transmitted over Ethernet and how link-local and stateless autoconfigured addresses are formed. It obsoletes RFC 1972 and supports all Ethernet variants and VLAN technologies, such as 802.1Q and Cisco's Inter-Switch Link (ISL). WiFi looks much like Ethernet from higher layers, so there is not even a specific RFC to cover it.

Ethernet hardware addresses use a 48-bit addressing scheme. Ethernet hardware manufacturers are assigned blocks of Ethernet addresses, known as *OUI* or *company ID*. No two Ethernet hardware interfaces should have the same address, because each vendor assigns the addresses within its block in sequence. An Ethernet frame can be of variable size, but it can be no smaller than 64 bytes and no larger than 1,518 bytes (header, data, and CRC). Packets over Ethernet have a default MTU of 1,500 bytes, although many devices support jumbo frames, which can be up to 9,000 bytes. A smaller MTU can be configured through Router Advertisements containing an MTU option or through manual configuration of each device. If a Router Advertisement contains an MTU larger than 1,500 bytes or larger than a manually configured MTU, the Router Advertisement must be ignored.

The Ethernet header contains the source and destination Ethernet addresses and the Ethernet type code. The Ethernet type code for IPv6 is `0x86DD`. Figure 5-1 shows the Ethernet header for an IPv6 datagram.

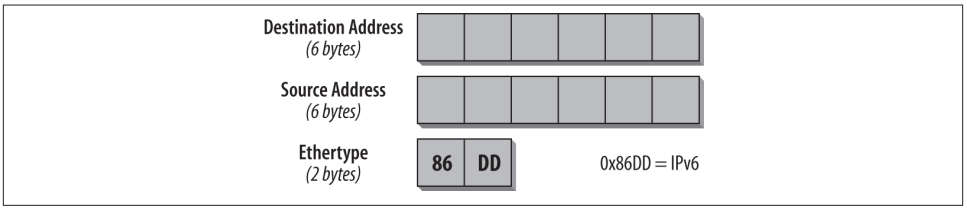

Figure 5-1. The Ethernet header for an IPv6 datagram

The Destination and Source Address fields each have six bytes, and the Ethernet Type field takes two bytes, containing the value `0x86DD` for IPv6.

For Stateless Address Autoconfiguration (SLAAC), the MAC address can be used to build the IPv6 interface ID. Chapter 2 explains how this process works. If the Destination address in the IPv6 header is a multicast address, the first two bytes of the MAC address are set to `3333` and the last four bytes are the last four bytes of the IPv6 destination multicast address. Figure 5-2 shows the format.

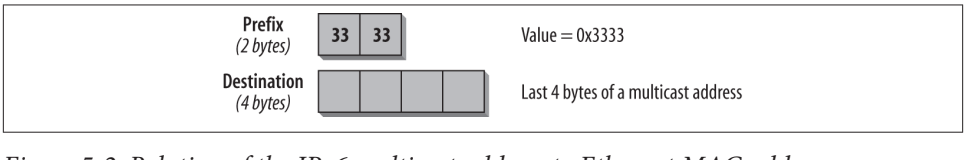

Figure 5-2. Relation of the IPv6 multicast address to Ethernet MAC address

Figure 5-3 shows how this looks in a trace file.

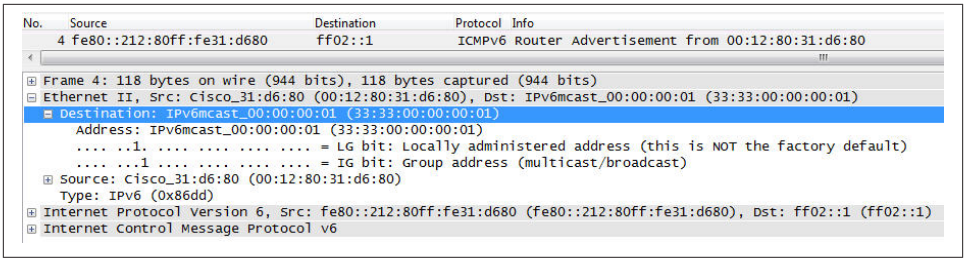

Figure 5-3. MAC header for an IPv6 multicast Destination address

In the summary line at the top of the figure, you can see the IPv6 Source address, which is the address of my router. The Destination address is the all-nodes multicast address. The Ethernet destination prefix shows `3333`, which identifies this MAC address as a multicast address, and the remaining four bytes contain the last four bytes of the IPv6 Destination address—in this case `00-00-00-01`. The Ethernet Source address contains the MAC address of the router, and the Ethertype has the value for IPv6, which is `0x86DD`.

There is an update to RFC 2464, one of the shortest RFCs I ever read; it is RFC 6085, "Address Mapping of IPv6 Multicast Packets on Ethernet." It allows the mapping of a multicast address into an Ethernet link-layer unicast address in the case where it is clear that only one address is relevant.

For useful information about Ethernet, refer to Charles E. Spurgeon's site. He is also the author of *Ethernet: The Definitive Guide* (O'Reilly).

Point-to-Point Protocol (RFC 5072)

Point-to-Point Protocol (PPP) is a mechanism for running IP and other network protocols over a serial link. It supports synchronous and asynchronous lines. RFC 5072 describes the method for transmitting IPv6 packets over PPP and how IPv6 link-local addresses are formed on PPP links. RFC 5172 defines the compression parameter for use in IPv6 datagram compression.

PPP's control protocol for IPv6, IPV6CP, is responsible for establishing and configuring IPv6 communication over PPP. One IPv6 packet can be encapsulated in a PPP Data Link layer frame, and the protocol field is set to `0x0057` for IPv6. If the PPP link is to support IPv6, the MTU size must be configured to IPv6's minimum MTU size of IPv6, which is 1,280 bytes. A higher value (1,500 bytes) is recommended.

IPV6CP has a distinct set of options for the negotiation of IPv6 parameters. Currently, the only defined options for IPV6CP are Interface-Identifier and IPv6-Compression

Protocol. A PPP interface does not have a MAC address. The Interface-Identifier option provides a way to negotiate a 64-bit interface identifier, which must be unique within the PPP link. The IPv6-Compression option is used to negotiate a specific packet compression protocol, which applies only to IPv6 packets transmitted over the PPP link. The option is not enabled by default.

IPv6 address negotiation is different from IPv4. It is done through ICMPv6 Neighbor Discovery and not through PPP, as it is with IPv4. For ISPs, PPP in combination with IPv6 offers many advantages. For instance, it is no longer a problem to assign static addresses to customers, because the IPv6 address space is large enough. With IPv4, due to the shortage of addresses, ISPs often have to use dynamic addresses. The IPv6 functionality for address autoconfiguration supports easy administration and customer configuration with minimal cost. Prefix assignment to the customer site can be done through Router Discovery or through IPv6 Prefix Options for DHCPv6 (RFC 3633). To get IPv6 to work over ADSL, ISPs need to choose an encapsulation that meets their needs, such as PPP over ATM (PPPoA) or PPP over Ethernet (PPPoE). IPv6 also has an impact on the Authentication, Authorization, Accounting (AAA) process. With IPV6CP, the address assignment occurs after the authentication. ISPs should note that Radius must support IPv6 attributes.

IEEE 802.15.4 (RFC 4944)

This standard specifies the physical layer and media access control for *low-rate wireless personal area networks* (LR-WPANs). It is designed to offer the lower fundamental network layers to support low-cost, low-speed communication between any type of devices. This standard does not define higher layers. There are specifications such as 6LoWPAN, ZigBee and others, which build on this standard.

6LoWPAN is the acronym for *IPv6 over Low power Wireless Personal Area Networks*. There is an IEEE working group. RFC 6282 defines a compression format for IPv6 datagrams over IEEE 802.15.4 networks. RFC 6775 defines optimizations to the Neighbor Discovery protocol (see Chapter 4) to work in low-power and lossy networks.

ZigBee is a specification for communication protocols using small, low-power radios. ZigBee allows networks to be formed ad hoc. Applications include light switches, electrical meters, traffic management systems, and any type of system that requires short-range wireless transfer of data at low rates. ZigBee is not optimized for IPv6 like 6LoWPAN is.

This type of technology will be the base for many future services, mostly sensor-based types of communication. They can be used in any area: industrial, safety (earthquake detection), health, fun, and entertainment. Typically these devices will have very limited resources; this is why they need lean, optimized stacks.

ATM (RFC 2492)

Asynchronous Transfer Mode (ATM) is a connection-oriented, high-speed network technology that is used in both LANs and WANs. It works over optical fiber and operates at up to gigabit speed by using special hardware and software mechanisms.

RFC 2492 describes the transmission of IPv6 packets over an ATM network in a companion document to RFC 2491, "IPv6 over Non-Broadcast Multiple Access (NBMA) Networks."

Frame Relay (RFC 2590)

Frame Relay is a connection-oriented, high-speed network technology used in WANs. It was developed in the Bell Labs in the late 1980s as part of the ISDN specification. The standard was refined in the early 1990s. By using a short, two-byte header, Frame Relay is very efficient in forwarding packets.

RFC 2590 specifies how IPv6 packets are transmitted over Frame Relay links, how IPv6 link-local addresses are formed, and how IPv6 addresses are mapped to Frame Relay addresses.

The next chapter covers routing and routing protocols in an overview. There are several choices you will have to make and we discuss the available options.

Upper-Layer Protocols

The impact of IPv6 on upper-layer protocols is minimal because the datagram service has not changed substantially. This chapter discusses UDP and TCP over IPv6, and describes changes for upper-layer protocols such as DNS, DHCP, SLP, FTP, Telnet, and HTTP when used over IPv6. The most important changes are always needed where an IP address is used. Any process or application that uses an IP address needs to be updated to be able to handle the extended 128-bit address format. Applications that use a hard-coded 32-bit IPv4 address should be updated to use a DNS name instead, so that DNS can return either an IPv4 or an IPv6 address to make the IP protocol fully transparent.

UDP/TCP and Checksums

Checksumming is done on different layers. Remember from Chapter 3, the IPv6 header does not have a checksum. But a checksum is important on the transport layer to determine packet delivery problems. Other upper-layer protocols may use a checksum, too. All checksum calculations that include the IP address in the calculation must be modified for IPv6 to accommodate the new 128-bit address.

Transport protocols such as UDP and TCP attach checksums to their packets. A checksum is generated using a *pseudoheader*. The TCP and UDP pseudoheader for IPv6

contains fields for Source and Destination address, payload length, and Next Header value (RFC 2460). If the IPv6 packet contains a routing header, the Destination address used in the pseudoheader is the address of the final destination. If the Source or Destination address was changed in transit, the value of the pseudoheader at the destination will not match the value of the initial packet, which causes checksum calculation failure and an error report.

Because the IPv6 address is so much longer than the IPv4 address, the IPv6 specification includes a new version of the pseudoheader. The IPv6 pseudoheader specification takes into account that an unknown number of Extension headers can be present before the UDP or TCP layer, which is essential when calculating the payload length for the pseudoheader. IPv6 nodes that receive a UDP packet with a value of 0 in the checksum field should discard the packet and log the error.

With IPv4, a checksum in the UDP header was optional. With IPv6, the computation of a checksum is mandatory for UDP.

The source node calculates and stores the checksum, and the destination node verifies it. Figure 5-4 shows the format of the pseudoheader that is built and used to calculate TCP and UDP checksums.

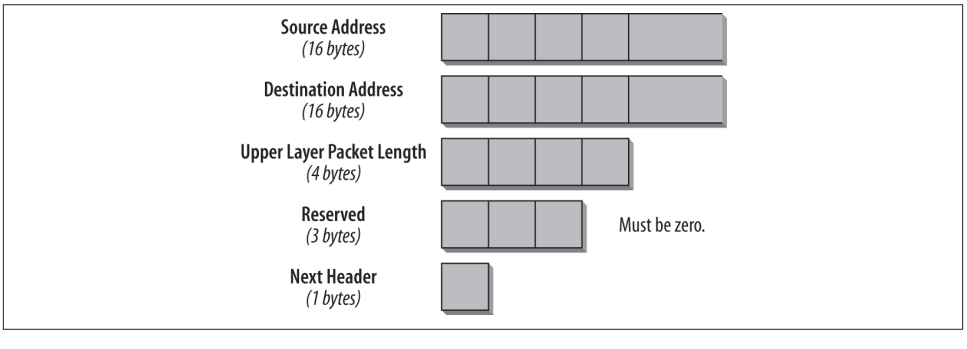

Figure 5-4. Format of the pseudoheader

The following list describes each of the fields:

Source Address (16 bytes)
: The Source address of the IPv6 packet.

Destination Address (16 bytes)
: The Destination address of the IPv6 packet. If there is a routing header in the packet, the address of the final destination is used for the checksum calculation. On the

first node, this address is the last address in the list of the routing header. At the final destination, this is the Destination address in the IPv6 header.

Upper Layer Packet Length (4 bytes)
: This field contains the length of the Upper-Layer Protocol header plus data.

Next Header (1 byte)
: The Next Header field identifies the type of the header using the values listed in Table 3-1 in Chapter 3.

The same algorithm used with IPv4 is used to calculate the checksum with IPv6. The 16-bit checksum is computed over the entire pseudoheader. By including the Source and Destination addresses in the checksum calculation, any alteration of the addresses en route would be detected.

Multicast

Multicast is discussed in different chapters of the book. Multicast addresses are described in Chapter 2, while all the multicast-based Neighbor Discovery functions are discussed in Chapter 4. This section here aims to consolidate it all and emphasize the importance of multicast in an IPv6 network.

Multicast is an efficient way to deliver data whenever it is one source distributing the same data to multiple receivers. This can be video or audio streaming, conferencing, distribution of financial information, or sports games, software updates, e-learning, and many more. With multicast, data packets do not have to be sent individually to each receiver. A single multicast packet can reach all receivers and therefore reduces the number of packets significantly, especially with a large number of receivers. Each multicast data stream is uniquely identified by its Source address (IPv6 unicast address) and its group or multicast IPv6 address. Multicast routing ensures packet delivery from the sender to all receivers. It must be enabled on all routers between the sender and the receivers. The router receives the multicast packet on the receiving interface and forwards it out over all other interfaces with registered receivers.

IPv4 multicast was not an integral part of the original IPv4 specification. It has been introduced later and then optimized based on experience and operational practice. The IPv6 multicast specification builds on that experience and has some advanced features that make it more efficient and scalable. Multicast is extensively used in IPv6 to perform basic functions, such as Neighbor Discovery.

Find information on how multicast is used for Neighbor Discovery in Chapter 4.

Multicast Addressing

The format of the multicast address makes use of the larger addressing space. The official prefix for multicast addresses is `ff00::/8`. The most important difference to IPv4 multicast is that the IPv6 multicast address has 4 bits to identify a scope. The scope determines how far the multicast will reach. A scope of 2 (`ff02`) is a link-local scope and will only be distributed on the link. A scope of 5 (`ff05`) is a site-local scope and will be routed to the border of the site. The global multicast scope is E (`ff0e`). Other scopes can be defined and configured by the network administrator.

Find all details about the multicast address and the flags and scope field in Chapter 2.

Group Membership Management

In order to receive data destined to a multicast group, a node must register for the multicast group. This is done through the use of a multicast management protocol. In IPv4 this is IGMP (Internet Group Management Protocol). In IPv6 it is called *MLD* (*Multicast Listener Discovery*) and is based on ICMP messages. There is MLDv1 (corresponding to the functionality of IGMPv2) and MLDv2 (corresponding to the functionality of IGMPv3). In most cases, MLDv2 is used. The main difference in MLDv2 is that it supports source-specific multicast. With source-specific multicast, a node cannot only register for a group, it can specify the desired source from which it wants to receive the data (or specific sources from which it does not want to receive the data).

The ICMPv6 messages defined are:

- Multicast Listener Query
- Multicast Listener Report
- Multicast Listener Done

You can find a detailed description of MLDv1 and MLDv2 messages in Chapter 4.

The multicast messages to register or deregister from a group are relevant only for the next hop router and therefore have a multicast scope of link-local (`ff02`). The router needs to know which multicast groups are listening on each of its interfaces. For this

purpose the router keeps a list of registered receivers for each multicast group, or in case of a more granular registration, for each data stream (sender/group). It will then only forward multicast data over an interface if the group is in its multicast list. As soon as the last member of a group has left the group, the router will stop forwarding that data for this group over this interface.

Multicast Layer 2 Protocols

With Layer 2 Multicast Management protocols, switches can be made multicast-aware, so that they don't have to flood all interfaces with multicast messages. *MLD snooping*, the IPv6 version of IGMP snooping, is available. With MLD snooping, switches will forward multicast messages only to ports that have listeners for the multicast group, as IPv6 multicast data is selectively forwarded to a list of ports that want to receive the data, instead of being flooded to all ports in a VLAN. This list is dynamically constructed by snooping IPv6 multicast control packets. Be aware and careful about the fact that multicast is essential to the basic functionality of IPv6. So make sure that your switch is aware of all the groups that a node must listen to.

Multicast Routing

With MLD, routers learn about listeners that are directly connected to their interfaces. To build the best path for multicast traffic from source to listeners, routers must exchange information about their listeners with each other.

For this purpose, an *MDT* (*Multicast Distribution Tree*) is used. The branches of the tree lead to the listeners. As listeners join and leave, branches are added or deleted. The tree has the root at the source of the traffic and is called *SPT* (*Shortest Path Tree*). The SPT is identified by the source address and the multicast group address. All routers that are part of a tree must maintain state for it. When multiple sources share the same group address, the *ST* (*Shared Tree*) is rooted in an administratively selected router called *Rendezvous Point*. A rendezvous point can handle multiple groups.

Control messages are always sent from the receiving end up toward the root of the tree. The process of finding the upstream neighbor is called *RPF* (*Reverse-Path Forwarding*) calculation. So while unicast routing is concerned about where the packet is going, multicast routing is concerned about where the packet is coming from. For each multicast data stream, only a single receiving interface is allowed on any given router. If a data stream were accepted over multiple interfaces, packet duplication would occur.

Protocol Independent Multicast

Multicast routing is the process of building the MDT. The topology information is maintained in the *TIB* (*Tree Information Base*). Many protocols had been developed to support this process. With deployment experience, the choice was cut down to several

flavors of *PIM* (*Protocol Independent Multicast*). For IPv6 three multicast routing protocols have been adopted:

PIM-SM (PIM Sparse Mode)
: Used when multiple sources transmit to the same group (videoconferencing, peer-to-peer gaming).

PIM-SSM (PIM Source-Specific Multicast)
: Subset of PIM-SM. Used when a single source transmits to multiple groups (content delivery such as video or audio).

PIM-Bidir (Bidirectional PIM)
: Used when all members of the group can be both receivers and sources.

For more information on Multicast and Multicast routing, refer to *Deploying IPv6 Networks* (Cisco Press) by Ciprian Popoviciu, Patrick Grossetete, and Eric Levy-Abegnoli.

Routing Protocols

Forwarding an IPv6 datagram beyond a directly attached subnet requires a router. Routers look at the datagram's destination IPv6 address and search for a matching prefix in their local routing tables. It is very important for the router to have all relevant destinations in its routing table. But how do they get there? We can enter them manually on all routers, which is called *static routing*, but this is not very economical. A much more efficient automatic approach can be achieved by deploying routing protocols. *Routing protocols* define exchange procedures to synchronize the routing table between routers dynamically (called *dynamic routing*). So the obvious advantage of using routing protocols is that they automatically adjust the routing tables to changes in the network without administrative intervention.

Routing information needs to be distributed either within an *Autonomous System* (*AS*) or between Autonomous Systems. An AS is defined as a set of networks governed by a single authority. Routing protocols that distribute information within an AS are called *Interior Gateway Protocols* (*IGP*). OSPF version 3 for IPv6, RIPng, IPv6 support on integrated IS-IS, and EIGRP for IPv6 belong to this category. Protocols that distribute information between Autonomous Systems are called *Exterior Gateway Protocols* (*EGP*). BGP-4 and its extensions for IPv6 represent such a protocol.

This section gives a short overview of common routing protocols such as RIPng, OSPF for IPv6, IS-IS, EIGRPv6, and BGP-4 support for IPv6. They represent the most important routing protocols in use today. I don't describe the protocols in detail, but simply mention the most important features that provide IPv6 support. This is followed by a

final summary of what routing protocol choices you will have to make for your future network environment with IPv6.

This chapter in previous editions of the book included in-depth coverage of OSPFv3 and IS-IS. At the time of writing this third edition there are many excellent books on the market that are focused on routing protocols (which was not so much the case back in 2005). I have therefore decided to shorten this chapter, as I do not see routing to be part of *IPv6 Essentials* anymore.

Most of the routing protocols mentioned here can be used only for the exchange of IPv6 routing information. If IPv4 and IPv6 are deployed on the same network, separate routing protocols must be implemented: one for IPv4 and one for IPv6—for example, OSPFv2 for IPv4 routing and OSPFv3 for IPv6 routing. Currently, the exceptions are the routing protocols BGP-4 and IS-IS. They can exchange routing information for both IP protocols within the same instance. In the future, OSPFv3 will also support both address families in one process (RFC 5838, "Support of Address Families for OSPFv3.")

The term *router* in this book stands for any device capable of IPv6 packet forwarding and/or processing the appropriate routing protocol.

The Routing Table

Each router maintains a routing table (also known as forwarding table) for each protocol it is configured to route. So a dual-stack router will usually have two routing tables, an IPv4 routing table and an IPv6 routing table. An IPv6 routing table is not so much different from an IPv4 routing table. The structure of an IPv6 address is simpler in the way that the first 64 bits are network information and the last 64 bits are the interface ID.

In this section when we mention routing table, we usually refer to the IPv6 routing table. Each entry in the IPv6 routing table represents an IPv6 destination, from now on called an IPv6 route. Each IPv6 route in the table is stored in the form of an IPv6 address prefix and its length. For each IPv6 route, additional information is stored in the routing table. The next hop information, for instance, tells the router where to forward a packet destined for this particular IPv6 route. Another type of information would be the metric of the IPv6 route, allowing the router to select the best path (smallest metric) to each IPv6 route in case of multiple entries.

Routing table lookup and content

For each incoming IPv6 packet, the router inspects the Destination address and looks it up in the routing table. For each IPv6 route in the routing table, the router applies the

prefix length to the Destination address to calculate a Destination address prefix. If this calculated prefix corresponds with the prefix of the IPv6 route, a match was found. To optimize the lookup, the searching algorithm looks through the entries based on prefix length, starting with longest prefix. If a match was found, the rest of the routing table can be ignored, as the longest matched prefix is always the preferred IPv6 route. Of course, this is a simplified representation of the lookup process. The actual algorithms behind it are complex and highly optimized.

Once the router has found a matching entry, the datagram is forwarded according to the next-hop information associated with this entry. In addition, the value of the hop limit within the datagram's IPv6 header is decremented by one. If no match is found in the routing table, there may be a default route (see below), or if there is no default route or the hop limit value has reached zero, the datagram is dropped. Figure 5-5 shows an example of such a routing table.

```
ASW1#show ipv6 route
IPv6 Routing Table - 12 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C   2001:DB8:CAFE:B0::/64 [0/0]
     via ::, Vlan10
L   2001:DB8:CAFE:B0:C000:11FF:FE50:0/128 [0/0]
     via ::, Vlan10
C   2001:DB8:CAFE:B1::/64 [0/0]
     via ::, Vlan20
L   2001:DB8:CAFE:B1:C000:11FF:FE50:0/128 [0/0]
     via ::, Vlan20
C   FD85:87:69:1::/64 [0/0]
     via ::, FastEthernet0/0
L   FD85:87:69:1::4/128 [0/0]
     via ::, FastEthernet0/0
C   FD85:87:69:2::/64 [0/0]
     via ::, FastEthernet0/1
L   FD85:87:69:2::4/128 [0/0]
     via ::, FastEthernet0/1
C   FD85:87:69:404::/64 [0/0]
     via ::, Loopback0
L   FD85:87:69:404::4/128 [0/0]
     via ::, Loopback0
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
```

Figure 5-5. An IPv6 routing table

For each route, the router keeps the following entries in the routing table:

IPv6 prefix and prefix length
: The prefix length defines the number of relevant bits of the IPv6 prefix. Normally the nonrelevant bits are set to zero in the routing table. The prefix length is also used to determine whether the Destination address of an incoming datagram matches this route.

Next Hop address
: The IPv6 address (normally link-local) of the first router along the path to the IPv6 route. If the route is directly connected to the router through a local interface, there is no need for a Next Hop address.

Next Hop interface
: The local interface of the router that is used to reach the Next Hop address.

Metric
: A number indicating the total distance to the destination. This metric depends on the routing protocol that has put this entry into the routing table. Metric calculations done by the different routing protocols cannot be compared to each other. If the same route is known by different routing protocols, the router must prefer one routing protocol over the other. This is done by assigning a priority value to each routing protocol (e.g., Cisco Systems calls this priority the *administrative distance*). Directly connected routes always have the best priority and are assigned the metric of the Next Hop interface (normally set to zero).

Timer
: The amount of time since the information about the route was last updated.

Route Source, also known as protocol
: The entity that provided information for this entry. For example, this may be a static entry, a directly connected route, or a route from a routing protocol, such as RIPng, OSPF for IPv6, BGP, etc.

Default route

A *default route* represents a route to all Destination addresses that are not explicitly listed in the routing table. It can be used when a router does not need to know all destinations specifically—for example, a router connecting a remote branch office to the main site. It does not need to know all routes of the entire autonomous systems. It only needs to know the local routes of the remote office; all other routes can be reached only via the connection to the main site, hence a default route.

A default route must be entered into the routing table just like any other route. The Next Hop address of the default route is also called the *default router* or *default gateway*. The entire data traffic for unknown routes is sent to the default router. It is assumed that the default router knows all the routes or has a default router itself. It is at the discretion of the network designer to determine whether and how such a chain of default routers

should be implemented. The top router of such a chain is typically a boundary router to another network area or autonomous system. It is here that the default route is entered statically and then distributed over the appropriate network area via a routing protocol. The advantage of distributing the default route is to reduce the number of routing updates to be distributed throughout the network area. Default routes should not be propagated further than intended—that is, they should not leave the network area or the autonomous system. A metric is assigned to the default route at its origin to establish precedence among multiple default routers. Default routes and distribution must be planned and implemented with care to avoid routing inconsistencies.

Any prefix with a length of zero is considered to be a default route, but normally an IPv6 prefix of `0:0:0:0:0:0:0:0` (or simply `::`) with a prefix length of zero is used. A Destination address of an incoming datagram will always match the default route, as the number of relevant bits for comparison is zero. The default route, however, is always the last route in the routing table, and hence a match is found only if no other routes in the routing table produce a match.

The most common use of a default route is on the endsystems attached to the network, such as PCs, servers, printers, etc. Each system must have a *default gateway* to send traffic to destinations outside the local subnet. Unlike with IPv4, there is no DHCPv6 option for a default gateway. An IPv6 node learns the default route through the Router Advertisement.

RIPng

History is repeating itself. Just as with IPv4, the first dynamic routing protocol to reach production was RIP, in this instance called *RIPng*. RIPng is a routing protocol based on a distance-vector algorithm known as the Bellman-Ford algorithm. Most of the concepts for RIPng have been taken over from RIPv1 and RIPv2, which have been implemented for IPv4 for quite some time. RIPv1 is defined in RFC 1058; RIPv2 in RFC 2453. RIPng is defined in RFC 2080 (January 1997). It is rarely used for reasons explained below.

Distance-Vector Algorithm for RIPng

RIPng uses a simple mechanism to determine the metric (cost) of a route. It basically counts the number of routers (hops) to the destination. Each router counts as one hop. Routes with a distance greater than or equal to 16 are considered to be unreachable. The router periodically distributes information about its routes to its directly connected neighbors using RIPng response messages. Upon receiving RIPng response messages from its neighbor, the router adds the distance between the neighbor and itself (usually one, as in one hop) to the metric of each route received. The router then processes the newly received route entry using the Bellman-Ford algorithm.

When the routers are first initialized, they know only their directly connected routes. This information is passed to all neighbors, processed, and then distributed to their neighbors. Eventually, all IPv6 routes are known by all routers. The routers keep sending response messages periodically to prevent valid routes from expiring. The time it takes for all routers to learn about the new routes is called *convergence time*.

Limitations of the protocol

RIPng, like the earlier versions of RIP, is primarily designed for use as an IGP in networks of moderate size. The limitations specified for RIP versions 1 and 2 apply to RIPng as well. They are described in the following list:

The RIPng diameter is limited.
: The longest path to any IPv6 route is limited to a metric of 15 when propagated with RIPng. Normally this corresponds with a path over a maximum of 15 hops. The protocol allows for larger costs to be assigned to any link, limiting the number of hops even further. Routes with a metric of 16 or greater are unreachable.

Routing loops can cause high convergence time.
: When IPv6 routes that are no longer valid are being propagated in a looped environment, RIPng continues to increase the metric by one. The routes would be passed around indefinitely ("counting to infinity"). The mechanism of limiting the metric to 16 prevents this from happening. The routes will circle until they reach the maximum metric and are eventually eliminated.

The metric does not reflect line speed.
: RIPng uses a fixed metric normally set to one for each link crossed. A route cannot be chosen based on bandwidth or real-time parameters such as measured delay, load, or reliability.

Changes in topology and preventing instability

A change in topology happens when a route is newly added or has gone down. Newly added routes are advertised within the next response message sent by the router having the direct connection to that route. Its neighbors process the route and pass it on to their neighbors. Eventually, all routers know about the newly added route.

What happens if a route goes down or a router crashes? These routes will eventually time out, as they are no longer being advertised. The questions are just how long this process will take and whether this time is acceptable for the network.

In certain scenarios, RIP has serious limitations. To overcome some of these limitations, two processes have been defined: with *split horizon* a router never advertises a route back over its next hop interface. An additional option is *split horizon with poison reverse*. With this option, a router always advertises a route back over its next hop interface with a metric of 16. Although RIPng supports IPv6, we do not recommend using it.

OSPF for IPv6 (OSPFv3)

OSPF for IPv6 modifies the existing OSPF for IPv4 to support IPv6. The fundamentals of OSPF for IPv4 remain unchanged. Some changes have been necessary to accommodate the increased address size of IPv6 and the changes in protocol semantics between IPv4 and IPv6. OSPF for IPv6 is defined in RFC 5340, which emphasizes the differences between OSPF for IPv4 and OSPF for IPv6. It contains a large number of references to the documentation of OSPF for IPv4, which makes it hard to read.

Overview of OSPF for IPv6

OSPF for IPv4 (OSPFv2) is standardized in RFC 2328. In addition to this document, several extensions to OSPF have been defined. RFC 1584 describes IPv4 multicast extensions to OSPF. RFC 3101 adds Not-So-Stubby Areas (NSSAs) to OSPF. RFC 5340 modifies OSPF to support the exchange of routing information for IPv6. OSPF for IPv6 has a new version number: version 3.

OSPF is classified as an IGP, which are used within autonomous systems. It was designed to overcome some of the limitations introduced by RIP, such as the small diameter, long convergence time, and a metric that does not reflect the characteristics of the network. In addition, OSPF handles a much larger routing table to accommodate large number of routes.

Differences between OSPF for IPv4 and OSPF for IPv6

Most of the concepts of OSPF for IPv4 have been retained; following is a brief overview of the changes:

Protocol processing per-link, not per-subnet
: IPv6 connects interfaces to links. Multiple IP subnets can be assigned to a single link, and two nodes can talk directly over a single link even if they do not share a common IP subnet. OSPF for IPv6 runs per-link instead of per-subnet. The terms *network* and *subnet* used in OSPF for IPv4 should be replaced with the term *link*, i.e., an OSPF interface now connects to a link instead of an IP subnet.

Removal of addressing semantics
: IPv6 addresses are no longer present in OSPF packet headers. They are only allowed as payload information.

  Router-LSA and Network-LSA do not contain IPv6 addresses. OSPF Router ID, Area ID, and Link State ID remain at 32 bits, so they cannot take the value of an IPv6 address. Designated Routers (DRs) and Backup Designated Routers (BDRs) are now always identified by their Router ID and not their IP address.

Flooding scope
: Each LSA type contains an explicit code to specify its flooding scope. This code is embedded in the LS type field. Three flooding scopes have been introduced: link-local, area, and AS.

Explicit support for multiple instances per link
: Multiple OSPF protocol instances can now run over a single link. This allows for separate Autonnomous Systems, each running OSPF, to use a common link. Another use of this feature is to have a single link belong to multiple areas.

Use of link-local addresses
: OSPF assumes that each interface has been assigned a link-local unicast address. All OSPF packets use the link-local address as the Source address. The routers learn the link-local addresses of all their neighbors and use these addresses as the next hop address. Packets sent on virtual links, however, must use either the global or local IP address as the source for OSPF packets.

Authentication
: Because OSPF for IPv6 runs over IPv6, it relies on the IP Authentication Header and the IP Encapsulating Security Payload to ensure integrity and authentication of routing exchanges. The authentication of OSPF for IPv4 has been removed. One integrity check remains, which comes in the form of the checksum that is calculated over the entire OSPF packet.

OSPF packet format changes
: See the section "Encapsulation in IP datagrams" on page 141.

LSA format changes
: Type 3 (Summary Link) has been renamed Inter-Area-Prefix-LSA.

  Type 4 (AS Summary Link) has been renamed Inter-Area-Router-LSA.

  Two new LSAs carry IPv6 prefix information in their payload. Link-LSA (type 8) carries the IPv6 address information of the local links, and Intra-Area-Prefix-LSA (type 9) carries the IPv6 prefixes of the router and network links.

  For other changes, such as Link State ID and the Options field, see the section "The Link State Database."

Handling unknown LSA types
: Instead of simply discarding them, OSPF for IPv6 introduces a more flexible way of handling unknown LSA types. A new LSA handling bit has been added to the LS Type field to allow flooding of unknown LSA types.

Stub area support
: The concept of stub areas has been retained in OSPF for IPv6. An additional rule specifies the flooding of unknown LSAs within the stub area.

Encapsulation in IP datagrams

The routers use OSPF packets to exchange LSA information and to establish and maintain neighbor relations (adjacencies). OSPF packets are directly encapsulated in IPv6, specified by protocol number 89 in the Next Header field of the IPv6 header. This means that OSPF does not run over TCP or UDP.

OSPF doesn't use fragmentation, therefore relying entirely on IP fragmentation when sending packets larger than the MTU. Fragmentation should be avoided whenever possible. Potentially large OSPF packets such as Database Description packets or Link State Update packets can easily be split into multiple packets by OSPF itself.

OSPF messages normally use the link-local IPv6 address of the outgoing interface as their Source addresses. The exceptions are messages sent on a virtual link. They use the local or global unicast address of the virtual link as their source. Depending on the situation, OSPF messages can be sent as a unicast to a specific neighbor or as a multicast to multiple neighbors. The following two multicast addresses are set aside for this purpose:

AllSPFRouters (`ff02::5`*)*
: All routers running OSPF must listen to this multicast address. Hello packets are always sent to this address, with the exception of nonbroadcast-capable networks. This address is also used for some packets during LSA flooding.

AllDRouters (`ff02::6`*)*
: Both the DR and the BDR on a multiaccess medium must listen to this multicast address. This address is used for some packets during LSA flooding.

  OSPF packets sent to the multicast address have link-local scope, and their IPv6 hop limit is set to 1. They will never be sent over multiple hops.

Support for multiple address families

In the original OSPFv3 specification, there was no support for IPv4. This means that in a dual-stacked network you have to run OSPFv2 for IPv4 and OSPFv3 for IPv6. OSPFv2 has some limitations, especially for mobile operations. OSPFv3 can overcome some of these limitations. RFC 5838, "Support of Address Families in OSPFv3," adds multiprotocol support for OSPFv3. With this RFC implemented, you can run two instances of OSPFv3, one for IPv4 and one for IPv6. Each OSPFv3 instance maintains its own adjacencies, link state database, and shortest path computation. The protocols are differentiated by using the Instance ID field in the packet header. Address-family enabled routers can establish peer relations based on IPv6 link-local addresses and advertise IPv4 routes. This way IPv4 routers in different subnets can peer with each other through the IPv6 network.

The alternative to this choice is to use IS-IS (described next), which also has multiple address family support. Some larger organizations have chosen to migrate their OSPFv2

environment to IS-IS for this reason, to have one protocol to manage their transitional dual-stack network.

Routing IPv6 with IS-IS

IPv6 support with IS-IS is defined in RFC 5308. This document is based on the specifications for integrated IS-IS as defined in RFC 1195. Without in-depth knowledge of integrated IS-IS, the IPv6 extension cannot be understood.

IS-IS originally defines the exchange of routing information between *Intermediate Systems* (ISs, otherwise known as routers) for the OSI network layer protocols *CLNP* (*Connectionless Network Protocol*) and *CONP* (*Connection-Oriented Network Protocol*). Other protocols use other routing protocols. Having separate routing protocols for each network layer is sometimes referred to as "ships in the night." Each routing protocol uses its own resources, such as CPU and memory, and therefore operate independent from each other.

Integrated IS-IS is an interior routing protocol based on link state updates. OSPF and IS-IS have many similarities: if you know one, the other is easy to grasp. OSPF runs within an AS, and i/IS-IS runs within a routing domain.

Integrated IS-IS provides for the inclusion of variable-length fields (Type, Length, Value fields, or TLVs) in all IS-IS packets (Hello, LSP, and SNP). Relevant addressing information is stored in TLV fields. Hello packets and LSP packets carry a field specifying the network layer protocols. Each supported network layer protocol is specified by its NLPID, assigned by ISO. The value of the IPv6 NLPID is 142 (`0x8E`).

The RFC defines two new TLVs for IPv6. They are described in the following list:

The IPv6 Reachability TLV (type 236)
: Defines the IPv6 prefix advertised within L1-LSP and L2-LSP. Within an L2-LSP, it can also be used to advertise an IPv6 prefix external to the routing domain by setting the external bit in the Control field. The following fields make up this TLV: Prefix Length, IPv6 Prefix, Metric (4 bytes), and the Control field.

IPv6 Interface Address TLV (type 232)
: Defines the IPv6 addresses of one or more interfaces of the router. It is advertised in Hello packets, L1-LSP, and L2-LSP. For Hello packets, it must contain the link-local IPv6 address assigned to the interface that is sending the Hello packet. In LSP, it must contain the global/unique-local addresses assigned to the router.

EIGRP for IPv6

Enhanced Interior Gateway Protocol (EIGRP) is an interior routing protocol (IGP) developed by Cisco Systems, Inc. It runs in an autonomous system called EIGRP domain. The main objective of EIGRP was to eliminate limitations of a distance vector

routing protocol (see the discussion of RIP, earlier in this section) without developing another link state based protocol. Link state protocols with their complexity and database demand higher CPU performance and more memory of the routers. EIGRP was therefore developed as a hybrid protocol combining the best of both worlds. It uses a so-called *Diffuse Update Algorithm (DUAL)* to calculate the routes. It allows for fast convergence and ensures loop-free operations at every instant throughout route computation. Only routers affected by a change are involved.

EIGRP has always supported different network layer protocols. For each network layer protocol, EIGRP runs a separate instance in a "ships in the night" fashion. There are modules for IPv4, IPX, Appletalk, and now also for IPv6. The basic functions for all protocols are the same. The semantics of the different protocols are implemented using protocol-dependent TLVs (Type, Length, Value) fields.

Cisco is opening up EIGRP as an open stack. It is in draft status at the time of writing. It remains to be seen if it is standardized and if other vendors are picking up on it.

BGP-4 Support for IPv6

There is no actual BGP for IPv6. The IPv6 support derives from the capability of BGP-4 to exchange information about network layer protocols other than IPv4. These multiprotocol extensions of BGP-4 are defined in RFC 4760. The base RFC that defines BGP-4 is RFC 4271. BGP-4 is the primary interdomain routing protocol used in the Internet.

BGP-4 overview

Each AS runs its interior routing protocol (RIP, OSPF, etc.) to distribute all routing information within the AS. BGP is an exterior routing protocol whose primary function is to exchange information about the reachability of networks between Autonomous Systems. Each AS receives a unique AS number assigned by the numbering authority, such as IANA and RIRs like ARIN, RIPE NCC etc.

BGP messages are carried on top of TCP connections, which can be established over either IPv4 or IPv6. The source and destination IP addresses of the datagram depend on the peer configuration. They are always unicast. BGP connections use the well-known TCP port 179. Remember that only one TCP connection is established between two peering routes.

BGP Multiprotocol Extension for IPv6

BGP-4 carries only three pieces of information that are truly IPv4-specific:

- `NLRI` (feasible and withdrawn) in the `UPDATE` message contains an IPv4 prefix.
- `NEXT_HOP` path attribute in the `UPDATE` message contains an IPv4 address.

- BGP Identifier is in the `OPEN` message and in the `AGGREGATOR` attribute.

To make BGP-4 available for other network layer protocols, the multiprotocol NLRI and its next-hop information must be added. RFC 4760 extends BGP to support multiple network layer protocols. IPv6 is one of the protocols supported, as emphasized in a separate document (RFC 2545). To accommodate the new requirement for multiprotocol support, BGP-4 adds two new attributes to advertise and withdraw multiprotocol NLRI. The BGP Identifier stays unchanged. BGP-4 routers with IPv6 extensions therefore still need a local IPv4 address. To establish a BGP connection exchanging IPv6 prefixes, the peering routers need to advertise the optional parameter BGP capability to indicate IPv6 support. BGP connections and route selection remain unchanged. Each implementer needs to extend the RIB to accommodate IPv6 routes. Policies need to take IPv6 NLRI and next-hop information into consideration for route selection.

An `UPDATE` message advertising only IPv6 NLRI sets the unfeasible route length field to 0 and carries no IPv4 NLRI. All advertised or withdrawn IPv6 routes are carried within the `MP_REACH_NLRI` and `MP_UNREACH_NLRI`. The `UPDATE` must carry the path attributes `ORIGIN` and `AS_PATH`; in IBGP connections it must also carry `LOCAL_PREF`. The `NEXT_HOP` attribute should not be carried. If the `UPDATE` message contains the `NEXT_HOP` attribute, the receiving peer must ignore it. All other attributes can be carried and are recognized.

An `UPDATE` message can advertise both IPv6 NLRI and IPv4 NLRI having the same path attributes. In this case, all fields can be used. For IPv6 NLRI, however, the `NEXT_HOP` attribute should be ignored. IPv4 and IPv6 NLRI are separated in the corresponding RIB.

Routing Protocol Choices for Network Designs with IPv6

While creating your network design for your future network including IPv6, you have many choices to make. Take your time to analyze and understand the new features of IPv6, because only then can you unfold its potential. If you simply try to mirror your IPv4 designs to the IPv6 world, you lose the opportunity to create networks that will not only be ready to support similar network services, but which scale to support future applications and services.

Routing principles in general are not different in an IPv6 network. We have some features that should optimize routing efficiency, such as a fixed length IPv6 header, the use of Extension headers that are only inserted if the options are needed, and the fact that IPv6 routers do not fragment anymore. We can eventually use the flow label to optimize data flows (once the community has agreed on a common practice).

The original plan for IPv6 was to not allocate Provider Independent (PI) addresses. It was an early design goal for IPv6 to not only solve the address situation, but also the problem with overflowing Internet routing tables. So the IPv6 address space is distributed to the RIRs based on geography in order to keep the root routing tables as small

as possible. However, it turned out that the zero PI space wasn't a sustainable policy in the real world. So we are back to having PI space, which partially breaks the hierarchical model based on geography and thereby creates more entries in global routing tables. Also, the IPv6 routing tables do not contain 32-bit address entries, but 128-bit address entries. And during the transition time in dual-stack networks, routers will maintain two routing tables, one for IPv4 and one for IPv6. The vendors will have to make sure that forwarding is efficient, done in hardware, and that the routers make efficient use of the resources so the routing tables use the minimum possible memory.

To summarize the information about the different routing protocols, we provide the following overview.

For IPv6 networks, the following IGPs are available:

RIPng (RFC 2080)
: RIP is a distance-vector protocol. It uses the Bellman-Ford algorithm. It is an easy-to-use protocol but it is far less efficient than OSPF and IS-IS. It has all the limitations that RIPv4 always had, such as a limited diameter, routing loops can create long convergence times, and the metrics don't represent line speeds because they are based on hop counts. It is not a recommended routing protocol for enterprise networks.

OSPFv3 (RFC 5340)
: OSPFv3 is a link-state based protocol. It uses the Dijkstra Algorithm to calculate a tree of shortest paths (SPF). It uses link-local addresses to exchange routing information (which is very helpful in the case of renumbering) and OSPFv2 authentication was removed, because it now uses standard IPv6 authentication. OSPFv3 as defined in RFC 5340 runs as a separate process. You still need OSPFv2 to manage your IPv4 network and each version maintains its separate routing tables. RFC 5838 defines extensions that support multiple address families in OSPFv3. At the time of writing, there is limited vendor support. Ask for the vendor's roadmaps.

IS-IS (RFC 5308)
: IS (Intermediate System) is OSI's term for router. IS-IS is a link-state protocol and also uses the Dijkstra Algorithm. It is an ISO protocol and does not rely on IP to exchange routing information. It is similar to OSPF but is considered to be easier to configure and manage by many administrators. IPv6 is fully integrated, and does not run as a separate process like in the current OSPF versions. For many years it has mainly been used in provider networks and wasn't so common in the United States. In the last years it became more common and has started to be used more and more in the enterprise space.

EIGRP for IPv6
: EIGRP was developed by Cisco Systems. It is a hybrid protocol taking the best of both the distance-vector and link-state based world and is based on the Diffuse

Update Algorithm (DUAL). It runs as a separate process, so to manage IPv4 and IPv6 two instances must be used. For larger environments we recommend the use of OSPF or IS-IS. Besides being more scalable, EIGRPv6 is currently only supported on Cisco gear, which creates a vendor lock-in and can also cause delays when updates are necessary, which are often faster in a competitive multivendor-supported standard. Cisco is opening up EIGRP and it is in draft status for standardization. It remains to be seen if it will be adopted by other vendors.

For your dual-stack network of the future the choices are most probably OSPFv2 and OSPFv3 versus IS-IS. RIPng doesn't scale in enterprise networks, and EIGRP is currently a proprietary solution that comes with a vendor lock-in.

There are probably not any real technically based pros or cons for OSPF versus IS-IS. Some companies decide to run both OSPF versions and have no issue with that. As multifamily support for OSPFv3 is on the horizon, this may become another option. Other companies prefer to migrate their OSPFv2 to IS-IS in order to have one single instance in the future. We expect IS-IS to become more and more popular in enterprise multiprotocol environments. Having one single instance also means that IPv4 and IPv6 are sharing fate. If you have a clear requirement that the routing of IPv4 and IPv6 should be independent of each other, two instances of OSPF may be your choice. The routing protocol decision will also be influenced by other factors such as building new know-how if you want to integrate IS-IS and have used OSPF so far. You will also have to learn about OSPFv3, but the difference to OSPFv2 is not so big. It also depends on corporate culture and market factors such as available know-how and resources on the market.

Quality of Service

In the beginning, the Internet was designed to be a simple communications platform, mainly used to support file transfer and email. Over the past 25+ years, it has grown to be a very complex global communications infrastructure with a multitude of applications and services. IPv4 is based on a simple packet switching model, delivering packets with best effort and no guarantee for delivery. TCP adds guaranteed delivery but has no options to control parameters such as delay and jitter or to do bandwidth allocation.

Multimedia services (such as voice over IP and videoconferencing) can have significant bandwidth demands and are often very sensitive to timely delivery. The Type of Service byte (ToS) in the IPv4 header was designed to provide prioritized treatment of certain traffic. However, it was never widely implemented, one reason being that its use would delay the forwarding of packets on routers. As there were almost no real-time services in those days, there was little pressure to find better solutions.

The development of IPv6, combined with the growing demand for real-time services—and, therefore, Quality of Service (QoS) features—was an opportunity to look for other

solutions. Despite the availability of several different approaches, the topic of QoS is still a matter of research, and there are many ideas under development.

Let me begin by saying that implementing QoS with IPv6 is not really different from implementing QoS with IPv4. This section aims to give a short introduction into QoS for readers that are not familiar with the concepts and then discuss the features in IPv6 that support QoS.

QoS Basics

The default IP model treats all packets alike. They are all forwarded with best-effort treatment according to the *first-come, first-served* principle. Which path a packet takes through the network depends on the available routers, routing tables, and general network load.

QoS protocols have the task of providing different data streams with priorities and guaranteeing qualities such as bandwidth and delay times. There are currently two main architectures: Integrated Services (IntServ) and Differentiated Services (DiffServ). Both architectures use traffic policies and can be combined to allow for QoS in the LAN as well as in the WAN.

Traffic policies can be used to make the transmission of data dependent on certain criteria—for example, whether there are enough resources available to forward the data according to its QoS requirements. Traffic policies can also monitor data streams and make adjustments or restrictions if necessary. Besides ensuring QoS requirements for delay-sensitive traffic, they can also be used for commercial reasons, such as controlling cost depending on different service levels.

Integrated Services

The Integrated Services architecture (IntServ) is based on the paradigm that bandwidth and all related resources per flow are reserved on an end-to-end basis. This presupposes that routers store information about flows and analyze each packet to determine whether it belongs to a specific flow in order to forward the packet according to the criteria for that specific flow.

RSVP (Resource Reservation Protocol, RFC 2205) is part of the IntServ architecture. RFC 2210, "The Use of RSVP with IETF Integrated Services," describes the use of RSVP with IntServ. RSVP is a signaling protocol used to reserve bandwidth and other QoS resources across an IP network. IntServ combined with RSVP can be complex to implement and, because of its limited scalability, is inadequate to offer a general QoS solution for the global Internet.

To find an updated list of IntServ service and parameter names and their associated values, go to *http://bit.ly/1na8Lmh*.

If you are interested in further reading about RSVP and other QoS signaling protocols, refer to the informational RFC 4094, "Analysis of Existing Quality-of-Service Signaling Protocols."

Differentiated Services

While IntServ offers the capability to allocate bandwidth to different flows, the Differentiated Services (DiffServ) architecture was designed to make a less granular differentiation of classes in order to increase its scalability and usability in large networks and in the Internet.

Differentiated Services is specified in RFCs 2474 and 2475. RFC 2474, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," specifies the DS field. This is implemented in the ToS field in the IPv4 header and the Traffic Class field in the IPv6 header. The DS field is used by DiffServ routers to determine the QoS forwarding requirements of packets. Communicating nodes can categorize their communication through a so-called *Per-Hop Behavior* (*PHB*). Based on the PHB, packets receive specific treatment on DiffServ routers.

A *DiffServ (DS) domain* is a contiguous group of DS routers that work with a common service policy implemented on all routers. A DS domain is defined by DS boundary routers. The boundary routers classify incoming data streams and ensure that all packets traversing the domain are labeled appropriately and use a Per-Hop Behavior from the set available for the domain. Routers within the domain choose the forwarding rules based on the DiffServ values in packets, which they map to the corresponding PHBs. The Differentiated Services Codepoint (*DSCP*; refer to Figure 5-6, shown later) value can use either the default mapping (`DSCP=0`) or an individually configured mapping for the domain. A DS domain usually consists of one network or a set of networks, which constitute an administrative unit.

A *DS region* is a set of contiguous DS domains. DS regions can ensure DS services for domain spanning paths. The single domains can use individual PHB definitions and PHB-codepoint mappings internally. Between the domains within a region, Traffic Conditioners are responsible for providing correct translation of the different PHBs and mappings. If the policies, PHB groups, and codepoint mappings are the same in all the domains within the region, no Traffic Conditioners are needed.

Packet classifiers choose packets from a data stream based on information in the packet headers and according to predefined rules. There are two types of classifiers: the

Behavior Aggregate classifier (BA) classifies packets based on the DS field, and the Multi Field classifier (MF) classifies packets based on either different header fields or a combination of header fields, such as Source or Destination address, DS field, protocol number, source or destination port, or information such as incoming interface.

QoS in IPv6 Protocols

The designers of IPv6 have focused not on requiring specific mechanisms for QoS, but on offering as much flexibility as possible to support different QoS mechanisms. This section describes the elements in the IPv6 header and the Extension headers that can be used for QoS services.

IPv6 Header

There are two fields in the IPv6 header that can be used for QoS: the Traffic Class and the Flow Label field.

Traffic Class. The use of the 1-byte Traffic Class field is specified in RFC 2474. As already mentioned, this RFC introduces the term *DS field* for the Traffic Class field. The goal of this specification is that DiffServ routers have a known set of DS routines, which are determined by the value in the DS field. These DSCP values are mapped to Per-Hop Behaviors (PHB) and can be either performance- or class-based. Figure 5-6 shows the DS field.

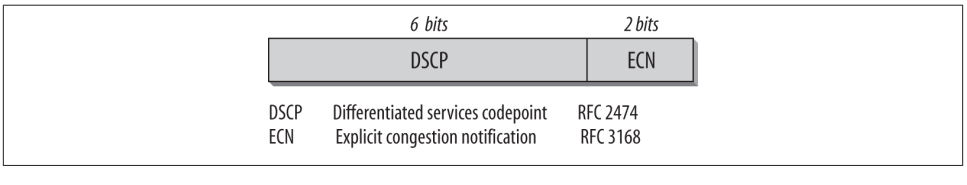

Figure 5-6. Format of the DS field

The DSCP field within the DS field (the six most significant bits of the DS field) is used for the codepoint, which specifies the PHB. With this field, 64 different codepoints can be specified. This codepoint pool has been divided into three parts to control the assignment of PHBs. Table 5-1 shows the division of the DSCP pools.

Table 5-1. The codepoint pools

Pool	Codepoint space	Assignment policy
1	xxxxx0	Standard use
2	xxxx11	Experimental/local use
3	xxxx01	Experimental/local use; potential standard use in the future

A pool of 32 recommended codepoints (pool 1) is assigned through formal standardization; a pool of 16 more codepoints (pool 2) is reserved for experimental or local use; the final pool of 16 codepoints (pool 3) is initially available for experimental or local use but should be used as an overflow pool if pool 1 is used up.

The PHBs specify how packets should be forwarded. A default PHB denominated by an all-zeros DS codepoint must be provided by any DS router. The default PHB describes the common, best-effort forwarding behavior available in existing routers. Such packets are forwarded without adhering to any priority policy; in other words, the network will deliver as many of these packets as possible as soon as possible, based on existing resources such as memory or processing capacity. Packets received with an undefined codepoint should also be forwarded as though they were marked for the default behavior.

The DS field does not specify PHBs; it specifies codepoints. The number of codepoints is limited to 64, whereas the number of PHBs is unlimited. There are recommended mappings of codepoints to PHBs. These mappings can be defined individually within administrative domains, which makes the number of possible PHBs unlimited. The coding rules for PHB IDs are specified in RFC 3140, "Per Hop Behavior Identification Codes."

RFC 2597 defines a PHB group called *Assured Forwarding (AF)*. Assured Forwarding PHB group is a means for a provider DS domain to offer different levels of forwarding assurances for IP packets received from a customer DS domain. Four AF classes are defined, where each AF class is in each DS node allocated a certain amount of forwarding resources (buffer space and bandwidth). IP packets that wish to use the services provided by the AF PHB group are assigned by the customer or the provider DS domain into one or more of these AF classes according to the services that the customer has subscribed to. RFC 3246 defines a PHB called *Expedited Forwarding (EF)*. The intent of the EF PHB is to provide a building block for low loss, low delay, and low jitter services.

Recommended codepoints and PHB IDs are assigned by IANA. The list of codepoints can be found at *http://www.iana.org/assignments/dscp-registry*, and the list of PHB IDs at *http://www.iana.org/assignments/phbid-codes*.

Figure 5-7 shows the DS field in a trace file.

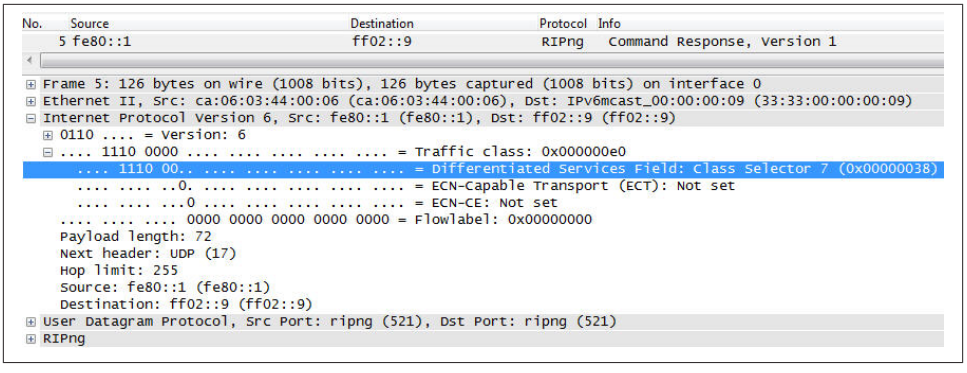

Figure 5-7. The DS field in a trace file

This is a RIPng (RIP Next Generation) Response from our router. It is sent to the RIP Routers Multicast address of `ff02::9`. The DS field is set to `0xE0` (decimal notation `224`, binary notation `1110 0000`).

The remaining two bits of the DS field (see Figure 5-6) are not used according to RFC 2474, and are specified in RFC 3168, "The Addition of Explicit Congestion Notification (ECN) to IP." They provide four possible codepoints (`00` to `11`) that are used for Congestion Notification. Traditionally the overload of a router could only be determined based on packet loss. With the use of these Congestion Notification codepoints, a router can signal overload before packet loss. This method is similar to Frame Relay's use of BECNs and FECNs (Backwards and Forwards Explicit Congestion Notification, respectively).

The two bits are used as follows:

- 00: Packet does not use ECN.
- 01/10: Sender and receiver are ECN-enabled.
- 11: Router signals congestion.

It should be mentioned that it has become more and more common that even cheaper switches can interpret the DSCP values and put packets in different queues as a consequence.

Flow Label. The 20-bit Flow Label field in the IPv6 header may be used by a source to label packets for which it requests special handling by the IPv6 routers, such as non-default QoS or real-time service. A flow label is assigned to a flow by the flow's source node. Between a sender and a receiver, there can be multiple flows active in parallel, along with the exchange of packets with no QoS requirements. New flow labels must be chosen randomly from the range `00001` to `FFFFF`. The purpose of the random

allocation is to make any combination of bits within the Flow Label field suitable for use as a hash key by routers for looking up the state associated with the flow.

Hosts or routers that do not support the functions of the Flow Label field (most of today's applications, which will not be modified to use the Flow Label, or which do not need QoS handling) are required to set the field to all zeros when sending a packet, to pass the field on unchanged when forwarding a packet, and to ignore the field content when receiving a packet.

All packets belonging to the same flow must be sent with the same IP Source address, IP Destination address, identical source and destination ports, and a nonzero flow label. If any of these packets includes a Hop-By-Hop Options header, they all must be originated with the same Hop-By-Hop Options header contents (excluding the Next Header field of the Hop-By-Hop Options header, which is allowed to differ).((("Hop-by-Hop Options header"))) If any packet includes a Routing Extension header, they all must be created with the same contents in all Extension headers up to and including the Routing Extension header (again excluding the Next Header field in the Routing Extension header). The routers or receivers are allowed to verify that these conditions are satisfied. If a violation of these consistency rules is detected, a corresponding error message is returned, indicating the exact location of the rule violation.

The handling of the flow label on routers is efficient, and when IPsec is used, it is always available because the IPv6 header is not encrypted by ESP or authenticated by AH (in transport mode). This implies that the integrity of the information in the DS field cannot be guaranteed by IPsec.

RFC 6437, "IPv6 Flow Label Specification," is a new specification of the Flow Label. A *flow* is defined as a sequence of packets from a sender to a specific unicast, anycast, or multicast address labeled as a flow by the sender. A flow is not necessarily associated with a transport connection. A host running multiple sessions with another host should be able to assign a different flow label to each session. Where the original specification defines a flow based on five criteria, the new specification defines a flow based on three criteria (Source and Destination address and Flow Label). The reason for this is that these three fields are always available for examination by routers, whereas the source and destination port number can be hidden by ESP.

The use of the Flow Label. The Flow Label is the only new field in the IPv6 header. 20 Bits are set aside, but in practice it is not used. There was a lot of debate in the IETF about the best use of this label and, partially through these uncertainties and other more pressing priorities, it has been ignored by most vendors. RFC 6294, "Survey of Proposed Use Cases for the IPv6 Flow Label," discusses the variety of proposals that have been published, and whether they are compatible with the existing standard.

In the meantime two specifications were published that offer alternative uses. RFC 7098, "Using the IPv6 Flow Label for Load Balancing in Server Farms," describes how the flow label can be used for load balancing and how it can enhance layer 3/4 load balancers. RFC 6438, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels," describes how to use the flow label for load balancing by equal cost multipath routing and for link aggregation, particularly for IP-in-IPv6 tunneled traffic.

IPv6 Extension header

The Hop-By-Hop Options header can be used to transport a maximum of one router alert signaling message per IP packet (RFC 2711) to every router on the path of QoS-sensitive traffic, indicating that each router should specifically process the IP packet. The use of the Hop-By-Hop Options header allows fast processing by the router because no analysis of higher-level protocol headers is required. Routers that are unable to recognize the router alert option type are required to ignore this option and continue processing the header. Also, routers are not allowed to change the option while the packet is in transit. Router alert types that have been defined so far are shown in Table 5-2.

Table 5-2. Currently defined router types

Value	Description
0	IP packet contains a Multicast Listener Discovery message.
1	IP packet contains an RSVP message.
2	IP packet contains an Active Networks message—the sender is attempting to load a program into the router for executing customized functions.
3–35	IP packet contains an Aggregated Reservation Nesting Level (RFC 3175, RSVP).
36–65,535	Reserved to IANA for future use.

A detailed description of these headers can be found in Chapter 3. An updated list of router alert types can be found at *http://www.iana.org/assignments/ipv6-routeralert-values*.

Provisioning

Two main network services are indispensable when it comes to operating an IP network, DHCPv6 for addressing systems and DNS for locating services. I expect DHCPv6 to be widely used in IPv6 enterprise networks, even though IPv6 provides SLAAC. The reason is that most organizations want to have the possibility to log and account for address use. This is not easily doable when using SLAAC. And when running IPv6 networks with the long addresses and even more while operating dual-stack networks and

expecting applications running over both protocol versions to be accessible for all users, DNS becomes more important than ever.

DHCP

DHCP is widely used to configure hosts with their IPv4 addresses and additional information. If you have an IPv6 network, you do not need DHCP to configure your hosts with address information. The Stateless Address Autoconfiguration mechanism (SLAAC) will configure your hosts for their IPv6 addresses without the need to set up a DHCP server. All you need to do is configure your IPv6-enabled routers with the prefix information for the links to which they are attached. But you might still choose to have DHCP servers in many cases. Host configuration that includes the assignment of IPv6 addresses using DHCP is called *Stateful Address Autoconfiguration* or *Stateful DHCPv6*. Maybe you have a specific IPv6 addressing scheme; or you need dynamic assignment of DNS servers; or you wish to implement dynamic updates to DNS (RFC 2136); or you need traceability-reporting features for the use of your IP addresses. In these cases, you can use DHCP for address configuration. You can also combine SLAAC and DHCPv6 configuration by using SLAAC for the IPv6 address configuration and DHCP servers to provide additional configuration information including DNS server IP addresses, DNS domains, or other DHCPv6 options.

RFC 3736 offers an additional configuration option. It defines a Stateless DHCP service for IPv6. A Stateless DHCP server can configure hosts that already have an IP address with additional information such as DNS or SIP servers. It cannot do address assignment, though. Stateless DHCP is explained later in this chapter, after this section on Stateful DHCPv6.

DHCPv6 and DHCPv4 are independent. If you want to configure hosts with DHCP in a dual-stack network, currently you will need two separate DHCP services running, one for each protocol. In this case, you will also have to watch out for configuration conflicts. In the DHCPv4 world, the client is configured to know whether to use DHCP. In the DHCPv6 world, the Router Advertisement has options to inform the client whether to use DHCP. There may be differing configuration information arriving at the client from different sources, or a node may have multiple interfaces, e.g., one being IPv4-only and one being dual-stacked. DHCPv6 uses a unique identifier (DUID), which does not exist for DHCPv4. In the realm of DHCPv4, MAC address and client ID resemble the DUID in DHCPv6 but are not synonymous. There is RFC 4361, which makes the DUID available for DHCPv4.

In RFC 4477, the DHCP working group is further assessing requirements and evaluating solutions, which will allow dual-stack hosts to be configured for both protocols by one or more DHCP server. The RFC describes issues identified with dual IP version DHCP interactions. The most important aspect is how to handle potential problems in clients processing configuration information received from both DHCPv4 and DHCPv6

servers. It includes a possible solution that would be to specify IPv4 options for DHCPv6 servers so that in a dual-stack environment you could run a DHCPv6 server and have it also configure IPv4 options for the dual-stack clients. In such a scenario, having DUIDs for DHCPv4 would be helpful.

DHCPv6 is specified in RFC 3315. All references in this chapter relate to DHCPv6. For the development of DHCPv6, the following guidelines were originally defined:

- It must be possible to combine DHCP and SLAAC.
- The configuration of DHCP and the interaction with other mechanisms (e.g., SLAAC) are the responsibility of the administrator.
- The clients do not need to be configured manually.
- DHCP must be able to configure multiple addresses per interface.
- A DHCP server is not needed in every subnet. Relay agents must be able to forward DHCP packets.
- A client must be able to deal with multiple DHCP replies coming back from different DHCP servers.
- It must be possible to have subnets where only some of the clients are configured by DHCP.
- DHCP must be able to do dynamic DNS updates to register allocated addresses in DNS. The administrator can decide to update DNS manually.
- DHCP must support and simplify the renumbering of a network.

The DHCPv6 specification includes authentication for DHCPv6 messages, which must be supported on the DHCPv6 client and server. Refer to the respective section on DHCPv6 authentication in this chapter.

DHCP Terms

Let us define some common terms used for DHCPv6:

DHCP Client
: A DHCP client sends requests to a DHCP server to get configuration information.

DHCP Server
: A DHCP server is preconfigured to reply to client requests. It knows the configuration for each client. When it receives a client request, it sends the information back to the client. A DHCP server may or may not be on the same link as the client.

DHCP Relay Agent
: If there is no DHCP server on the client link, a relay agent must be configured on the client link. The relay agent receives the client request and forwards it to one or

more DHCP server(s) on another subnet. When the relay agent receives the answer from the DHCP server, it forwards it to the client.

DHCP Unique Identifier (DUID)
: Each DHCP client and server has a DUID. DHCP servers use DUIDs to identify clients for the selection of configuration parameters and in the association of IAs (see below) with clients. DHCP clients use DUIDs to identify a server in messages where a server needs to be identified.

Identity Association (IA)
: A collection of addresses assigned to a client. Each IA has an associated Identity Association Identifier (IAID), which is assigned by the client. A client can have multiple IAs—for example, one for each interface.

Identity Association Identifier (IAID)
: An identifier for an IA chosen by the client. Each IA has an IAID, which is chosen to be unique among all IAIDs for IAs belonging to that client.

Transaction ID
: A value used to match requests and replies.

DHCP uses the following multicast addresses:

All_DHCP_Relay_Agents_and_Servers (`ff02::1:2`)
: All DHCP agents (servers and relays) are members of this multicast group. DHCP clients use this link-scoped multicast address to reach DHCP agents on their link. So clients do not need to know the agent's link-local address.

All_DHCP_Servers address (`ff05::1:3`)
: All DHCP servers within a site are members of this multicast group. This site-scoped address is used by DHCP relays to reach all DHCP servers within a site. They either do not know the server's unicast address, or they want to reach all DHCP servers within the site.

The following UDP ports are used with DHCPv6:

UDP port 546—Client port
: Clients listen on port 546 for DHCP messages. DHCP servers and relays use it as the destination port to reach DHCP clients.

UDP port 547—Server/Agent por
: DHCP servers and relays listen on port 547 for DHCP messages. DHCP clients use this port as the destination port to reach DHCP servers and relay agents. DHCP relays use this port as the destination port to reach DHCP servers.

The message types shown in Table 5-3 have been specified in RFC 3315.

Table 5-3. DHCPv6 message types

Message type	Description
SOLICIT (1)	Used by clients to locate DHCP servers.
ADVERTISE (2)	Used by servers as a response to Solicit.
REQUEST (3)	Used by clients to get information from servers.
CONFIRM (4)	Used by clients to verify that their address and configuration parameters are still valid for their link.
RENEW (5)	Used by clients to extend the lifetime of their IP address and renew their configuration parameters with their original DHCP server when their lease is about to expire.
REBIND (6)	Used by clients to extend the lifetime of their address(es) and renew their configuration parameters with any DHCP server when their lease is about to expire and they have not received a reply to their Renew message.
REPLY (7)	Used by DHCP servers to respond to Solicit messages with a Rapid Commit Option, as well as to Request, Renew, and Rebind messages. A Reply to an Information Request message contains only configuration parameters, but no IP address. A Reply to a Confirm message contains a confirmation that the client's IP address(es) are still valid for the link (or a Decline). A server sends a Reply as an acknowledgment for a Release or Decline message.
RELEASE (8)	Used by clients to release their IP address. The message is sent to the server from which the address was received.
DECLINE (9)	Used by clients to indicate to the server that one or more addresses assigned to them are already in use on the link. This is determined by the client through Duplicate Address Detection (DAD).
RECONFIGURE (10)	Used by DHCP servers to inform clients that the server has new or updated configuration information. The clients then must initiate a Renew or Information Request message in order to obtain the updated information.
INFORMATION REQUEST (11)	Sent by clients to request additional configuration parameters (without IP address information).
RELAY-FORW (12)	Used by DHCP relays to forward client messages to servers. The relay encapsulates the client message in an option in the Relay Forward message. The message can be sent directly to a DHCP server or via other relay agents. If a DHCP message is relayed multiple times, it is encapsulated multiple times.
RELAY-REPL (13)	Used by DHCP servers to send messages to clients through a relay. The client message is encapsulated as an option in the Relay Reply message. The relay decapsulates the message and forwards it to the client. The Relay Reply message takes the same path back through which the Relay Forward message traveled and may therefore also be encapsulated multiple times if there is more than one relay agent on the path.

The DHCP server-initiated configuration exchange is a great new feature. It can be used, for example, when links in the DHCP domain have to be renumbered or when new services or applications have been added and need to be configured on the clients. When services or applications need to be configured on the client, the DHCP server sends out a Reconfigure message (type 10) to the unicast address of each client. A client receiving this message must initiate a Renew or Information Request message exchange to get the updated information. Haven't we been waiting for this? This is an IPv6 implementation feature that solves a long-standing problem we had with DHCPv4. It can be done with DHCPv4, but it has rarely been implemented. The IPv4 way of doing this is defined in

RFC 3203. A DHCPv4 server sends a DHCPforcerenew message, which triggers the client to the Renew state in which it tries to renew its lease.

DHCPv6 header format

The general DHCPv6 header format is much simpler than the one used with DHCPv4. I describe it next.

Client-server messages. All DHCP messages exchanged between server and client have a fixed header with a variable part for options.

Figure 5-8 shows the header format.

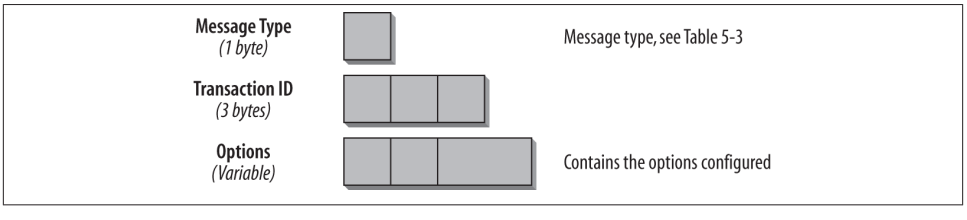

Figure 5-8. Format of the DHCP header

The Message Type field defines the type of message. You saw the list of message types in Table 5-3. For each request, the client generates a new transaction ID and writes it into the Transaction ID field. It is used in all messages relating to this specific request. When troubleshooting DHCP, it is important to check the transaction ID and make sure to associate the corresponding requests and replies.

Options are used to provide configuration information and parameters. The options fields have an identical base format, which is shown in Figure 5-9.

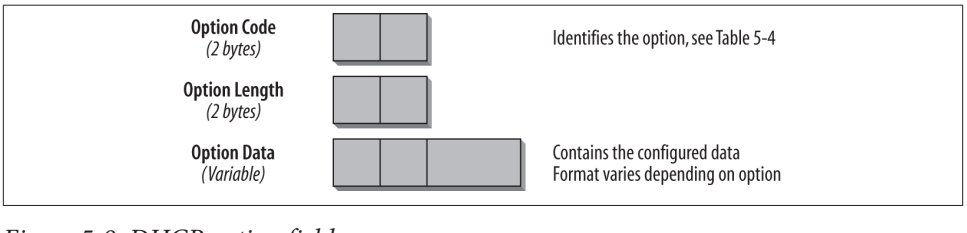

Figure 5-9. DHCP option fields

The *Option Code field* defines the type of the option. Find an overview of the available option types in Table 5-4. The *Option Length field* indicates the length of the option in bytes. The *Option Data field* finally contains the information configured for the option. Its format and length varies depending on the option type.

The options defined in RFC 3315 are a base set of options. In the future, additional options will be defined and specified in separate RFCs. Table 5-4 shows an overview.

Table 5-4. DHCP options

Option	Value	Description
Client Identifier	1	Used for the client DUID. A DUID is a unique identifier (described later in this chapter).
Server Identifier	2	Used for the server DUID.
Identity Association for Nontemporary Addresses (IA_NA)	3	Used to indicate the IA_NA, the parameters, and the nontemporary addresses associated with it.
Identity Association for Temporary Addresses (IA_TA)	4	Used to indicate the IA_TA, the parameters, and the temporary addresses associated to it. All addresses contained in this option are used as temporary addresses by the client (according to RFC 3041, "Privacy Extensions for Stateless Address Autoconfiguration").
IA Address	5	Used to indicate the addresses associated with an IA_NA or IA_TA.
Option Request	6	Used in a message between client and server to identify a list of options. Can be contained in a Request, Renew, Rebind, Confirm, or Information Request message. The server can use this option in a Reconfigure message to indicate which options have been changed or added.
Preference	7	Sent by the server to influence the choice of a client for a DHCP server.
Elapsed Time	8	Contains the time when the client started the DHCP transaction. Indicated in hundredths of a second. In the first message sent by a client it is set to `0`. Can be used by a secondary DHCP server to detect whether a primary server responds in time.
Relay Message	9	Contains the original message in a Relay Forward or Relay Reply message (remember that the original message is encapsulated in a Relay Forward or Reply message).
Authentication	11	Contains information to authenticate the identity and the content of DHCP messages.
Server Unicast	12	The server sends this option to the client to indicate that unicast can be used for communication. The option contains the IP address of the DHCP server, which is to be used by the client.

You can find an updated list of all defined DHCPv6 options at *http://bit.ly/1na92Wj*. For general DHCP information refer to the DHCP working group at *http://www.ietf.org/html.charters/dhc-charter.html*.

Relay Agent—server message format

Relay Agents forward client and server messages if the two are not on the same link. A DHCP message can be forwarded by more than one Relay Agent to one or more server(s). The reply by the server has to follow the same path back through which the original request came in, and it has to be forwarded by the same Relay Agents. Figure 5-10 shows the header fields in the Relay Agent and server messages.

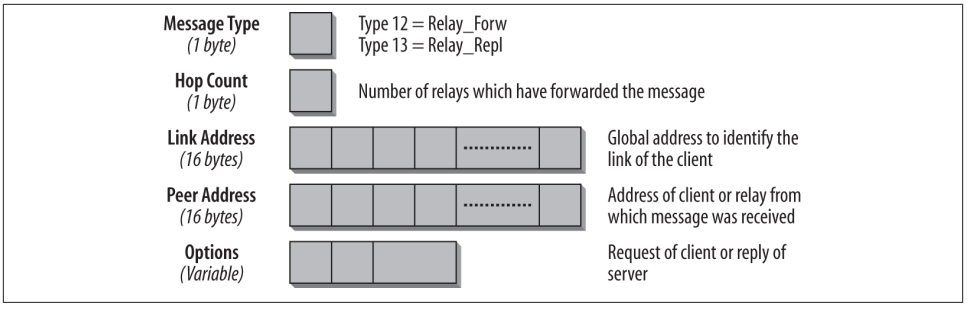

Figure 5-10. Header fields in Relay Agent and server messages

Relay Forward and Relay Reply messages have the same format and are identified by the value in the *Message Type field*. Type 12 is a Relay Forward message, type 13 a Relay Reply message.

The *Hop Count field* in a Relay Forward message shows how many Relays have already forwarded this message. Each forwarding Relay increases the value by one. The Relay can be preconfigured with a Hop Count Limit to limit the number of Relays that forward the message. When a Relay receives a message in which the Hop Count has reached the value configured in the Hop Count Limit, it discards the message. The default value for the Hop Count Limit is 32. In a Relay Reply message, the Hop Count field value is taken from the Hop Count field in the corresponding Relay Forward message.

The *Link Address field* contains a global IPv6 address. Based on this field in a Relay Forward message, the server can identify the link where the requesting client sits. The RFC also mentions the site-local address as a possible value for this field because the DHCPv6 RFC was published before the site-local address was deprecated. In a Relay Reply message, the value in this field is taken from the corresponding Relay Forward message.

The *Peer Address field* contains the address of the client or the Relay from which the message was received. This field is copied from the Relay Forward message into the corresponding field in the Relay Reply message.

The variable size *Options field* contains a Relay Message Option (option type 9). In a Relay Forward message, it contains the client request; in a Relay Reply message, it contains the server reply.

This field can also contain additional information that can be preconfigured on Relay Agents and that they insert when forwarding the message. This is specified in RFC 6422, "Relay-Supplied DHCP Options." It is described in the section "Relay Agent communication" on page 165.

DHCP Unique Identifier (DUID)

Each DHCP client and server has a DHCP Unique Identifier (DUID) that is used to identify each other. A server uses the client DUID to choose the corresponding client configuration to be sent. The DUID has to be unique across all servers and clients and should not be changed after initial assignment. RFC 3315 specifies three different types of DUIDs. Additional types may be specified in the future. A DUID contains a 2-byte type code followed by a variable number of bytes containing the identifier. The three types specified currently are as follows:

- Link-layer address plus time (DUID-LLT)
- Vendor-specific unique ID based on enterprise number (DUID-EN)
- Link-layer address (DUID-LL)

Identity Association

An *Identity Association* (IA) is an object used by the server and the client to identify and manage a group of addresses. Each IA is identified by a corresponding IAID and contains individual configuration information. A client has at least one IA per interface, which is to be configured by a DHCP server. The client uses the IA to get the right configuration for the interface from the server. Each IA has to be associated to only one interface. It is the client that chooses the IAID, and it must be unique among all IAIDs for IAs belonging to that client. The configuration information of an IA contains one or more IPv6 addresses plus the T1/T2 timers (Renewal and Rebinding timers, explained in "Renew/Rebind" on page 164).

A DHCP server chooses the configuration information for the IA according to the address allocation policies defined by the administrator. It chooses the configuration based on the following criteria:

- The link to which the client is connected
- The DUID of the client
- Other information provided by the options from the client
- Other information taken from the options, which have been added by Relay Agents

DHCP communication

There are different processes in the DHCP communication. There is the client-server interaction and the forwarding of messages over Relay Agents. The following sections describe these processes in more detail. Many processes are similar to DHCPv4, differing only in IPv6-related adaptation. Other processes are new—for instance, the way messages are forwarded over Relay Agents.

Client and server communication. A client uses multicast Solicit messages to find a DHCP server. If a client wishes to contact a specific DHCP server, it uses the server DUID in a Server Identifier Option (option type 2). All DHCP servers will receive this message, but only the server specified by the DUID will reply. In some cases, a client can use a unicast address to reach a specific server. This is possible only if the server is configured to send a Server Unicast Option (option type 12) indicating that unicast communication is possible and stating the IP address to be used. In this case, it has to be considered that these unicast messages will not be forwarded over Relay Agents, so any additional configuration done on the Relay Agent will not be inserted into the unicast DHCP messages. If a DHCP server receives a unicast message from a client to which it has not sent the Unicast option, it replies with a Reply message containing the Status Code "use multicast" (option 13, code 5).

The client receives one or more Advertise messages in answer to its Solicit message. If it receives more than one, it applies the following criteria to choose a DHCP server:

- The message with the highest Server Preference value is preferred.
- If there are several messages with an equal Server Preference value, it chooses the one with the preferred configuration.
- The client may also choose a message with a lower Server Preference value if it contains more appropriate configuration parameters.

The list of servers and their corresponding Preference values are stored at the client. Should it not receive replies from its preferred DHCP server, it will choose the next one in the list. If a client does not receive an answer from a DHCP server within a certain amount of time, it either initiates a new Discovery process by sending out another Solicit message or ends the configuration and creates an error message.

In reply to the Advertise message, the client sends a Request message to one of the DHCP servers including its IA Option, its client DUID, and an Option Request option, which contains the desired DHCP options. The server replies with a Reply message containing the requested options. If the server received the Request forwarded by a Relay Agent in a Relay Forward message, it will reply with a Relay Reply message forwarded over the same Relay Agents like the incoming Request message. The server flags the addresses given out in the Reply message as allocated. If the client receives multiple Replies, it chooses the most appropriate one and uses these addresses. The addresses allocated by other servers through their Advertise messages remain allocated but are not used. They will be reused by the DHCP server when their lifetimes have expired.

The client has to perform Duplicate Address Detection (DAD) for each address allocated by the DHCP server.

For an explanation of DAD, refer to the section "Neighbor Discovery" on page 87 in Chapter 4.

A typical DHCP communication performed by a client that does Stateful Address Autoconfiguration looks as follows:

1. Client sends Solicit message.
2. Server(s) reply with Advertise message.
3. Client sends Request message to one server.
4. Server replies with Reply message.

This communication can be shortened to only two messages with the *Rapid Commit option*. In this case, the client sends a Solicit message with the Rapid Commit option included. The server replies with a Reply message that also contains the Rapid Commit option. If the client sent out a Solicit message with a Rapid Commit option, it will ignore any Replies that do not contain a Rapid Commit option. If the client does not receive any Reply including a Rapid Commit option, it may accept an incoming Advertise message and continue the regular configuration process. If a server receives a Solicit message with a Rapid Commit option and is not configured to use it, it replies with a regular Advertise message. While it is true that Rapid Commit offers a more efficient approach to address assignment by using only two messages, depending on the configuration and the number of DHCP servers, it could result in wasted address space or a situation where multiple DHCPv6 servers believe that they each assigned addresses to requesting clients. Once a DHCP server has allocated an address in a Reply message with a Rapid Commit option, it has to commit the IP address to the client. Obviously with the vast address space in a /64, this may not be a big concern.

A client uses Request, Renew, Rebind, Release, and Decline messages as necessary for the lifetime of its server-assigned addresses. If the client switches link or subnet (for instance, in a wireless network or after waking up from sleep mode), it has to initiate a Confirm/Reply exchange. It does this by sending its IAs and the corresponding addresses and options. If the client does not receive an answer to its Confirm message, it should continue to use the previously allocated addresses.

To release one or more of its addresses, a client sends a Release message, which contains the IA and the corresponding addresses and options. The server answers with a Reply. If the client does not receive a Reply, it sends another Release message. This is not possible in all cases—for instance, if the client is shutting down. If a DHCP server did not receive a Release message, it will reuse the addresses when their lifetimes have expired.

If a client notices that an allocated address is already in use (for instance, through DAD), it sends a Decline message to the server. This message contains a Transaction ID, the client identifier, the server identifier, and the address(es).

Renew/Rebind. If a client wants to refresh the lifetime of its valid and preferred addresses, it sends a Renew (type 5) message containing the IA Address option and the addresses corresponding to this IA. The server identifies the corresponding lifetimes and sends a Reply message to the client. Doing this may also add new addresses or remove old addresses by setting their lifetime to 0.

If a server receives a Renew message for an IA for which it has no entry, it replies with a Reply message setting the Status code to "no binding" (option 13, code 3). If the client wants to renew an address that is not valid for its link, the server sends a Reply message setting the lifetimes for the addresses to 0.

The server controls the intervals in which a client has to renew its addresses through the Timers T1 and T2 preconfigured and associated to each IA. When the client reaches the time indicated by T1, it has to start the Renew process. When a client reaches the time indicated by T2, this indicates that its Renew messages have not been answered. In this case, it sends a Rebind message to all DHCP servers. The Rebind message contains an IA option with the currently allocated addresses and an Option Request option with all desired DHCP options.

When a server receives a Rebind message and finds the corresponding IA, it answers with a Reply message. If the addresses are not valid for the link anymore, it sets the lifetimes to 0. If the client does not receive an answer to a Rebind message, it cannot make further use of the address(es). In this case, it has two options:

- Restart the address configuration by sending out a Solicit message to find a DHCP server.
- If the client has other valid IAs, it can ignore the expired IA and use other addresses.

Information Request. If the client already has IP addresses but wants to get other DHCP information, it sends an Information Request message. This message contains an Option Request option to indicate the desired DHCP options. If, for instance, the client is configured by Stateless Address Autoconfiguration and the router is configured to set the(((“O-Flag”))) O-Flag (other Stateful configuration) in the Router Advertisement, this causes the client to send an Information Request message to get additional information such as DNS, NTP, or SIP server configuration. The Information Request message is also sent by the client in answer to a Reconfigure message from the server.

Reconfigure process. The server sends a Reconfigure message to trigger the client to send a Renew or an Information Request message. This is useful when the server has been

updated with new or modified information, to make sure the new information is propagated as quickly as possible. In the Reconfigure message, the Transaction ID is set to 0 and contains a Server Identifier option including the server DUID and a Client Identifier option containing the client DUID. Additionally, an Option Request option can be sent along to indicate to the client which options have been changed or added. The Option Request option contains an IA Address option (type 5) if the client needs to reconfigure its IP address. With the Reconfigure message option (type 19), the server indicates whether the client has to send a Renew or an Information Request message.

Because of the danger of DoS attacks, the use of security mechanisms is mandatory in Reconfigure messages, which means that the server has to use DHCP authentication. The server sends a Reconfigure message to a unicast IPv6 address of each client. If it doesn't know the unicast address of the client, it sends the message as a Relay Reply message to a Relay Agent. While a client is in a Reconfigure process, it does not accept further Reconfigure messages. A new process can be started only once the initial process has been completed.

Relay Agent communication. The way a Relay Agent forwards DHCP messages with DHCPv6 is quite different from the way it is done with DHCPv4. The following section describes the Relay Agent communication in detail.

A Relay Agent uses the All_DHCP_Servers multicast address (`ff05::1:3`) to forward messages to DHCP servers. It can be configured to use a unicast address. The Relay Agent takes the message coming from the client and builds a Relay Forward message. Figure 5-10 shows the header of this message. In the Link Address field, it sets its global IPv6 address with the prefix for the link on which the client resides. From this address, the DHCP server determines for which prefix it has to allocate addresses. The Hop Count is set to 1. The Source address from the original address (i.e., the client IP address) is copied into the Peer Address field of the Relay Forward message. The original DHCP message is copied into the Relay Message Option field. The Relay Agent can now add other information that has been preconfigured by the administrator.

When a Relay Agent receives a Relay Forward message from another Relay Agent and the value of the Hop Count field reaches the preconfigured value for the Hop Count Limit, it ignores the message. With the Hop Count Limit, the number of Relay Agents that forward a DHCP message can be limited. If the Hop Count is smaller than the Hop Count Limit, the message is forwarded. It encapsulates the packet into another Relay Forward header, increases the Hop Count by one, and copies the Source address of the previous Relay Agent into the Peer Address field. The Link Address field is set to 0. The message received is copied into the Relay Message Option.

As already mentioned, the Relay Reply message has to be forwarded over the same Relay Agents as the Relay Forward message. With the process just described, each Relay Agent encapsulates the received message into a new Relay Forward header, which makes it

possible for the DHCP server to track the way back. In the last Relay Message Option, the server finds the original request from the client. It replies to it and copies the answer into the Relay Message Option of a Relay Reply message. It encapsulates this reply into as many Relay Reply headers as the Relay Forward message has received. So the Relay Reply travels the same way back through the same Relay Agents. Each Relay Agent on the path decapsulates the exterior header and forwards the message to the next Relay Agent. The last Relay Agent on the path receives a Relay Reply message, which contains the server reply in the Relay Message Option field. It removes the Relay Reply header and forwards the server reply to the client.

Table 5-5 shows the entries in the header fields for a packet that has been forwarded over two Relays, Relay A and Relay B.

Table 5-5. The headers in Relay Forward and Relay Reply messages

Header field	Packet 2	Packet 3	Packet 4	Packet 5
	Relay A to Relay B	Relay B to Server	Server to Relay B	Relay B to Relay A
Message Type	Relay Forward (type 12)	Relay Forward (type 12)	Relay Reply (type 13)	Relay Reply (type 13)
Hop Count	1	2	2	1
Link Address	Relay A	0	0	Relay A
Peer Address	Client C	Relay A	Relay A	Client C
Relay Message Option	Client Request	Packet 2	Packet 5	DHCP Reply

The communication looks as follows:

1. Client C sends a DHCP Request (packet 1, not shown in Table 5-5).
2. Relay Agent A forwards the client request in a Relay Forward message (type 12) to Relay Agent B (packet 2). It copies its address into the Link Address field. The client request is copied into the Relay Message Option.
3. Relay Agent B forwards the message to the DHCP server (packet 3). It sets the Link Address field to 0 and copies the address of Relay Agent A into the Peer Address field. Packet 2 received from Relay Agent A is copied into the Relay Message Option.
4. The DHCP server sends a Relay Reply (type 13) to Relay Agent B (packet 4). The Hop Count, Link Address, and Peer Address fields are copied from the Relay Forward message. The Relay Message Option contains the packet, which has to be sent from Relay Agent B to Relay Agent A (packet 5).
5. Relay Agent B decapsulates packet 5 and forwards it to Relay Agent A.
6. Relay Agent A takes the server reply from the Relay Message Option and forwards it to Client C.

As mentioned before, the Relay Agent may have information for the client. But it has no way to send that information to the client. RFC 6422 specifies a *Relay-Supplied Options option* (*RSOO*, option code 66). The Relay encapsulates these additional options in an RSOO. The DHCP server can then add those options to the DHCP reply that is sent to the client. These options must be specifically defined as an RSOO-enabled option with reference to RFC 6422. Options that have been defined before publication of RFC 6422 are not RSOO-enabled.

A list of RSOO-enabled options can be found at *http://bit.ly/1na9bZQ* (scroll way down).

DHCPv6 communication in the trace file. In this section I would like to show you a DHCPv6 trace file, captured in a classroom.

As described in Chapter 4, the client learns through a Router Advertisement (RA) that it needs to use DHCPv6 in order to get an IPv6 address. Figure 5-11 shows what the Router Advertisement looks like.

```
⊟ Internet Control Message Protocol v6
    Type: Router Advertisement (134)
    Code: 0
    Checksum: 0x7710 [correct]
    Cur hop limit: 64
  ⊟ Flags: 0x80
      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
```

Figure 5-11. DHCPv6 flags in the Router Advertisement

During the boot process, the client sends out a Router Solicitation. If there is an IPv6 router and it has been accordingly configured,(((“O-Flag”))) the client will get a Router Advertisement with one or both DHCPv6 flags set to 1, the *Managed Configuration flag* (*M-Flag*) and the *Other Stateful Configuration flag* (*O-Flag*). When the client gets this it will initiate a DHCP request.

Figure 5-12 shows the communication.

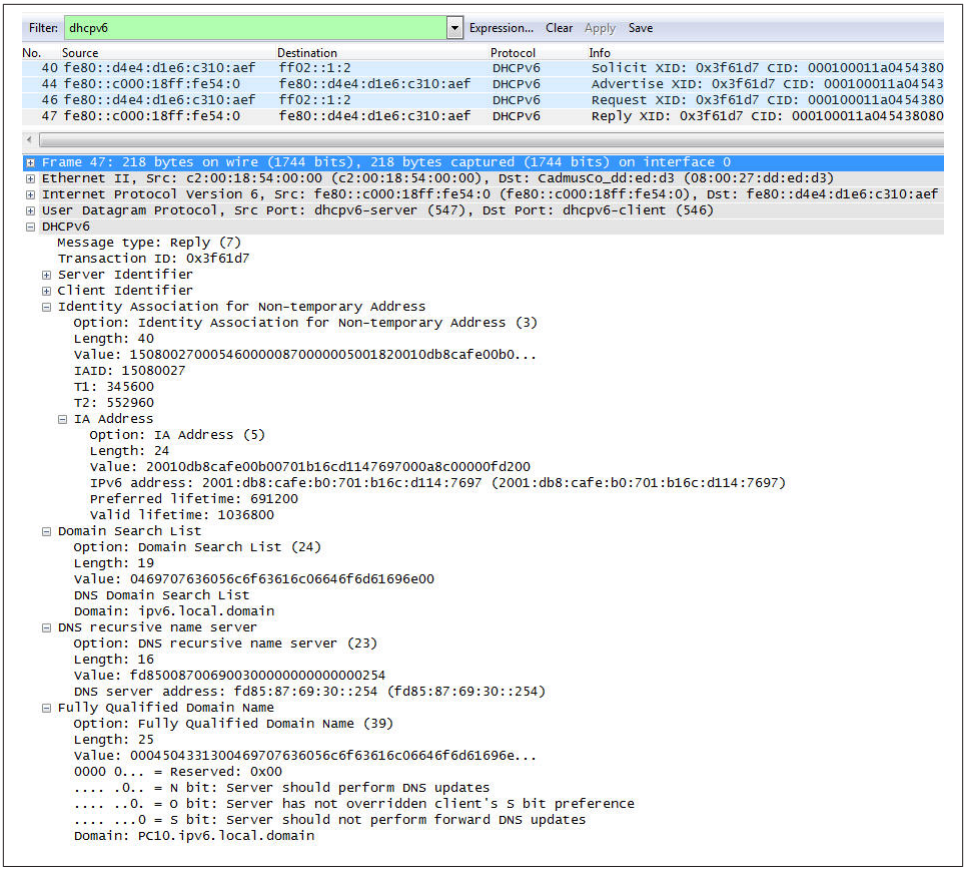

Figure 5-12. DHCPv6 communication in the trace file

As you can see in the uppermost part of the figure, I set a filter in Wireshark to only display DHCPv6 communication. The summary line therefore shows the four DHCPv6 packets described before, Solicit, Advertise, Request, and Reply. Packet number 47 is marked and the details of this reply packet are displayed in the lower part of the figure.

Let me shortly describe the three packets of which you cannot see the details in the screenshot:

- In packet number 40 we see the Solicit message. The source address is the link-local address of the client and it goes to the well-known multicast address for All_DHCP_Relay_Agents_and_Servers (`ff02::1:2`). UDP source port is 546 and UDP destination port is 547.

- Packet number 44 is the Advertise message of the DHCP server. The source address is the DHCP server's address and the packet is sent to the link-local address of the client.
- In packet 46, the client sends its DHCPv6 request to the multicast address `ff02::1:2`.

Packet number 47 is the Reply message of the DHCP server and this is the packet of which you can see the details in the screenshot. It is sent from the DHCP server's address to the link-local address of the client. The UDP source port is 547 (server port) and the UDP destination port is 546 (client port). The first field in the DHCPv6 header shows the message type (7 for Reply, refer to Table 5-4 for all option type numbers). The next field contains the transaction ID. For a given DHCP communication, the transaction ID must be the same (important to watch when troubleshooting). The Reply message contains the Server Identifier (option 2), Client Identifier (option type 1), Identity Association (option type 3) including the two timers T1 and T2, the IA address option (option type 5) including the IPv6 address and the lifetime parameters, the Domain Search List (option type 24), DNS recursive name server (option type 23), and the fully qualified domain name (FQDN, option type 39).

Stateless DHCP

In environments where Stateless Address Autoconfiguration is used for IP address information, there was no way to configure additional information on the client, such as DNS information or other options. Several solutions were discussed, one being to add such options to the Router Advertisement. Finally, RFC 3736 specified a new service called Stateless DHCP Service for IPv6. A Stateless DHCP server has an implementation of only a subset of the DHCPv6 specification. Its use requires that hosts are already configured for an IPv6 address.

A Stateless DHCP server replies to Information Request messages (message type 11) that contain an Option Request option (option type 6) with a Reply message (message type 7). The Stateless DHCP server can also act as a Relay Agent. This allows configuration of a part of the clients on a link using SLAAC while getting additional information from the Stateless DHCP server. Meanwhile, other clients use Stateful Address Autoconfiguration, and their DHCP messages are forwarded by the Stateless DHCP server acting as a Relay Agent.

Finally both options were realized. Stateless DHCPv6 was developed and RFC 6106 defines "Router Advertisement Options for Recursive DNS Server and DNS Search List." It must be implemented on routers as well as on the client side in order to work. Currently, Microsoft is not supporting it on Windows. For scenarios such as ad hoc networks, this can be a good option to configure clients without a stateful or stateless DHCPv6 server. In a corporate environment, a DHCPv6 server may be chosen in most

cases, because of the need for traceability and also in order to configure other additional options.

Prefix Delegation

RFC 3633, "IPv6 Prefix Options," defines options that can be used to send prefix information from a delegating router or a DHCPv6 server to a requesting router that has DHCPv6 client functionality. It is useful in environments where the delegating router has no information about the topology of a network connected to the requesting router. A delegating router or a DHCPv6 server in an ISP network uses this option to configure a router in a customer network for its prefix. The delegating router can, for example, assign a /48 prefix to the border router in the customer network. The border router can subdivide the /48 prefix to /64 subnets and advertise these prefixes with Router Advertisements. DHCP Prefix Delegation (DHCP-PD) is independent from DHCP address assignment, but the two can be combined.

RFC 6603, "Prefix Exclude Option for DHCPv6-based Prefix Delegation," updates RFC 3633. The prefix exclusion mechanism is defined for deployments where DHCPv6-based prefix delegation is used, where a single aggregated route/prefix has to represent one customer, instead of using one prefix for the link between the delegating router and the requesting router and another prefix for the customer network. This mechanism allows a delegating router to use a prefix out of the delegated prefix set on the link through which it exchanges DHCPv6 messages with the requesting router. It is intended for use in networks where each requesting router is on its own Layer 2 domain.

Security considerations

Attacks based on DHCP functionality are possible in the IPv4 world as well as in the IPv6 world. The points of attack to be watched are the same:

- External, unknown DHCP servers allocating false addresses to DHCP clients
- Faulty or malicious DHCP servers in the intranet that assign false addresses or other false configuration information to DHCP clients
- Unknown external clients that attach to the corporate network and receive internal addresses
- Intentional exhaustion of IP addresses by malicious clients, resulting in valid clients being unable to obtain a valid IP address and/or configuration options
- Malicious client(s) transmitting such high volumes of requests that a DHCP server is unable to respond to valid requests

To protect your network from external DHCP servers from outside the corporate network, a firewall closing the ports for DHCP is a good protection. It is important to protect your network from internal DHCP servers. It doesn't even need to be a malicious

attack. Very often the problems come from improperly configured test servers. A client can be attacked by a malicious DHCP server configuring it with false information. For instance, a bad DNS or NTP server can be configured, or it can be configured in a way that it cannot communicate in the local network anymore. To protect from such attacks, Authentication should be used (see below). Another protection is to use DHCP Guard, which is described in the section "First-hop security" on page 204 in Chapter 6.

With DHCPv4, the ways to protect from such attacks are limited. Firewalls only protect from outside attacks. The possibility to use Authentication for DHCP communication exists only in the form of vendor solutions in addition to DHCPv4.

The specification for DHCPv6 includes an Authentication mechanism, which is based on Authentication for DHCPv4 (RFC 3118). New hosts must be authorized and authenticated before they receive configuration information from a DHCP server, the sender of a message must be authenticated, and the content of the message must be protected.

The following section gives an overview of the Authentication mechanisms specified in RFC 3315. If you are not familiar with security concepts and terms, please refer to Chapter 6 first.

Security for messages between Relay Agents and DHCP servers. For a secure exchange of messages between Relay Agents and DHCP servers, IPsec (in transport mode with ESP) is used. Between each Relay Agent and its communication peers, an independent two-way trust relationship has to be established. If the content of the message is not considered confidential, encryption is not required (null encryption). As the Relay Agents and the DHCP servers are within the corporate network, private keys can be used.

In addition to this, DHCP servers and Relay Agents are configured with the addresses of trusted communication peers. It is therefore not possible for an unknown DHCP server or Relay Agent to intrude into the communication.

DHCP Authentication. The authentication of DHCP messages can be accomplished through the use of the Authentication option (option 11). The authentication information carried in the Authentication option can be used to reliably identify the source of a DHCP message and to confirm that the contents of the DHCP message have not been changed.

Figure 5-13 shows the format of the Authentication option.

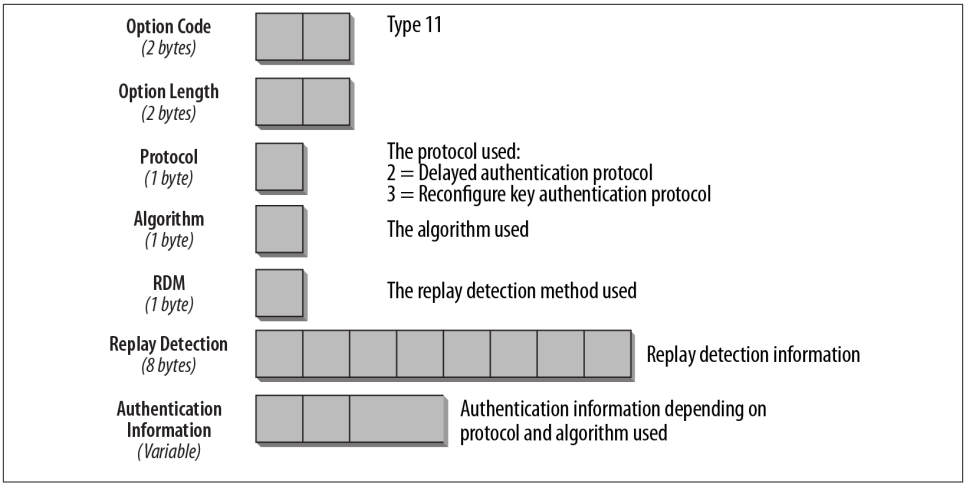

Figure 5-13. The format of the Authentication option

Multiple authentication protocols can be used with the Authentication option. Two such protocols are specified in RFC 3315: the Delayed Authentication Protocol and the Reconfigure Key Authentication Protocol (section 21 in RFC 3315). If the Delayed Authentication Protocol is chosen, Protocol number 2 is used. If the Reconfigure Key Authentication Protocol is chosen, Protocol number 3 is used. Additional protocols may be specified in the future with separate RFCs.

Unfortunately, there is limited support for DHCPv6 authentication. Specifically, Microsoft does not support it on Windows.

Further development

IP address management is an important aspect of efficient network management. There is intense work going on in the IETF DHCP working group to optimize DHCP. The DHCP working group is the best place to find updated information not only on the current status of DHCP (for IPv4 and IPv6) but also on possible future developments.

The DHCP working group can be found at *http://datatracker.ietf.org/wg/dhc*.

A special focus is on how dual-stack networks or IPv6-only networks can best be managed. Currently, there is DHCPv4 for the configuration of IPv4 interfaces and DHCPv6 for the configuration for IPv6 interfaces. So in a dual-stack network we would need both, a DHCPv4 and a DHCPv6 server. Special care has to be taken that there is not

overlapping and contradicting configuration coming from each DHCP server. RFC 4477, "Dynamic Host Configuration Protocol (DHCP): IPv4 and IPv6 Dual-Stack Issues," discusses these challenges. Another scenario that is becoming more and more important is IPv6-only networks. Even in those cases, where IPv4 might be treated as a service in an IPv6-only network, hosts may need some IPv4 configuration information.

Currently the working group is working on drafts that would provide DHCPv4 information over IPv6 transport or encapsulated in DHCPv6 messages. One draft describes the definition of IPv4 options for DHCPv6 servers. The draft called "Provisioning IPv4 Configuration Over IPv6 Only Networks" provides an overview of the use cases and a summary and discussion of the possible solutions. See the draft reference list at the end of this chapter for a list of the current drafts.

Dynamic updates to DNS

With the widespread use of DHCP and autoconfiguration for dynamic IP address configuration, the need for a dynamic update of DNS for addition and deletion of records arose. RFC 2136 introduced the mechanism called Dynamic DNS (DDNS). It is supported since BIND versions 8 and 9 and many popular DNS implementations. The update functionality is usually used by applications such as DHCP, but it can be implemented on hosts as well. With IPv6, dynamic addresses are often assigned using Stateless Address Autoconfiguration, which means there may not be a DHCP server in the network. A DNS update mechanism is necessary on each host to update its DNS records. There are important security aspects to consider when DDNS updates are made. It is important that you can control which nodes are authorized to make changes to your DNS records. Update policies must be implemented and Transaction Signatures (TSIG; see RFC 2845) or Domain Name System Security Extensions (DNSSEC; see RFCs 3007, 4033, 4034, and 4035) mechanisms should be used. RFC 4339, "IPv6 Host Configuration of DNS Server Information Approaches," discusses some of these general DNS aspects for IPv6 hosts.

For hosts that are configured through DHCPv6, RFC 4704 defines an client FQDN (Fully Qualified Domain Name) option that can be added to the Solicit, Request, Renew or Rebind messages. This allows the DHCPv6 client or server to update the DNS accordingly.

For hosts that are configured with SLAAC, a similar mechanism is about to be defined. A draft is under way called "Registering self-generated IPv6 Addresses in DNS using DHCPv6." It defines a new DHCPv6 message type that can be used by clients to request a DHCPv6 server to add FQDN options to DNS dynamically.

DNS

DNS is used in the IPv4 world to do name-to-address mappings and vice versa. This is not changing in the IPv6 world. The need for DNS is actually much greater because of

the length of IPv6 addresses. Mixed IPv4/IPv6 environments need multiple host entries in DNS. A host communicating with both versions of TCP/IP needs at least two entries in DNS—one with its IPv4 address and the other with its IPv6 address. A new DNS record type has been defined for IPv6 hosts. RFC 3596 defines the AAAA type record (called *Quad-A*). RFC 2874 defines the A6 type record, which was designed to make renumbering of networks and prefix changes easier to administer. A6 has been moved to experimental status and is not used. The other DNS record types (NS and PTR records) remain unchanged, adjusting only to support the IPv6 address format.

AAAA records and IP6.ARPA

RFC 3596 describes DNS extensions for IPv6 implementations based on AAAA records. This record type can store an 128-bit IPv6 address, and the DNS value for this type of record is 28 (decimal notation). A host that has more than one IPv6 address has a AAAA record for each address. The corresponding reverse lookup domain is `IP6.ARPA`. The reverse lookup records are PTR records of type 12.

A AAAA type record can look like this:

```
moon.universe.com.   IN   AAAA   2001:db8:1:2:3:4:567:89ab
```

For reverse lookups, each subdomain level under `IP6.ARPA` represents 4 bits of the 128-bit address. The least significant bit appears at the far left of the domain name. Omitting leading zeros is not allowed in this case, so the PTR record for the previous example looks like this:

```
b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA.
IN  PTR   moon.universe.com.
```

Note that there are several ways to represent reverse IPv6 addresses in DNS. It depends on the implementation, so refer to your vendor's documentation to find out which format is expected.

Originally, the reverse domain was called IP6.INT. It has been deprecated (RFC 4159) and replaced by IP6.ARPA.

DNS servers

BIND implements IPv6 DNS in versions of BIND 8.4 and higher and in BIND version 9.

DNS implementations based on these versions of BIND support IPv6. A good reference site for BIND is the Internet Systems Consortium home page. The same site has a list of vendor implementations based on BIND. There are also links to versions of BIND that run on different versions of the Microsoft operating system.

The most important file for configuring a name server on Unix is */etc/named.conf*. The file itself contains detailed information on how to configure it. To make name resolution work over IPv6, you need to add one important entry: `listen-on-v6 { any }`. This entry tells the name server to listen for IPv6 queries. Then update */var/named* with the entries for all IPv6 hosts.

The entries in our zone record file are shown in Figure 5-14.

```
$TTL 3h
$ORIGIN universe.com
@       IN      SOA     ford.universe.com. mail.universe.com.(
        20141017        ; Serial
        3h              ; Refresh
        1h              ; Retry
        1w              ; Expire
        1h )            ; Minimum

        universe.com    IN NS   ford.universe.com

        ford    IN A    192.168.0.99
                IN AAAA 2001:db8:cafe:b0:a43d:d55b:d1c:897a

        arthur  IN A    192.168.1.66
                IN AAAA 2001:db8:cafe:b1:a00:2751:b5d:2c99

        marvin  IN A    192.168.2.22
                IN AAAA 2001:db8:cafe:b2:0:0:0:4
```

Figure 5-14. The zone record file

For a detailed explanation of BIND and DNS configuration, refer to *DNS and BIND on IPv6*, by Cricket Liu (O'Reilly).

DNS resolvers and DNS design

Resolvers are the client part in DNS communication. The resolver sends out DNS requests for IP addresses to DNS servers. It can be part of an operating system or an application. DNS servers also have a resolver implemented to send out DNS requests to other DNS servers.

When a dual-stacked host queries a DNS server for a service name, for instance by entering an URL in the browser, the client will send out two DNS requests, one for an A record and another for a AAAA record. The DNS server may respond with either an A record, or a AAAA record or with both, depending on how it is configured. In case the client receives two addresses, it will, based on the default address selection rules (RFC 6724), prefer native IPv6 over IPv4.

Default Address Selection is discussed in Chapter 2.

Which transport is used for resolving a name with DNS is independent of the connection that is used. So for instance, Windows XP was not able to resolve DNS names over IPv6. A Windows XP client always needed a DNS server that it could reach over IPv4. But when the client is dual-stacked and the DNS server responds with a AAAA record, the Windows XP client could initiate a session over IPv6.

There is a problem if a client gets a AAAA record but does not have IPv6 connectivity or only a very poor one. IPv6 stacks are generally configured to prefer IPv6 over IPv4 by default, so the client will try to connect over IPv6. But because it does not have connectivity, there is a long waiting time until the client eventually reverts to using IPv4 (in case it got an A record for the same service). The user will in most cases not understand why it takes so long to access that website and will probably blame the website owner for it.

This is the reason why earlier (before IPv6 World Launch Day on June 6, 2012) large dual-stacked websites such as Google and Facebook did not give out AAAA records for their main domain. If you wanted to access Google or Facebook over IPv6 in those days, you had to use a v6-specific domain name such as *http://ipv6.google.com*. These large sites did not want to accept performance hits due to users connecting from networks with bad IPv6 Internet connectivity. If such a user experiences long timeouts while accessing Google, the user will think that Google has a bad performance, while the timeout comes from not being able to connect over IPv6 and waiting to connect over IPv4. These sites often used DNS whitelisting for ISPs with good IPv6 performance, where they could be sure that users coming from that ISP have no problems accessing the website over IPv6. So only if you were connected from such a provider would you get a AAAA record for *http://www.google.com*. World IPv6 Launch Day on June 6, 2012, showed that only a tiny fraction of users experienced problems. Since then many of these sites have enabled AAAA records for the main domain permanently. To find more information on World IPv6 Launch day, go to *http://www.worldipv6launch.org*. It shows the participants and information about measurements.

If you like to get an updated picture of the current status of deployment of IPv6 in the world, there are two sites you can refer to: Google Statistics and Cisco Statistics.

Happy Eyeballs. In order to improve user experience in a dual-stack Internet, a specification has been defined, which is called Happy Eyeballs. It is defined in RFC 6555.

When a client gets two addresses for a given service, an IPv4 and a native IPv6 address, it will by default connect to the IPv6 address by initiating a TCP handshake. If there is no reply to the request, the client will, after a long timeout, revert to using IPv4. This can happen if the IPv6 path from the client to the service is broken or very slow. With a Happy Eyeball implementation, the client will try both protocols and then use the faster one for the connection. There are several ways to do it. Figure 5-15 shows such an algorithm.

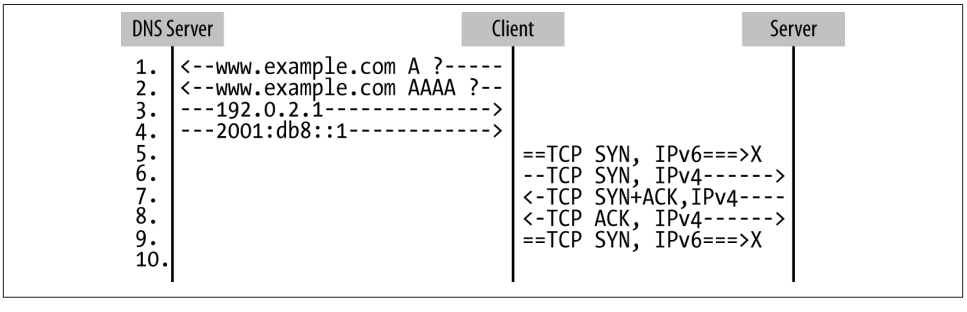

Figure 5-15. Happy Eyeballs with IPv6 connection broken

In this case, the client sends two requests to the DNS server for *http://www.example.com*, an A request and a AAAA request. The service is dual-stack so DNS returns two addresses, an IPv4 and an IPv6 address. Next we see the client issue two TCP Syn requests (the first packet in a TCP handshake), one over IPv6 and one over IPv4. In this case the IPv6 connection was broken, so the client only gets a reply (SYN+ACK) over IPv4. In the next packet the client confirms the TCP handshake with a TCP Ack and now the communication with the service will run over IPv4.

Different operating systems and browsers have different types of implementations of Happy Eyeballs. RFC 6555 describes the implementation in Google Chrome and Firefox as follows:

1. Call `getaddrinfo()`, which returns a list of IP addresses sorted by the host's address preference policy.
2. Initiate a connection attempt with the first address in that list (e.g., IPv6).
3. If that connection does not complete within a short period of time (Firefox and Chrome use 300 ms), initiate a connection attempt with the first address belonging to the other address family (e.g., IPv4).
4. The first connection that is established is used. The other connection is discarded.

Microsoft has not implemented Happy Eyeballs in Internet Explorer but it has a similar mechanism to optimize connection setup depending on protocol performance. Apple also has an OS-specific implementation similar to Happy Eyeballs.

Name space fragmentation. With regard to DNS design, there is an important point to be aware of: *name space fragmentation*. When a resolver tries to resolve a name it will start at the root and follow referrals until it reaches an authoritative name server for the name. If the resolver happens to reach a name server that is only reachable over a protocol that the resolver can't use, the name cannot be resolved, and so the DNS query is unsuccessful.

So when IPv6 starts to get deployed more and more in the Internet, the name space may get fragmented because there may be name servers that can only be reached over IPv4 and more and more name servers that may only be reached over IPv6. So we have to find mechanisms to avoid the situation where the resolver chain breaks due to two name servers in the resolution process that do not speak the same language (IP version).

Here are the DNS recommended guidelines to avoid this (quote from RFC 3901). In order to preserve name space continuity, the following administrative policies are recommended:

- Every recursive name server SHOULD be either IPv4-only or dual stack.

  This rules out IPv6-only recursive servers. However, one might design configurations where a chain of IPv6-only name server forward queries to a set of dual-stack recursive name servers actually performing those recursive queries.
- Every DNS zone SHOULD be served by at least one IPv4-reachable authoritative name server.

  This rules out DNS zones served only by IPv6-only authoritative name servers.

Note: zone validation processes SHOULD ensure that there is at least one IPv4 address record available for the name servers of any child delegations within the zone.

For your integration of IPv6 it is probably a good idea to plan for dual-stacked DNS services, as this is the best and most flexible protection of fragmentation. With BIND9 you can also configure a dual-stack server. When a recursive name server needs to look up data in a zone served only by a name server that doesn't speak the same protocol, it can forward a recursive query to the dual-stack server.

Make sure that you configure AAAA records only when the services are fully reachable over IPv6, otherwise your clients may experience long timeouts or not reach the service at all. We also have to move away from entering host names only in DNS. A host may be dual-stacked (and have two DNS entries), while some services running on that host may be either IPv4 or IPv6 services. In a large enterprise it could be recommendable

for administrative reasons to not mix IPv4 services and IPv6 services on dual-stacked hosts, but place IPv4 services on IPv4 hosts and IPv6 services on IPv6 hosts. At the same time you have to make sure that all clients that get AAAA records for IPv6-only services can connect over IPv6 to that network where the service is. If a service is available on both protocols, make sure the IPv6 service gets precedence over the IPv4 service, so your traffic can slowly shift to using IPv6 more and more whenever it is available.

DNS communication in the trace file

Figure 5-16 shows DNS queries and replies in the trace file.

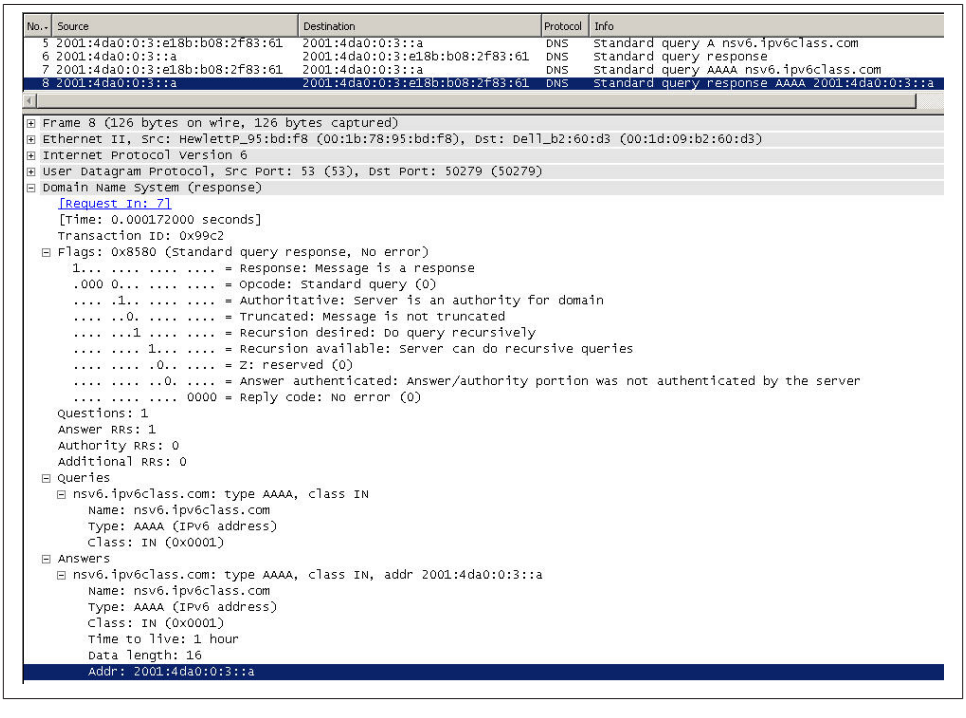

Figure 5-16. DNS communication in the trace file

A client issues DNS requests for *http://nsv6.ipv6class.com* to a DNS server. In packet 5, the client asks for an A record and receives a reply in packet 6 with no Answer Record. The requested service is an IPv6-only service and therefore has no A record. In packet 7, the client asks for a AAAA record and gets the Answer record in packet 8. The details of packet 8 are displayed in the lower part of the figure. The DNS transaction ID is `099c2` (both in Request and Reply). The flags are set for Response, Authoritative Server, Recursion desired, and Recursion available. Below that you can see the Query and the corresponding Answer Record with Time to Live and the IPv6 address.

The whole DNS communication goes over IPv6 in this case. As mentioned before, it would also be possible that the DNS resolution goes over IPv4 and then the connection to the IPv6-only service over IPv6.

The next chapter discusses Security with IPv6.

References

Here's a list of the most important RFCs and drafts mentioned in this chapter. Sometimes I list additional subject-related RFCs for your personal further study.

RFCs

- RFC 1195, "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments," 1990
- RFC 1321, "The MD5 Message Digest Algorithm," 1992
- RFC 2080, "RIPng" for IPv6," 1997
- RFC 2104, "HMAC: Keyed-Hashing for Message Authentication," 1997
- RFC 2136, "Dynamic Updates in the Domain Name System," 1997
- RFC 2149, "Multicast Server Architectures for MARS-based ATM multicasting," 1997
- RFC 2205, "Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification," 1997
- RFC 2210, "The Use of RSVP with IETF Integrated Services," 1997
- RFC 2324, "Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)," 1998
- RFC 2328, "OSPF Version 2," 1998
- RFC 2362, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," 1998
- RFC 2365, "Administratively Scoped IP Multicast," 1998
- RFC 2430, "A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)," 1998
- RFC 2453, "RIP Version 2," 1998
- RFC 2464, "Transmission of IPv6 Packets over Ethernet Networks," 1998
- RFC 2467, "Transmission of IPv6 Packets over FDDI Networks," 1998
- RFC 2474, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," 1998
- RFC 2475, "An Architecture for Differentiated Services," 1998
- RFC 2491, "IPv6 over Non-Broadcast Multiple Access (NBMA) networks," 1999

- RFC 2492, "IPv6 over ATM Networks," 1999
- RFC 2597, "Assured Forwarding PHB Group," 1999
- RFC 2590, "Transmission of IPv6 Packets over Frame Relay Networks Specification," 1999
- RFC 2710, "Multicast Listener Discovery (MLD)" for IPv6," 1999
- RFC 2715, "Interoperability Rules for Multicast Routing Protocols," 1999
- RFC 2845, "Secret Key Transaction Authentication for DNS (TSIG)," 2000
- RFC 2884, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks," 2000
- RFC 2894, "Router Renumbering for IPv6," 2000
- RFC 2914, "Congestion Control Principles," 2000
- RFC 2963, "A Rate Adaptive Shaper for Differentiated Services," 2000
- RFC 2983, "Differentiated Services and Tunnels," 2000
- RFC 2998, "A Framework for Integrated Services Operation over Diffserv Networks," 2000
- RFC 3006, "Integrated Services in the Presence of Compressible Flows," 2000
- RFC 3007, "Secure Domain Name System (DNS) Dynamic Update," 2000
- RFC 3008, "Domain Name System Security (DNSSEC) Signing Authority," 2000
- RFC 3086, "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification," 2001
- RFC 3118, "Authentication for DHCP Messages," 2001
- RFC 3124, "The Congestion Manager," 2001
- RFC 3140, "Per Hop Behavior Identification Codes," 2001
- RFC 3162, "Radius and IPv6," 2001
- RFC 3168, "The Addition of Explicit Congestion Notification (ECN) to IP," 2001
- RFC 3246, "An Expedited Forwarding PHB," 2002
- RFC 3247, "Supplemental Information for the New Definition of the EF PHB (Expedited Forwarding Per-Hop Behavior)," 2002
- RFC 3260, "New Terminology and Clarifications for Diffserv," 2002
- RFC 3289, "Management Information Base for the Differentiated Services Architecture," 2002
- RFC 3290, "An Informal Management Model for DiffServ Routers," 2002
- RFC 3306, "Unicast-Prefix-based IPv6 Multicast Addresses," 2002
- RFC 3307, "Allocation Guidelines for IPv6 Multicast Addresses," 2002

- RFC 3315, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," 2003
- RFC 3317, "Differentiated Services Quality of Service Policy Information Base," 2003
- RFC 3353, "Overview of IP Multicast in a Multi-Protocol Label Switching (MPLS) Environment," 2002
- RFC 3569, "An Overview of Source-Specific Multicast (SSM)," 2003
- RFC 3590, "Source Address Selection for the Multicast Listener Discovery (MLD) Protocol," 2003
- RFC 3596, "DNS Extensions to Support IP Version 6," 2003
- RFC 3633, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6," 2003
- RFC 3646, "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," 2003
- RFC 3717, "IP over Optical Networks: A Framework," 2004
- RFC 3736, "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6," 2004
- RFC 3810, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6," 2004
- RFC 3901, "DNS IPv6 Transport Operational Guidelines," 2004
- RFC 3956, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address," 2004
- RFC 3971, "SEcure Neighbor Discovery (SEND)," 2005
- RFC 3972, "Cryptographically Generated Addresses (CGA)," 2005
- RFC 3973, "Protocol Independent Multicast—Dense Mode (PIM-DM): Protocol Specification (Revised)," 2005
- RFC 4033, "DNS Security Introduction and Requirements," 2005
- RFC 4034, "Resource Records for the DNS Security Extensions," 2005
- RFC 4035, "Protocol Modifications for the DNS Security Extensions," 2005
- RFC 4074, "Common Misbehavior Against DNS Queries for IPv6 Addresses," 2005
- RFC 4076, "Renumbering Requirements for Stateless Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," 2005
- RFC 4094, "Analysis of Existing Quality-of-Service Signaling Protocols," 2005
- RFC 4135, "Goals of Detecting Network Attachment in IPv6," 2005
- RFC 4159, "Deprecation of "ip6.int," 2005
- RFC 4192, "Procedures for Renumbering an IPv6 Network without a Flag Day," 2005

- RFC 4243, "Vendor-Specific Information Suboption for the Dynamic Host Configuration Protocol (DHCP) Relay Agent Option," 2005
- RFC 4271, "A Border Gateway Protocol 4 (BGP-4)," 2006
- RFC 4282, "The Network Access Identifier," 2005
- RFC 4286, "Multicast Router Discovery (MRD)," 2005
- RFC 4338, "Transmission of IPv6, IPv4, and Address Resolution Protocol (ARP) Packets over Fibre Channel," 2006
- RFC 4339, "IPv6 Host Configuration of DNS Server Information Approaches," 2006
- RFC 4361, "Node-specific Client Identifiers for DHCPv4," 2006
- RFC 4472, "Operational Considerations and Issues with IPv6 DNS," 2006
- RFC 4477, "Dynamic Host Configuration Protocol (DHCP): IPv4 and IPv6 Dual-Stack Issues," 2006
- RFC 4489, "A Method for Generating Link-Scoped IPv6 Multicast Addresses," 2006
- RFC 4601, "Protocol Independent Multicast—Sparse Mode (PIM-SM): Protocol Specification (Revised)," 2006
- RFC 4604, "Using MLDv2 for Source Specific Multicast," 2006
- RFC 4703, "Resolution of Fully Qualified Domain Name (FQDN) Conflicts among Dynamic Host Configuration Protocol (DHCP) Clients," 2006
- RFC 4704, "The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option," 2006
- RFC 4760, "Multiprotocol Extensions for BGP-4," 2007
- RFC 4944, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," 2007
- RFC 4957, "Link-Layer Event Notifications for Detecting Network Attachments," 2007
- RFC 5015, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)," 2007
- RFC 5072, "IP Version 6 over PPP," 2007
- RFC 5172, "Negotiation for IPv6 Datagram Compression using IPv6 Control Protocol," 2008
- RFC 5308, "Routing IPv6 with IS-IS," 2008
- RFC 5340, "OSPF for IPv6," 2008
- RFC 5796, "Authentication and Confidentiality in PIM-SM Link-local Messages," 2010
- RFC 5838, "Support of Address Families in OSPFv3," 2010
- RFC 5887, "Renumbering Still Needs Work," 2010

- RFC 5942, "IPv6 Subnet Model: the Relationship between Links and Subnet Prefixes," 2012
- RFC 6085, "Address Mapping of IPv6 Multicast Packets on Ethernet," 2011
- RFC 6104, "Rogue IPv6 Router Advertisement Problem Statement," 2011
- RFC 6105, "IPv6 Router Advertisement Guard," 2011
- RFC 6106, "IPv6 Router Advertisement Options for DNS Configuration," 2010
- RFC 6119, "IPv6 Traffic Engineering in IS-IS," 2011
- RFC 6221, "Lightweight DHCPv6 Relay Agent," 2011
- RFC 6226, "PIM Group-to-Rendezvous-Point Mapping," 2011
- RFC 6282, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," 2011
- RFC 6294, "Survey of Proposed Use Cases for the IPv6 Flow Label," 2011
- RFC 6308, "The Internet Multicast Address Allocation Architecture," 2011
- RFC 6326, "Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS," 2011
- RFC 6334, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Option for Dual-Stack Lite," 2011
- RFC 6398, "IP Router Alert Considerations and Usage," 2011
- RFC 6422, "Relay Supplied DHCP Options," 2011
- RFC 6434, "IPv6 Node Requirements," 2011
- RFC 6436, "Rationale for Update to the IPv6 Flow Label Specification," 2011
- RFC 6437, "IPv6 Flow Label Specification," 2011
- RFC 6438, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels," 2011
- RFC 6553, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams," 2012
- RFC 6554, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)," 2012
- RFC 6555, "Happy Eyeballs: Success with Dual-Stack Hosts," 2012
- RFC 6556, "Testing Eyeball Happiness," 2012
- RFC 6603, "Prefix Exclude Option for DHCPv6-based Prefix Delegation," 2012
- RFC 6724, "Default Address Selection for Internet Protocol Version 6 (IPv6)," 2012
- RFC 6775, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," 2012

- RFC 6822, "IS-IS Multi-Instance," 2012
- RFC 6845, "OSPF Hybrid Broadcast and Point-to-Multipoint Interface Type," 2013
- RFC 6853, "DHCPv6 Redundancy Deployment Considerations," 2013
- RFC 6895, "Domain Name System (DNS) IANA Considerations," 2013
- RFC 6939, "Client Link-Layer Address Option in DHCPv6," 2013
- RFC 6977, "Triggering DHCPv6 Reconfiguration from Relay Agents," 2013
- RFC 6992, "Routing for IPv4-Embedded IPv6 Packets," 2013
- RFC 7031, "DHCPv6 Failover Requirements," 2013
- RFC 7037, "Radius Option for the DHCPv6 Relay Agent," 2013
- RFC 7078, "Distributing Address Selection Policy Using DHCPv6," 2014
- RFC 7084, "Basic Requirements for IPv6 Customer Edge Routers," 2013
- RFC 7098, "Using the IPv6 Flow Label for Load Balancing in Server Farms," 2013

Drafts

Drafts can be found at *http://www.ietf.org/ID.html*. To locate the latest version of a draft, refer to *https://datatracker.ietf.org/public/pidtracker.cgi*. You can enter the draft name without a version number and the most current version will come up. If a draft does not show up, it was possibly deleted. If it was published as an RFC, the RFC number will be displayed. *http://tools.ietf.org/wg* is also a very useful site. More information on the process of standardization, RFCs, and drafts can be found in Appendix A.

Here's a list of drafts I refer to in this chapter, as well as interesting drafts that relate to the topics in this chapter:

"Transmission of IPv6 Packets over BLUETOOTH Low Energy"
draft-ietf-6lo-btle-01

"Transmission of IPv6 Packets over IEEE 802.11p Networks"
draft-petrescu-ipv6-over-80211p-01

"Registering self-generated IPv6 Addresses in DNS using DHCPv6"
draft-ietf-dhc-addr-registration-04

"Provisioning IPv4 Configuration Over IPv6 Only Networks"
draft-ietf-dhc-v4configuration-05

"DHCPv4 over DHCPv6 Transport"
draft-ietf-dhc-dhcpv4-over-dhcpv6-08

"DHCPv4 over IPv6 Transport"
draft-ietf-dhc-dhcpv4-over-ipv6-09

"DHCPv6 Failover Design"
draft-ietf-dhc-dhcpv6-failover-design-04

"DHC Load Balancing Algorithm for DHCPv6"
draft-ietf-dhc-dhcpv6-load-balancing-01

"Populating the DNS Reverse Tree for DHCP Delegated Prefixes"
draft-ietf-dhc-dns-pd-01

"DHCPv6/SLAAC Address Configuration Interaction Problem Statement"
draft-ietf-v6ops-dhcpv6-slaac-problem-00

"DHCPv6/SLAAC Interaction Operational Guidance"
draft-liu-v6ops-dhcpv6-slaac-guidance-01

"Reducing Multicast in IPv6 Neighbor Discovery"
draft-yourtchenko-colitti-nd-reduce-multicast-00

"A comparison between the DHCPv6 and RA based host configuration"
draft-yourtchenko-ra-dhcpv6-comparison-00

CHAPTER 6

Security with IPv6

The developers of IPv4 did not rack their brains about security. The "Internet" in those early days connected a few trusted networks of some visionary researchers. The individuals who controlled these networks, as well as those who were allowed to use the networked resources, were implicitly trusted to not cause any malicious or destructive behavior. This is the reason why the original IP architecture does not include a security framework that can be used by all applications. If security was needed, it was usually rudimentary authentication/authorization and was included in the application code (e.g., the password for Telnet and FTP). Many years later, IPsec was introduced when IPv4 had already been widely deployed. Therefore, it needed to be retrofitted into existing deployments. Due to interoperability and performance issues and to the fact that it was developed later, IPsec is not as widely deployed as it could be in many IPv4 scenarios. This is in contrast to IPv6, which from the beginning had the notion that fundamental security functionality had to be included in the base protocol in order to be used on any Internet platform. In the early days of IPv6, a standards-conforming IPv6 implementation had to include IPsec to allow more secured communication once it was appropriately configured. This strict rule has been loosened recently, but more about this later. Before we dive into the technical details, I want to talk about some general security concepts and practices.

General Security Concepts

In order to protect data, one has to be aware of the possible threats. People often focus solely on malicious attacks from foreign networks. A comprehensive security concept needs to consider many other aspects. The following is a list of some possible points of weakness:

- Insufficient or nonexistent IT security concepts and corresponding provisions
- Nonobservance or insufficient control of IT security provisions

- Usurping of rights (password theft, privilege escalation)
- Incorrect use or faulty administration of IT systems
- Abuse of rights
- Weaknesses in software (e.g., buffer/heap overflows in conjunction with applications running with superuser rights, cross-site scripting)
- Manipulation, theft, or destruction of IT devices, software, or data (physical security)
- Network eavesdropping (sniffing wired or wireless networks) or replaying of messages
- Trojan horses, viruses, and worms
- Security attacks such as masquerading, IP spoofing, Denial of Service (DoS) attacks, man-in-the-middle attacks, or DNS poisoning
- Routing misuse

There are many statistics showing that malicious attacks from the outside are only a smaller fraction of all the possible risks. Many threats come from within the internal network and can in many cases be related to human misconduct or faulty administration. Many of these risks cannot be controlled by technical mechanisms. This chapter is not a guide to an overall security concept; it discusses the technology aspects of security with IPv6.

General Security Practices

Standard security practices involve two "triads" of thought, CIA and AAA. The CIA triad includes:

Confidentiality
: Stored or transmitted information cannot be read or altered by an unauthorized party.

Integrity
: Any alteration of transmitted or stored information can be detected.

Availability
: The information in question is readily accessible to authorized users at all times.

The AAA triad includes:

Authentication
: Ensuring an individual or group is who they say they are. The act of clarifying a claimed identity. Common forms of authentication include usernames and passwords or ATM card/PIN combinations.

Authorization
: Ensuring that the authenticated user or group has the proper rights to access the information they are attempting to access. Common implementations include Access Control Lists (ACLs).

Accounting
: The act of collecting information on resource usage. The log of an HTTP server would be a common form of accounting.

Nonrepudiation is not included in the CIA/AAA triads. Nonrepudiation means a specified action, such as sending, receiving, or deleting of information, cannot be denied by any of the parties involved.

These security requirements need to be provided by two basic security elements: encryption (to provide confidentiality) and secure checksums or hash (to provide integrity). Suitable combinations of these two elements may then be used to provide more complex services, such as authenticity and nonrepudiation.

There are two forms of encryption that are commonly used. The first is called *Secret Key Cryptography*, also termed *symmetric key encryption*, which requires the sender and recipient to agree on a shared secret (i.e., a key or password) that is then used to encrypt and decrypt the information exchanged. Common symmetric key algorithms are AES, DES, 3DES, IDEA, and RC-4.

The second is called *Public Key Cryptography*, also termed *asymmetric encryption*. An asymmetric encryption algorithm uses a key pair consisting of a known and distributed public key and an individual private key. When a message is encrypted using the public key and decrypted by the receiver with the corresponding private key, only the intended recipient is capable of seeing the encrypted message. This form of encryption can be used to establish a confidential data exchange. If in addition, the message was also encrypted with the sender's private key and then decrypted by the recipient with a corresponding public key, the security services of data origin authentication and nonrepudiation are added. Common asymmetric key algorithms are RSA, ElGamal, and elliptic curves cryptography (ECC).

Secure checksums or hash functions often provide data integrity. A hash function takes input of an arbitrary length and outputs fixed-length code. The fixed-length output is called the *message digest*, or the *hash*, of the original input message. These hashes are unique and thereby provide the integrity of the message. Common one-way hash functions are SHA-1 and MD-5.

The IPsec standard uses a combination of algorithmic choices based on symmetric and asymmetric cryptography, as well as one-way hash functions. This chapter describes the IPsec framework and the security elements in IPv6 and includes a discussion about special issues to be aware of when securing an IPv6 network.

IPsec Basics

IPsec, described in RFC 4301, defines a security architecture for both versions of IP for IPv4 and IPv6.

The following elements are part of the IPsec framework:

- A general description of security requirements and mechanisms at the network layer
- A protocol for encryption (Encapsulating Security Payload, or ESP)
- A protocol for authentication (Authentication Header, or AH)
- A definition for the use of cryptographic algorithms for encryption and authentication
- A definition of security policies and security associations between communication peers
- Key management

The configuration of IPsec creates a boundary between a protected and an unprotected area. The boundary can be around a single host or a network. The access control rules specified by the administrator determine what happens to packets traversing the boundary. The security requirements are defined by a *Security Policy Database* (SPD). Generally, each packet is either protected using IPsec security services, discarded, or allowed to bypass IPsec protection, based on the applicable SPD policies identified by the *selectors*. The selectors are the specific traffic-match criteria defined by an administrator—for example, a specific application being transmitted from a subnet to a specific end-host.

Security Associations

Security Associations (SA) are agreements between communication peers. Three elements are part of the agreement: a key, an encryption or authentication mechanism, and additional parameters for the algorithm. SAs are unidirectional, and each separate security service requires an SA. This means that two communication peers who want to encrypt and authenticate a two-way communication need four SAs (one pair for encryption and one pair for authentication). Bidirectional application traffic—for example, a bidirectional Telnet connection—also requires four SAs at each communication peer. Peer A must protect the traffic it initiates and the return traffic from Peer B. It also requires two additional SAs to ensure that if Peer B initiates a Telnet session, both it and the return traffic for this scenario are protected.

IPsec differentiates two modes of transport:

Transport mode
: The SA is made between two end nodes and defines the encryption or authentication for the payload of all IP packets for that connection. The IP header is not encrypted.

Tunnel mode
: The SA is usually made between two security gateways (usually a firewall). The whole packet including the original IP header is encrypted or authenticated by encapsulating it in a new header. This is the foundation for a virtual private network (VPN).

Key Management

Most of the security mechanisms provided by IPsec require the use of cryptographic keys. A separate set of mechanisms has been defined for putting the keys in place. Support for both manual and automated distribution of keys is a requirement. RFC 4301 specifies IKEv2 (described later in this section) as an automated key distribution mechanism. Other mechanisms may be used. IKEv1 is actually officially deprecated and to be replaced with IKEv2, but as it is still used widely today, we describe both versions in this book.

In order to establish a Security Association (SA), the communication peers have to agree on a cryptographic algorithm and negotiate keys. The negotiation of an SA often happens over insecure paths. Internet Key Exchange (IKE) specifies a protocol that allows for the exchange and negotiation of parameters for an SA.

IKEv1

IKEv1 is specified in RFC 2409 and updated in RFC 4109. It consists of selected functions from three different protocols:

ISAKMP (Internet Security Association and Key Management Protocol)
: ISAKMP specifies a framework for the management of SAs and key exchange without describing the process in detail. It therefore supports different key exchange mechanisms. It is specified in RFC 5996.

Oakley Key Determination Protocol
: The Oakley Key Determination protocol is used for the exchange of keys and is specified in RFC 2412. It is an extension of the Diffie-Hellman algorithm. It uses only a subset of the functions of the Oakley protocol.

SKEME (Versatile Secure Key Exchange Mechanism for the Internet)
: SKEME is a fast key exchange technique described in "SKEME: A Versatile Secure Key Exchange Mechanism for the Internet," from *IEEE Proceedings of the 1996 Symposium on Network and Distributed Systems Security* by H. Krawczyk. IKE uses only a subset of the functions defined for the SKEME protocol.

IKEv1 uses UDP on port 500 or 4500 and goes through two phases.

In phase one, the ISAKMP communication peers negotiate a secure, authenticated communication channel called ISAKMP Security Association. Note that some implementations use the term "IKE SA," which is synonymous with the ISAKMP SA. The phase one exchange is based on the Diffie-Hellman algorithm and encrypted identification tokens. The authentication can be provided by either preshared keys, an RSA checksum encrypted with the private key of the sender, or the public key of the receiver combined with its X.509 certificate.

In phase two, the cryptographic algorithms and the keys for other protocols (e.g., ESP and/or AH) can be exchanged over the secure communication channel established in phase one. The outcome of IKE phase two results in an IPsec SA. These IPsec SAs define the security services to be used for protecting the traffic in transit. Multiple IPsec SAs can be negotiated via the secure channel set up in phase one. This allows for more granular and more flexible security services to be negotiated. Both the IPsec SAs and ISAKMP SAs generate new cryptographic keys on a periodic basis to provide greater security. Typically, the IPsec SAs are rekeyed at a faster rate than the ISAKMP SAs.

RFC 4109 updates the original specification. The changes are made to ensure that the suggested and required algorithms reflect the current market situation. The changes are intended to be deployed for all IKEv1 implementations.

You can find all IKEv1 relevant and updated numbers and codes at *http://www.iana.org/assignments/ipsec-registry*.

IKEv2

IKEv2 is specified in RFC 5996, which combines and therefore obsoletes the following RFCs: RFC 4306, "Internet Key Exchange (IKEv2) Protocol," RFC 2407, "The Internet IP Security Domain of Interpretation for ISAKMP," RFC 2408, "Internet Security Association and Key Management Protocol (ISAKMP)," and RFC 2409, "The Internet Key Exchange (IKE)." IKEv2 brings, among other things, enhancements for the use of IPsec in combination with NAT traversal, for Extensible Authentication, and for Remote Address acquisition.

IKEv2 runs over UDP ports 500 and 4500, and a host running IKEv2 must accept packets from any ports and respond to those same ports.

The initial exchange (called *phase one* in IKEv1 terminology) normally consists of two pairs of messages. The first message pair negotiates cryptographic algorithms, exchanges nonces (number used once to prevent replay attack), and does a Diffie-Hellman

exchange. The second message pair authenticates the previous messages, exchanges identities and certificates, and establishes the first `CHILD_SA`.

In IKEv1, SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. The end with the shorter lifetime will have to request the rekeying if the lifetime policies are different. Another difference between the two IKE versions is that IKEv2 allows parallel SAs with the same traffic selectors between common endpoints. This difference, among other things, supports traffic with different Quality of Service (QoS) requirements among the SAs. Hence, unlike IKEv1, the combination of the endpoints and the traffic selectors may not uniquely identify an SA between those endpoints. Therefore, with IKEv2, SAs can no longer be deleted based on duplicate traffic selectors.

Opening an IPsec connection through a NAT creates some special problems. The changing of IP addresses changes the checksums, so they will fail and cannot be corrected by the NAT, because they are cryptographically protected. IKEv2 has improved support for such situations by negotiating UDP encapsulation of IKE and ESP packets. Port 4500 is reserved for UDP-encapsulated ESP and IKE. Because NATs often translate TCP and UDP port numbers, IPsec packets must be accepted from any port and sent back to that same port. The good news is that we don't build NATs in IPv6 networks, so this is not an issue anymore.

You can find all relevant and updated IKEv2 numbers and codes at *http://www.iana.org/assignments/ikev2-parameters*.

A summary of the changes to IKEv2 can be taken from RFC 5996. Here are some of the main points (refer to the RFC for the whole list):

- To define the entire IKE protocol in a single document, replacing RFCs 2407, 2408, and 2409, and incorporating subsequent changes to support NAT Traversal, Extensible Authentication, and Remote Address acquisition.
- To simplify IKE by replacing the eight different initial exchanges with a single four-message exchange.
- To decrease IKE's latency in the common case by making the initial exchange two round trips (four messages) and allowing the ability to piggyback setup of a `CHILD_SA` on that exchange. It also adds Denial of Service resistance to the initial exchange.
- To reduce the number of possible error states by making the protocol reliable. This is done by requiring all messages to be acknowledged and sequenced. This allows shortening `CREATE_CHILD_SA` exchanges from three messages to two.

- To maintain existing syntax and magic numbers to the extent possible to make it likely that implementations of IKEv1 can be enhanced to support IKEv2 with minimum effort.

A list of algorithms to be used with IKEv2 is specified as mandatory to implement in RFC 4307 "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)." This is in order to ensure interoperability between different implementations.

For a good overview of the relevant security RFCs and drafts, go to the IETF Security working groups at *http://www.ietf.org/html.charters/wg-dir.html*.

IPv6 Security Elements

IPsec describes general security mechanisms that can be used with both protocols, IPv6 and IPv4. This means that IPv6 is not more secure than IPv4.

In the early days of IPv6, the implementation of IPsec in every IPv6 stack was mandated and the use of IKE for key management recommended. With RFC 6434, "IPv6 Node Requirements," this strict rule was downgraded to "SHOULD." It is now up to the vendors to decide whether their IPv6 products require IPsec or not (or up to the customer to demand it). The main reasons for this were that it seemed impractical to demand full IPsec implementations on all sorts of special device types, for instance, sensors where resources are very limited, or device types with very special applications that may have an application-based security approach.

On the other hand, the same RFC states that nodes that implement IPsec have stronger requirements than before. Support for RFC 4301, "Security Architecture for the Internet Protocol," is now mandatory, which includes the use of IKEv2 for automatic key management and requires support for a minimum set of cryptographic algorithms, making IPsec more interoperable across vendor implementations.

The IPsec specification defines protocols for the Authentication Header (AH) and the Encapsulating Security Payload Header (ESP). With IPv6, these headers are included as Extension headers. An IPsec implementation must support ESP and may support AH. With the older specification, support for both protocols was required. The requirement for AH support has been removed because ESP can be used to provide integrity, which in most cases has proven to be sufficient.

To understand Extension headers and Next Header values, refer to Chapter 3.

Authentication Header

The Authentication Header (AH) provides integrity and authentication (no confidentiality) for all end-to-end data transported in an IP packet. It supports different authentication mechanisms. It is specified in RFC 4302 and is indicated by the Next Header value 51 in the preceding header.

The AH is located between the IPv6 header and upper-layer headers (e.g., TCP, UDP, ICMP). If Extension headers are present, it has to be placed after the Hop-by-Hop, Routing, and Fragment Extension headers.

The format of the AH is shown in Figure 6-1.

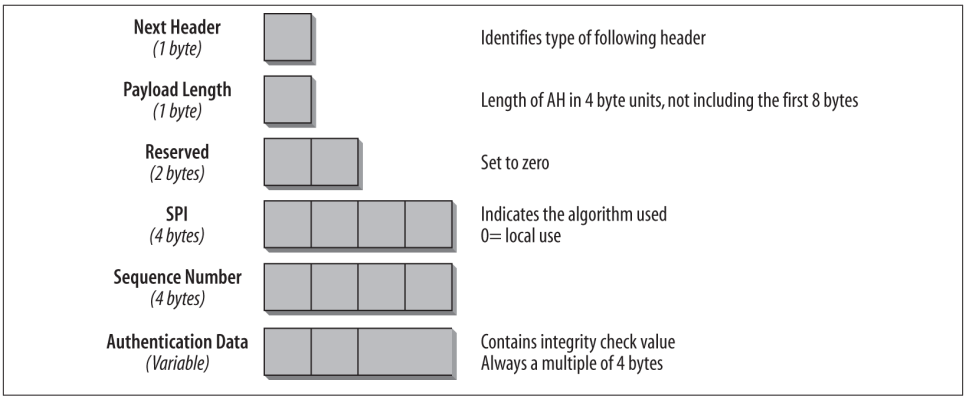

Figure 6-1. The format of the Authentication Header

Each field is discussed in the following list:

Next Header (1 byte)
: The Next Header field identifies the type of header that follows the Authentication Header. It uses the values listed in Table 3-1.

Payload Length (1 byte)
: Describes the length of the header in four-byte units, not including the first eight bytes in the calculation. This length indication is necessary because the authentication data in the AH may differ in length depending on the algorithm used.

Reserved (2 bytes)
: Not used; set to 0.

Security Parameter Index (SPI) (4 bytes)
: Arbitrary 32-bit value. Used by the receiver to identify the SA to which an incoming packet belongs. The SPI field is mandatory, and this mechanism for mapping inbound traffic to unicast SAs must be supported by all AH implementations. If an IPsec implementation supports multicast, it must additionally support multicast SAs using the de-multiplex algorithm specified for this purpose for mapping inbound IPsec datagrams to SAs. SPI values in the range of 1 through 255 are reserved by the Internet Assigned Numbers Authority (IANA) for future use. The SPI value of 0 is reserved for local, implementation-specific use and must not be sent on the wire.

Sequence Number (4 bytes)
: This 32-bit sequence number is a monotonically increasing counter value. It has to be set by the sender, but it is the choice of the receiver whether to act on it. It ensures that packets with identical data are not resent repeatedly. This prevents replay attacks in a unicast or single-sender SA. For a multisender SA, the anti-replay features of AH are not available, because the AH does not have a means to synchronize packet counters among multiple senders. On establishment of an SA, the value is set to 0 at the sender and at the receiver. The first packet always has the value 1, which is increased by one for every consecutive packet. When the value 232 is reached, the counter is reinitialized to 0.

Integrity Check Value (variable length)
: This field contains the checksum (Integrity Check Value, or ICV) for the packet. The length depends on the algorithm chosen on establishing the SA. It is always a multiple of four bytes.

The AH Specification in RFC 4302 defines a new Extended (64-bit) Sequence Number (ESN). It cannot be seen in Figure 6-1 because only the low-order 32 bits of the Extended Sequence Number are transmitted. The high-order 32 bits are maintained as part of the sequence number counter by both transmitter and receiver, and are included in the computation of the ICV. The 64-bit sequence number is a new option designed to support high-speed IPsec implementations. The use of an Extended Sequence Number is negotiated on setup of the SA. The default with IKEv2 is ESN, unless 32-bit is explicitly negotiated.

The checksum is calculated over the following fields:

- All fields of the IP header or Extension header fields before the AH that do not change in transit or whose value when arriving at the destination can be predicted. For example, if a Routing Extension header is present, the last address in the Routing

Extension header is used for the calculation. The Traffic Class field, the Flow Label, and the Hop Limit are not included in the calculation.

- All fields of the Authentication Header.
- Other Extension headers present and the payload.
- The high-order bits of the ESN (if employed) and any implicit padding required by the integrity algorithm.

The following algorithms are considered suitable for IPsec:

- Keyed Message Authentication Codes (MACs) based on symmetric encryption algorithms.
- One-way hash functions (e.g., MD5, SHA-1, SHA-256).

Other algorithms can be negotiated. RFC 4305, "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)," lists the following implementation rules for AH:

- MUST, HMAC-SHA1-96 (RFC 2404)
- SHOULD, AES-XCBC-MAC-96 (RFC 3566)
- MAY, HMAC-MD5-96 (RFC 2403)

Weaknesses have become apparent in MD5; however, they should not affect the use of MD5 with HMAC.

The Authentication Header can be used in both transport and tunnel modes, as shown in Figure 6-2.

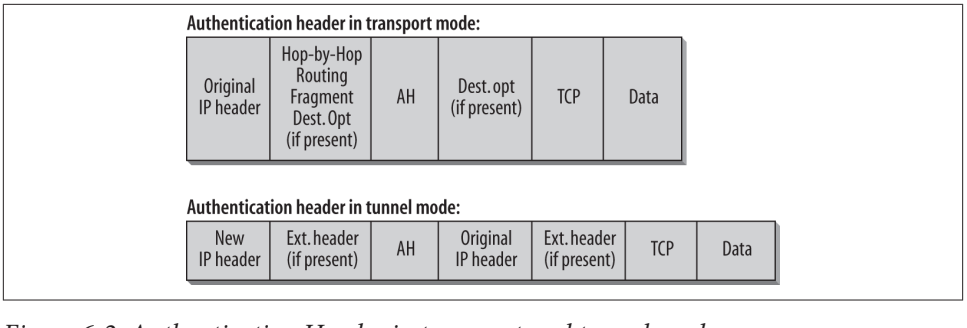

Figure 6-2. Authentication Header in transport and tunnel mode

In transport mode, the whole payload, including the fields of the IPv6 header, which do not change in transit, is secured. In tunnel mode, the inner packet contains the IP address of sender and receiver. The outer IP header contains the IP address of the tunnel

endpoints. In this case, the complete original packet, including the fields of the outer header that do not change in transit, is secured.

Encapsulating Security Payload Header

The Encapsulating Security Payload (ESP) header provides Integrity, Confidentiality, Data Origin Authentication, Anti-Replay Service, and limited Traffic Flow Confidentiality for all end-to-end data transported in an IP packet. The set of services provided is negotiated on establishment of the SA. The ESP is defined in RFC 4303 and indicated by a Next Header value of 50 in the preceding header.

The ESP header is located in front of the transport (e.g., UDP or TCP), network control (e.g., ICMP), or routing (e.g., OSPF) protocol header.

The format of the ESP is shown in Figure 6-3.

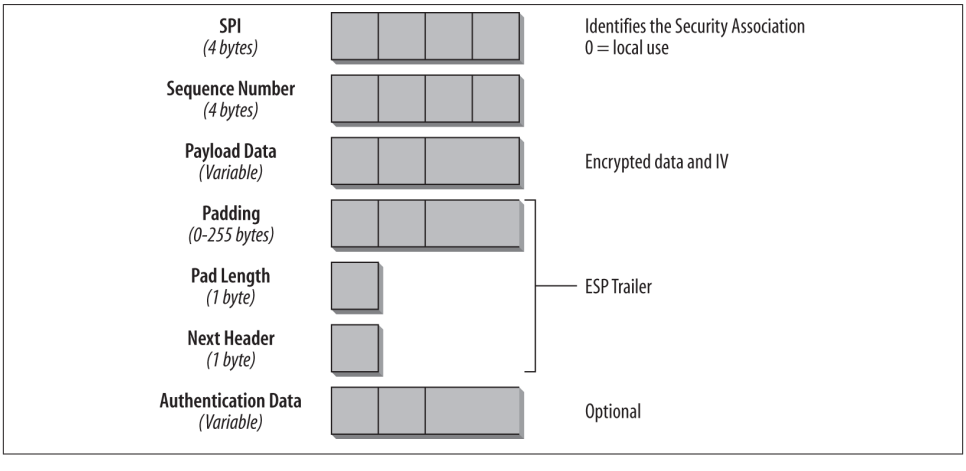

Figure 6-3. Format of the Encapsulating Security Payload header

Security Parameter Index (SPI) (4 bytes)
: Arbitrary 32-bit value. Used by the receiver to identify the SA to which an incoming packet belongs. The SPI field is mandatory, and this mechanism for mapping inbound traffic to unicast SAs must be supported by all ESP implementations. If an IPsec implementation supports multicast, it must additionally support multicast SAs using the de-multiplex algorithm specified for this purpose for mapping inbound IPsec datagrams to SAs. SPI values in the range of 1 through 255 are reserved by the Internet Assigned Numbers Authority (IANA) for future use. The SPI value of 0 is reserved for local, implementation-specific use and must not be sent on the wire.

Sequence Number (4 bytes)
: This 32-bit sequence number is a monotonically increasing counter value. It has to be set by the sender, but it is the choice of the receiver whether to act on it. It ensures that packets with identical data are not resent repeatedly. This prevents replay attacks in a unicast or single-sender SA. For a multisender SA, the anti-replay features of ESP are not available, because the ESP does not have a means to synchronize packet counters among multiple senders. On establishment of an SA, the value is set to 0 at the sender and at the receiver. The first packet always has the value 1, which is increased by one for every consecutive packet. When the value 2^{32} is reached, the counter is reinitialized to 0.

Payload Data (variable length)
: Contains the encrypted data as well as the encryption Initialization Vector (IV) if required by the encryption mechanism.

Padding (0 to 255 bytes)
: Used to align the packet to a multiple of 4 bytes and to reach a minimum packet size if the encryption mechanism requires one.

Pad Length (1 byte)
: Indicates the number of preceding padding *bytes*.

Next Header (1 byte)
: Identifies the type of header that follows the ESP header. It uses the values listed in Table 3-1. To facilitate the rapid generation and discarding of the padding traffic in support of traffic flow confidentiality, the protocol value 59 (no next header) designates a *dummy* packet. The receiver of a dummy packet must discard it without creating an error message.

Integrity Check Value (variable length)
: The Integrity Check Value (ICV) is a variable-length field containing a checksum computed over the ESP header, Payload, and ESP Trailer fields. The ICV field is optional. It is present only if the integrity service is selected, and it is provided by either a separate integrity algorithm or a combined mode algorithm that uses an ICV. The length of the field is specified by the integrity algorithm selected and associated with the SA.

The Padding, Pad Length, and Next Header fields are part of the ESP Trailer. The encryption algorithm is either specified manually and included in the SA for the packet stream or negotiated dynamically by the key exchange protocol.

The ESP specification in RFC 4303 defines an Extended (64-bit) Sequence Number (ESN). It cannot be seen in Figure 5-3, because only the low-order 32 bits of the Extended Sequence Number are transmitted. The high-order 32 bits are maintained as part of the sequence number counter by both transmitter and receiver and are included in the computation of the ICV. The 64-bit sequence number is a new option designed to

support high-speed IPsec implementations. The use of an Extended Sequence Number is negotiated on setup of the SA. The default with IKEv2 is ESN unless 32-bit is explicitly negotiated.

RFC 4835, "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)," lists the implementation rules for ESP.

Combined Mode algorithms provide both confidentiality and authentication services. They are expected to provide significant throughput and efficiency advantages. AES-CM (Advanced Encryption Standard Counter Mode), which has been adopted as the preferred mode for security in IEEE 802.11i, is defined in RFC 5084 and RFC 6655.

Since ESP authentication and encryption are both optional, support for NULL algorithms must be given for both. Note that only one of them can be set to NULL at a time.

The ESP can be used in both transport and tunnel modes, as shown in Figure 6-4.

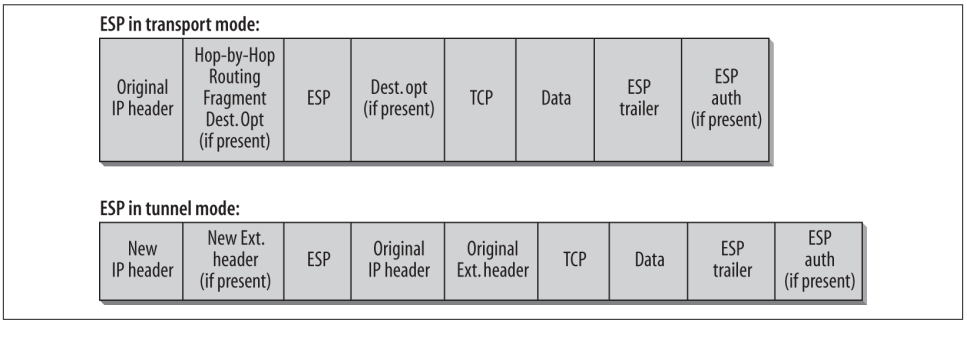

Figure 6-4. Encapsulating Security Payload header in transport and tunnel mode

In transport mode, the IP header and the Extension headers that follow are not encrypted; otherwise, the packet could not be forwarded. If the complete packet has to be encrypted, tunnel mode is the way to go. Just as with the AH in tunnel mode, the inner packet contains the IP address of sender and receiver, whereas the outer IP header contains the IP address of the tunnel endpoints.

The ESP header can be used with a NULL encryption option, which is defined in RFC 2410. With NULL encryption, only the authentication option of the ESP is used, and the packet is not encrypted.

Combination of AH and ESP

The two headers can be used in combination as well. In that case, the AH has to precede the ESP header to verify authenticity and integrity before the packet is decrypted. The

authentication option has been included in the ESP header to allow for authentication of encrypted packets with only one header.

If the AH header is used in tunnel mode, the first IP header is included in the authentication. If the ESP header is used, only the part of the packet following the ESP header is authenticated. If encryption and integrity of the IP addresses are required, both headers must be combined. If both headers are used, it is obviously not necessary to use the authentication in the ESP header. On the other hand, an ESP header with NULL encryption can be used if the given authentication is sufficient.

Interaction of IPsec with IPv6 Elements

The use of IPsec for IPv6 provides the same security as IPsec for IPv4 in the Internet. But there are some areas where IPsec cannot easily be combined with other services:

- *Tunneling*, a fundamental element of both IPsec and multiple transition mechanisms, creates difficulties for existing firewalls and security gateways at the edge of the internal network. An encrypted IPsec tunnel built through a firewall providing end-to-end security for hosts on either side makes it impossible for the firewall to detect dangerous or unauthorized content. One way to solve this issue is to define the SAs between the security gateways, not between the end nodes. Another issue is that the inner packet can contain information that presents a threat for the internal network. This could be routing information or network control messages (e.g., ICMP redirect).
- The extended mobility options with constantly changing IP addresses can lead to situations that are difficult to manage and control in IPsec environments. Dynamic addresses such as privacy addresses (RFC 4941) create difficulties if they are used for IKE identity checks.

IPv6 Security "Gotchas"

Security in an IPv6 network is not substantially different from security in an IPv4 network. Many of the existing well-known IPv4 attacks can be performed with IPv6, so our means for securing data is similar.

Just like in the IPv4 world, there will always be unethical hackers who find new ways to break into our networks. The designers of security concepts and the entire computer security community will have to stay alert and keep finding mechanisms to keep pace with the hackers and find ways to protect their networks from new attacks.

If both protocols (IPv4 and IPv6) are used in a network, each protocol needs its own security concept and provisions, which need to be coordinated.

Having a strong and well-thought-out security design in your dual-stack network is a must. But even before you deploy IPv6, you must make a first step of protection. IPv6 is included in most operating systems, usually rather simple to configure, and often activated by default. Often even tunnel mechanisms are activated by default. IPv4 network administrators may believe that they do not have to worry about IPv6, but they don't realize that there may be IPv6 traffic in their network already. This fact is used by IPv6 hackers to intrude into IPv4 networks.

Find the IPv6 traffic in your network by filtering your trace files for `0x86DD` in the MAC header or for protocol 41 in the IPv4 protocol type field. You may find a surprising number of Neighbor Discovery messages—a good indicator that you have active IPv6 nodes on your network.

As a first step of protection, while you have not deployed IPv6 yet, make sure you filter all IPv6 traffic coming into and going out from your network. This includes native IPv6 traffic and tunneled traffic. And in addition to this it could be wise to monitor and audit IPv6 traffic, specifically ND messages such as Router or Neighbor Advertisements that could be used to misconfigure your clients.

RFC 7123, "Security Implications of IPv6 on IPv4 Networks," provides useful suggestions on how to protect such an "unmanaged" IPv6 enterprise network.

IPsec, while being a good security mechanism, is not the be-all and end-all for security. Most security professionals agree that there is no *silver bullet* in securing a network from internal or external attacks. Combinations of best practices and user training can minimize risks. If you are deploying IPv6 in the near future, there are some security concerns that should be addressed. Be advised that this is not a complete list, as entire volumes of information could be written on the subject.

Native IPv6

When connecting to IPv6 natively, some security issues that should be considered are discussed in the following sections.

Public Key Infrastructure (PKI)

While RFC 4301 specifies the requirement for IPSec in the IPv6 protocol, it does not cover how the keys will be exchanged. You could manually set up preshared keying, but in large enterprises, this task becomes tedious and time consuming. In such environments, using a central certificate server is ideal. With IPv6, there were no such centralized certificate servers until recently. This has changed with servers that are based on the IKEv2 protocol, which is specified in RFC 5996. Benefits of IKEv2 are its usability on both IPv4 and IPv6 and its simpler implementation. It is not compatible with the first version of IKE, but both versions' header formats are similar enough to run them both unambiguously over the same UDP port.

Firewalls and intrusion detection/prevention systems

While end-to-end IPsec is considered one of the major advantages of IPv6, it also introduces new problems with existing firewalling and IDS/IPsec if ESP is used with encryption turned on. If the packets are encrypted from end to end, how does a middle device inspect the packets without decrypting them? Storing all the encryption keys in a central location leaves both a central point of failure and a central location for a black-hat hacker to break into and steal all the encryption keys for the network. Issues in current IPv4 IDS/IPS systems include lack of detection of tunneled IPv6 protocols and a general lack of attack signatures for IPv6-based attacks, although as IPv6 proliferates to more networks, these issues should eventually be solved.

Firewalls that support IPv6 are included in most of the major operating systems, but firewalls that keep the state of connections (*stateful firewalls*) are not available in older implementations. Cisco, Checkpoint, and Netscreen (Juniper), among others, have stateful inspection of IPv6 packets. Firewall products must be carefully evaluated and tested against a specific list of RFC requirements. Refer to Chapter 9 for more information. Even though most implementations have some basic features available, often important features that are needed in an IPv6 network are still lacking support.

The draft "Requirements for IPv6 Enterprise Firewalls" (*draft-gont-opsec-ipv6-firewall-reqs-01*) specifies a set of requirements for IPv6 firewalls, in order to establish some common-ground in terms of what features can be expected and should be requested in RFPs (Request for Proposal).

Implementation issues

Many IPv6 implementations are fairly new. This leads to two other possible security issues. The first issue is a lack of IPv6 assessment tools. It is a common practice in the information security field to audit your own network with well-known security auditing tools, and then secure the flaws found with those tools. Many of these popular tools are in the process of being ported to audit IPv6 networks, so you have to check the maturity level of your favorite tools. The second issue is untested code in IPv6 implementations

(which also ties into not having the tools to test the implementations). Code that has not been "put through the wringer" is probably going to have more security flaws than code that has been used in production environments for a long time. Vendors have to wait for market response and incident reports to fix their implementations. In your planning you need to allow for enough time to test stacks and security implementations in your lab before you go to production, and then establish a process for rapidly addressing security implementation risks and flaws coming up.

On the other hand we have to understand that in many cases transport layer security is overemphasized, while much more efficient measures such as phishing, Trojans, etc., are underestimated.

Neighbor Discovery issues

ARP in IPv4 has been replaced with ICMP messages in IPv6. Without using IPsec AH or SEND, however, NDP has many of the security problems that ARP presented in IPv4, such as redirect attacks (malicious nodes redirecting packets away from legitimate receivers), Denial of Service (DoS) attacks, and flooding attacks (redirecting other hosts' traffic to a victim node creating a flood of bogus traffic). Duplicate Address Detection and Router Advertisements can also be abused. These attacks are similar to the ARP/DHCP attacks in IPv4. A quote from RFC 4862 (IPv6 Stateless Autoconfiguration) states:

> If a node determines that its tentative link-local address is not unique, autoconfiguration stops and manual configuration of the interface is required.

This poses a possible Denial of Service attack, as multiple IPv6 addresses can be assigned to a single interface. A rogue workstation could be assigned several thousand addresses and deny other workstations the ability to acquire a link-local address. Or even much simpler, a software responder can be built that always responds with "address in use" such as new IPv6 tools from The Hacker's Choice.

Another point is that link-local addresses can be acquired without preconfiguration. An attacker can get access to a link without any further knowledge about the network. This feature gives a malicious node the opportunity to mount an attack on any other node attached to this link. Possible ways to protect from this are either link-layer authentication (Network Access Control) or the use of Cryptographically Generated Addresses (see below).

First-hop security

Router Advertisement spoofing is another security concern also known as *rogue RA*. Since multiple addresses are allowed on a single interface, multiple routes are allowed as well. A booting node sends a Router Solicitation to the all-routers link-local multicast

address (`ff02::2`). Each router on the link replies with a Router Advertisement containing configuration information for the client. This offers the possibility of sending traffic through a router through which it's not supposed to be sent (allowing traffic sniffing on the rogue router with the potential of mounting a man-in-the-middle attack). An attacker can use other options in the RA to misguide clients, for instance configuration of wrong DNS server information (if supported by the client stack), a very small MTU size for all nodes on the link, or a hop limit of 1 for all nodes. Obviously, this type of attack happens in the IPv4 world also; it differs only in the mechanisms used. Using the IPsec's AH component, SEcure Neighbor Discovery (SEND), or RA Guard are ways to mitigate this risk, among others.

RA Guard (Router Advertisement Guard) is defined in RFC 6105. The purpose of RA Guard is to filter Router Advertisements on Layer 2, based on a set of criteria, before they reach the target. The criteria can be to allow RAs from a set of defined sources, to disallow it on a given interface, to allow it from authenticated sources, etc. It can run in a stateful or stateless mode. The specification also defines the concept of a router authorization proxy that would check RAs on behalf of the clients according to the criteria defined. And then only send out RAs from the link that has been approved as coming from a legitimate source.

The efficiency of RA Guard depends on the capability of the Layer 2 device to detect Router Advertisement messages. Practice has shown that in some implementations it is possible to circumvent that capability with Extension headers, namely Fragmentation headers. When packets are fragmented into multiple fragments, Layer 2 devices may not find all the necessary information to perform packet filtering. RFC 6980, "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery," updates RFC 4861 such that the use of the IPv6 Fragment header with traditional Neighbor Discovery is forbidden. RFC 7113, "Implementation Advice for IPv6 Router Advertisement Guard (RA Guard)," describes the issues and proposes filtering rules. It is an update to RFC 6105 on RA Guard.

The specification in NDP suggests using IPsec to protect from attacks, without providing more detailed information on how to do this. In many cases, especially in public and wireless networks, the key management used with IPsec is too complex and impractical.

A reference that outlines the possible threats with NDP and provides guidelines is RFC 3756, "IPv6 Neighbor Discovery (ND) Trust Models and Threats."

RFC 6946 discusses the issue of *atomic fragments*. An atomic fragment is a packet that contains a Fragment header but is actually not fragmented. This can happen if a host

receives an ICMPv6 "Packet Too Big" message with a next-hop MTU of less than 1,280 bytes. The Fragment header contains "Fragment Offset" and "M-Bit" set to zero. This can then be used to cause the host to fragment packets and then launch a fragmentation-based attack against such traffic. Many implementations treat these atomic fragments like regular fragments. This can be used for attacks. It would be better to process them as regular packets, with no need to queue them and wait for more fragments.

The Fragment header is described in Chapter 3 and Neighbor Discovery in Chapter 4.

SEcure Neighbor Discovery (SEND) is defined in RFC 3971 to secure NDP without using IPsec. This approach involves the use of new NDP options to carry public key-based signatures. A zero-configuration mechanism is used for showing address ownership on individual nodes; routers are certified by a trust anchor. SEND is a protocol-established trust between IPv6 Layer 2 adjacent nodes, but it suffers from a lack of implementation in generic hosts such as Windows or Mac OS X and only a couple of routers have implemented it. This renders SEND a generally unusable solution.

You can find a description of SEND and Cryptographically Generated Addresses (CGA) in Chapter 4.

As part of your *first-hop security* strategy you will also configure mechanisms such as *ICMP Snooping*, *DHCPv6 Guard*, and *IPv6 Destination Guard*. Refer to your vendor's configuration guide to learn how to configure these features.

Fragmentation

As the section on ND and first-hop security have shown, there are several cases where Extension headers and especially the Fragment header can cause security issues. RFC 6980, "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery," describes these issues and forbids the use of Fragmentation with Neighbor Discovery. The RFC specifies that nodes MUST NOT employ IPv6 fragmentation for sending any of the following Neighbor Discovery and SEcure Neighbor Discovery messages:

- Neighbor Solicitation
- Neighbor Advertisement
- Router Solicitation

- Router Advertisement
- Redirect
- Certification Path Solicitation

If an IPv6 host receives ND messages with fragmentation, the message should be silently discarded. So this is an update to the original ND specification in RFC 4861. If you notice that ND messages are discarded, verify that fragmentation is the cause and find out where it is coming from.

Address and port scanning

Address scanning has become much more complex, if not impractical. The interface identifier in IPv6 has 64 bits. RFC 4846, "Local Network Protection for IPv6," states that "an attacker has to send out a simply unrealistic number of pings to map the network, and virus/worm propagation will be thwarted in the process. At full rate 40 Gbps (400 times the typical 100 Mbps LAN, and 13,000 times the typical DSL/Cable access link) it takes over 5,000 years to scan a single 64-bit space." Go figure! If Autoconfiguration is used without the Privacy option, some parts of the address (e.g., Vendor ID) can be guessed, but it is still a vast space. A simple and good protection is to avoid easy-to-guess addressing schemes, not using words such as `BEEF`, `F00D`, `CAFE`, `1234`, and `ABCD` as part of an IPv6 address, and not using sequentially numbered or easy-to-guess addresses to key infrastructure devices (such as `x::1` for routers).

On the other hand, getting an easy-to-remember address such as `x::1` for routers makes network operations easier, hence more reliable, hence more secure as well. The network operator has to balance the risk and the benefit of using predictable addresses.

RFC 5157, "IPv6 Implications for Network Scanning," discusses this topic and explains alternative methods attackers may use. It also makes recommendations on how you can address these issues in your design.

Because an IPv6 network cannot be scanned, this also renders network inventory more complex, and knowing who uses your network is, of course, also a key security ingredient. Netflow or periodic NDP cache scans by SNMP can help with making your network inventory. You may also use your IPAM (IP address management) or DDI (DNS, DHCP, IP Address Management) tools for information on your network.

Port scanning, on the other hand, commonly refers to scanning the TCP and UDP port space on a particular IP address to determine which services are running. Since IPv6

doesn't change the TCP or UDP port space from its existing 16 bits, port scanning is still a threat.

Multicast issues

IPv6 supports multicast addresses with site scope, which can potentially allow an attacker to identify certain important resources on the site if misused. Particular examples are the all-routers (`ff05::2`) and all-DHCP-Servers (`ff05::1:3`) addresses. An attacker that is able to infiltrate a message destined for these addresses onto the site will potentially receive in return information identifying key resources on the site. This information can then be used for directed attacks ranging from simple flooding to more specific mechanisms designed to subvert the device. The risk can be minimized by ensuring that all firewalls and site boundary routers are configured to drop packets with site scope Destination addresses. Also, nodes should not join multicast groups for which there is no legitimate use on the site, and site routers should be configured to drop packets directed to these unused addresses.

Transition and Tunneling Mechanisms

IPv4 is not expected to go away anytime soon. It is very likely that there will be IPv4 nodes on networks for many years to come. IPv4 hosts cannot communicate with IPv6 hosts without some sort of transition or tunnel mechanism, which can add complexity to the existing network topology and the underlying code for the network stack. Transition and tunnel mechanisms can also be used as backdoors into normally IPv4-only networks.

Using IPv6 as a back door into IPv4 networks has been a known practice since 2002. On December 17, one of the HoneyNet Project's (*http://www.honeynet.org*) Solaris 8 servers was compromised. The difference between this attack and earlier ones was that this attacker set up an IPv6 tunnel to another country and tunneled out the data he was trying to steal. This bypassed many intrusion detection systems of the time, and it may do so today. This practice has been made easier with the advent of UDP-based tunneling mechanisms designed to allow IPv6 from behind NATs (a practice that was almost impossible before these mechanisms were available). Teredo and Tunnel Setup Protocol (TSP) are two such mechanisms.

Generally, with tunneling one has to make sure that packets that enter the network through a tunnel cannot circumvent incoming packet filters. An attacker from the Internet could, for instance, send an IPv4 packet to the tunnel endpoint (the entry point to your network) that contains an IPv6 packet with an IPv6 Source address from the range of your internal network. The tunnel endpoint decapsulates the packet and forwards the IPv6 packet to the internal network. The receiver believes that this packet originates from a host from the internal network. Automatic tunnels are more dangerous in that respect because they have to accept packets from any source. So a partial

protection can be to configure the tunnel endpoint to accept only packets from a configured tunnel entry point or to use only manually configured tunnels. But the attacker can still spoof that address. Additional filter mechanisms have to be implemented on the tunnel endpoint. It has to be treated like any other ingress point into your network. RFC 4891, "Using IPsec to Secure IPv6-in-IPv4 Tunnels," goes into more details and gives guidance on securing manually configured IPv6-in-IPv4 tunnels using IPsec.

6to4 tunneling is a well-known transition mechanism for networks that do not currently have native IPv6 connectivity. It consists of using a dual-stacked border router with a routable IPv4 address

Refer to Chapter 7 for a detailed discussion of 6to4.

In the 6to4 scenarios, there are special considerations discussed in RFC 3964, "Security Considerations for 6to4." The issues here are that: a) all 6to4 routers must accept and decapsulate IPv4 packets from every other 6to4 router and from 6to4 relays, and b) all 6to4 relay routers must accept traffic from any native IPv6 node. The routing scenarios to be analyzed are as follows:

- From 6to4 to 6to4
- From native IPv6 to 6to4
- From 6to4 to native IPv6

Please refer to the RFC for a detailed discussion of the scenarios and best practices to protect your network. And if applying our general design guidelines, you will not deploy 6to4 in a production network anyway.

The 6to4 router can be used for a Distributed Denial of Service (DDoS) attack using a tool called *4to6DDoS*. 4to6DDoS does not require an IPv6 stack to be installed on either the attacking or the victim host. It sends IPv6 in IPv4 encapsulated packets directly from v4 to v4. The routers used for 6to4 tunneling can also be DoS attacked by simply connecting to the router several times with a private IPv4 address. These routers must accept and decapsulate IPv4 packets even if they are forged. They must also accept traffic from any native IPv6 node. 6to4 also does not guarantee symmetric routing, meaning that traffic can take one routing path going to its destination, and take a completely different path upon return. This situation may not be optimal from a security and performance standpoint.

Automatic tunnels can create another vulnerability. If a packet comes in through a tunnel and the destination IPv6 address does not match a valid interface within the IPv6

network, it can bounce back in and out of the tunnel until the hop limit is exceeded. This applies to all automatic tunnels such as 6to4, ISATAP, and 6rd (all protocol 41 tunnels). It can be abused as a vehicle for traffic amplifications to facilitate DoS attacks. RFC 6324 describes the issues and mitigation measures such as verification of endpoint existence, Destination and Source address checks, and operational measures.

A good reference with an overview of security issues is RFC 4942, "IPv6 Transition/Coexistence Security Considerations."

Enterprise Security Models for IPv6

End-to-end transparency and security has been lost in many IPv4 networks due to the need to introduce NAT because of the shortage of IPv4 addresses. IPv6 can restore the transparency. However, some people have become used to seeing NAT and private addressing schemes to provide security in enterprise networks by hiding the network topology from the outside. These people may perceive the IPv6 transparency as a threat to their network and may even plan to deploy IPv6 networks with private local addressing schemes and translators only for this reason.

The goal with IPv6 is to restore end-to-end connectivity by using the abundant address space. To secure an IPv6 network, a security concept has to be created and the security mechanisms have to be implemented. NAT should no longer be used with IPv6. If hiding network topology from the outside is a requirement, other mechanisms should be used, such as privacy extensions for Stateless Address Autoconfiguration (RFC 4941), Unique Local Addresses (ULAs, RFC 4193), or untraceable IPv6 addresses as described in RFC 4864, "Local Network Protection for IPv6."

The New Model

In IPv4 networks, the common model for security is to have perimeter firewalls and integrate NATs. Applying this same approach in an IPv6 network may be a good starting point, but is limiting in the long term. In IPv6 networks, you should aim to design an improved security model that increases the overall security of the network but also facilitates end-to-end communication. IPv6 provides IPsec capability in each node. Relying on one perimeter firewall can be dangerous. An attacker who manages to get behind the firewall will usually find an open unsecured field. The optimal security concept for IPv6 networks is most likely to be "defense in depth," a combination of centralized security policy repositories and distribution mechanisms that, in conjunction with trusted hosts, will allow network managers to place more reliance on security mechanisms at the end points and allow end points to influence the behavior of perimeter firewalls. Perimeter firewalls will be responsible for securing the network from

general attacks, and the end node will be responsible for securing itself from node-related attacks. The new security policy model for IPv6/IPsec networks must be an identity-based model in order to separate security policy from network IDs. This is crucial for networks that want to allow for automation, autoconfiguration, and mobility without compromising security. This new distributed security model is emerging, and some of the technologies required are still under development, including protocols to allow end nodes to control and inform firewalls. Initial IPv6 deployments probably make use of similar firewall and intrusion detection techniques as used in today's IPv4 networks (with the exception of NATs, which should not be used at all in IPv6 networks). But the final goal to introduce a new type of distributed security concept should be kept in mind as you go along, and the development of these technologies should be followed closely.

Using Directory Services for Controlling Access

With the address architecture of IPv6, we have to learn to detach identity from IP address. An IPv6 interface has usually multiple IPv6 addresses and can change them frequently (depending on address plan). So having security rules and ACLs based on IPv6 addresses is probably not easy to manage. And a user (identity) is often accessing systems from different devices. One approach to take care of this is to have two different prefixes, one for access to the outside world, where we want changing interface IDs so that we cannot easily be tracked on the Internet, and another prefix for access to internal systems where we have fixed interface IDs, so our users are trackable. This other internal prefix could be the ULA prefix or just a slice of the public prefix set aside for internal use.

A new approach is to base rule definitions on identities in a directory service. After all, we want to control access for specific groups, specific machines, or maybe specific users, depending on what machine they are coming from. With IP-based access rules, this can become very complicated in a dual-stacked or IPv6 world. So, if we instead assigned rules based on identities, controlling access would be a lot easier, and the security device could resolve the rule by querying directory services.

It seems that the market is going in this direction. Leading firewall vendors, including Checkpoint and Cisco, are developing what is currently called *identity-based firewalls*. This connects the firewall with a directory service (often Active Directory) and matches users and machine identities. It can show IP addresses related to a user or machine name and lets you define rules based on any of these properties. You can define a firewall rule for specific users when they send traffic from specific machines, or a firewall rule for a specific user regardless of which machine he sends traffic from.

So, when you design your future network, allow yourself to think in new ways. Directory services are a great example where leveraging an existing, powerful tool can change the operational paradigm of the environment. Especially with the introduction of IPv6, there are many possible connections. For instance, using directory services and a well-

designed implementation of Identity Management, you can centralize access control to your DMZ, letting partners, customers, and employees access certain services and displaying content based on who they are.

IPv6 Firewall Filter Rules

When you live in a dual-stack network, you will have two security concepts: one for the IPv4 world and another for the IPv6 world. The two concepts should be congruent, but need to respect the requirements of each protocol. So, for instance, where administrators often choose to completely block ICMP traffic in IPv4 networks, it is not possible to do that with IPv6, because it may break some IPv6 features, such as Path MTU Discovery.

Without trying to provide a full-fledged security and firewall guide, here are some ideas for IPv6 security provisions and firewall filters that should be considered:

- Ingress filter at perimeter firewall for internally used addresses to prevent spoofing.
- Filter unneeded services at the perimeter firewall.
- Deploy host-based firewalls for a defense in depth.
- Critical systems should have static, nonobvious (randomly generated) IPv6 addresses, even at the expense of more complex operations. Consider using static neighbor entries for critical systems (versus letting them participate in ND).
- Hosts for Mobile IPv6 operations should be separate systems (to protect them by separate rules).
- Ensure that end nodes do not forward packets with Routing Extension type 0 headers.
- Layer 3 firewalls should never forward link-local addressed packets.
- Firewalls should support filtering based on Source and Destination address, IPv6 Extension headers, and upper-layer protocol information.
- Check your network for external packets that did not enter through your main perimeter firewall as an indication of backdoor connections or surreptitious tunneling.

In IPv6 networks, ICMPv6 plays a fundamental role and provides great functionality. Uncontrolled forwarding of ICMP messages also creates security risks. RFC 4890, "Recommendations for Filtering ICMPv6 Messages in Firewalls," provides recommendations for the configuration of ICMPv6 firewall filtering rules (specifically, allowing the forwarding of messages that are important for the functioning of the network and dropping messages that are potential security risks).

Another draft is on the way, called "Requirements for IPv6 Enterprise Firewalls" (*draft-gont-opsec-ipv6-firewall-reqs-01*). It is in an early stage at the time of writing. Its goal is to initialize a more in-depth discussion about what the requirements for IPv6 firewalls

are. Not only for the network operator who must write requests for proposal, but also for firewall vendors, to help them identify the important options they need to implement. If you operate, buy, deploy, or manage firewalls, or if you are a firewall vendor, I highly recommend going through the lists of requirements in this draft. I assume that it will become an RFC in the near future.

For those of you who work in security, you will want to dive a lot deeper than a security chapter in a general IPv6 book can cover. The best advice I can give you is to take your time to plan your IPv6 security concept and to dive into the topics using a lab and testing it all. Most vendors by today (2014) have some type of security implementations, but as the real-world implementations are still in an early stage, there is not enough feedback from the market right now. So most implementations are not in a very mature state.

A good read on IPv6 security is the book *IPv6 Security* by Scott Hogg and Eric Vyncke (Cisco Press), two renowned industry experts. It's been a while since it's been published and I am hoping for a second edition soon, but I consider it a must-read for IPv6 security people.

The next chapter describes integration and transition mechanisms and scenarios that allow for a smooth, step-by-step introduction of IPv6 into your network. It also covers important aspects of the planning and design process as well as lessons learned by organizations who have already done the planning.

References

Here's a list of the most important RFCs and drafts mentioned in this chapter. Sometimes I include additional subject-related RFCs for your personal further study.

RFCs

- RFC 1828, "IP Authentication using Keyed MD5," 1995
- RFC 1829, "The ESP DES-CBC Transform," 1995
- RFC 1918, "Address Allocation for Private Internets," 1996
- RFC 2085, "HMAC-MD5 IP Authentication with Replay Prevention," 1997
- RFC 2403, "The Use of HMAC-MD5-96 within ESP and AH," 1998
- RFC 2404, "The Use of HMAC-SHA-1-96 within ESP and AH," 1998
- RFC 2405, "The ESP DES-CBC Cipher Algorithm With Explicit IV," 1998
- RFC 2407, "The Internet IP Security Domain of Interpretation for ISAKMP," 1998
- RFC 2408, "Internet Security Association and Key Management Protocol (ISAKMP)," 1998

- RFC 2409, "The Internet Key Exchange (IKE)," 1998
- RFC 2410, "The NULL Encryption Algorithm and Its Use With IPsec," 1998
- RFC 2412, "The OAKLEY Key Determination Protocol," 1998
- RFC 2451, "The ESP CBC-Mode Cipher Algorithms," 1998
- RFC 2474, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," 1998
- RFC 3056, "Connection of IPv6 Domains Via IPv4 Clouds," 2001
- RFC 3068, "An Anycast Prefix for 6to4 Relay Routers," 2001
- RFC 3168, "The Addition of Explicit Congestion Notification (ECN) to IP," 2001
- RFC 3260, "New Terminology and Clarification for DiffServ," 2002
- RFC 3493, "Basic Socket Interface Extensions for IPv6," 2003
- RFC 3526, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)," 2003
- RFC 3602, "The AES-CBC Cipher Algorithm and Its Use with IPsec," 2003
- RFC 3631, "Security Mechanisms for the Internet," 2003
- RFC 3715, "IPsec-Network Address Translation (NAT) Compatibility Requirements," 2004
- RFC 3739, "Internet X.509 Public Key Infrastructure: Qualified Certificates Profile," 2004
- RFC 3740, "The Multicast Group Security Architecture," 2004
- RFC 3748, "Extensible Authentication Protocol (EAP)," 2004
- RFC 3754, "IP Multicast in Differentiated Services (DS) Networks," 2004
- RFC 3756, "IPv6 Neighbor Discovery (ND) Trust Models and Threats," 2004
- RFC 3765, "NOPEER Community for Border Gateway Protocol (BGP) Route Scope Control," 2004
- RFC 3947, "Negotiation of NAT-Traversal in the IKE," 2005
- RFC 3948, "UDP Encapsulation of IPsec ESP Packets," 2005
- RFC 3964, "Security Considerations for 6to4," 2004
- RFC 3971, "Secure Neighbor Discovery," 2005
- RFC 3972, "Cryptographically Generated Addresses (CGA)," 2005
- RFC 4033, "DNS Security Introduction and Requirements," 2005
- RFC 4035, "Protocol Modifications for the DNS Security Extensions," 2005
- RFC 4106, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)," 2005

- RFC 4107, “Guidelines for Cryptographic Key Management,” 2005
- RFC 4109, “Algorithms for Internet Key Exchange version 1 (IKEv1),” 2005
- RFC 4285, “Authentication Protocol for Mobile IPv6,” 2005
- RFC 4301, “Security Architecture for the Internet Protocol,” 2005
- RFC 4302, “IP Authentication Header,” 2005
- RFC 4303, “IP Encapsulating Security Payload (ESP),” 2005
- RFC 4307, “Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2),” 2005
- RFC 4308, “Cryptographic Suites for IPsec,” 2005
- RFC 4309, " Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP),” 2005
- RFC 4359, “The Use of RSA/SHA-1 Signatures within Encapsulating Security Payload (ESP) and Authentication Header (AH),” 2006
- RFC 4380, “Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs),” 2006
- RFC 4581, “Cryptographically Generated Addresses (CGA) Extension Field Format,” 2006
- RFC 4835, “Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH),” 2007
- RFC 4862, “IPv6 Stateless Address Autoconfiguration,"2007
- RFC 4864, “Local Network Protection for IPv6,” 2007
- RFC 4890, “Recommendations for Filtering ICMPv6 Messages in Firewalls,” 2007
- RFC 4891, “Using IPsec to Secure IPv6-in-IPv4 Tunnels,” 2007
- RFC 4941, “Privacy Extensions for Stateless Address Autoconfiguration in IPv6,” 2007
- RFC 4942, “IPv6 Transition/Coexistence Security Considerations,” 2007
- RFC 4982, “Support for Multiple Hash Algorithms in Cryptographically Generated Addresses (CGAs),” 2007
- RFC 5084, “Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS),” 2007
- RFC 5095, “Deprecation of Type 0 Routing Headers in IPv6,” 2007
- RFC 5157, “IPv6 Implications for Network Scanning,” 2008
- RFC 5247, “Extensible Authentication Protocol (EAP) Key Management Framework,” 2008
- RFC 5722, “Handling of Overlapping IPv6 Fragments,” 2009

- RFC 5739, "IPv6 Configuration in Internet Key Exchange Protocol Version 2 (IKEv2)," 2010
- RFC 5909, "Securing Neighbor Discovery Proxy: Problem Statement," 2010
- RFC 5991, "Teredo Security Updates," 2010
- RFC 5996, "Internet Key Exchange Protocol Version 2 (IKEv2)," 2010
- RFC 6014, "Cryptographic Algorithm Identifier Allocation for DNSSEC," 2010
- RFC 6040, "Tunneling of Explicit Congestion Notification," 2010
- RFC 6071, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," 2011
- RFC 6081, "Teredo Extensions," 2011
- RFC 6092, "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service," 2011
- RFC 6104, "Rogue IPv6 Router Advertisement Problem Statement," 2011
- RFC 6105, "IPv6 Router Advertisement Guard," 2011
- RFC 6151, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms," 2011
- RFC 6169, "Security Concerns with IP Tunneling," 2011
- RFC 6273, "The Secure Neighbor Discovery (SEND) Hash Threat Analysis," 2011
- RFC 6324, "Routing Loop Attack Using IPv6 Automatic Tunnels," 2011
- RFC 6434, "IPv6 Node Requirements," 2011
- RFC 6494, "Certificate Profile and Certificate Management for SEcure Neighbor Discovery (SEND)," 2012
- RFC 6495, "Subject Key Identifier (SKI) SEcure Neighbor Discovery (SEND) Name Type Fields," 2012
- RFC 6620, "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses," 2012
- RFC 6655, "AES-CCM Cipher Suites for Transport Layer Security (TLS)," 2012
- RFC 6946, "Processing of IPv6 'Atomic' Fragments," 2013
- RFC 6959, "Source Address Validation Improvement (SAVI) Threat Scope," 2013
- RFC 6980, "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery," 2013
- RFC 7039, "Source Address Validation Improvement (SAVI) Framework," 2013
- RFC 7112, "Implications of Oversized IPv6 Header Chains," 2014

- RFC 7113, "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)," 2014
- RFC 7123, "Security Implications of IPv6 on IPv4 Networks," 2014
- RFC 7219, "SEcure Neighbor Discovery (SEND) Source Address Validation Improvement (SAVI)," 2014

Drafts

Drafts can be found at *http://www.ietf.org/ID.html*. To locate the latest version of a draft, refer to *https://datatracker.ietf.org/public/pidtracker.cgi*. You can enter the draft name without a version number and the most current version will come up. If a draft does not show up, it was possibly deleted. If it was published as an RFC, the RFC number will be displayed. *http://tools.ietf.org/wg* is also a very useful site. More information on the process of standardization, RFCs, and drafts can be found in the Appendix A.

Here's a list of drafts I refer to in this chapter, as well as interesting drafts that relate to the topics in this chapter:

"SAVI Solution for DHCP"
: *draft-ietf-savi-dhcp-25*, binding table for DHCP assigned addresses

"SAVI for Mixed Address Assignment Methods Scenario"
: *draft-ietf-savi-mix-06*, binding table for coexisting ND, DHCP, and SEND addresses

"Requirements for IPv6 Enterprise Firewalls"
: *draft-gont-opsec-ipv6-firewall-reqs-01*

CHAPTER 7
Transition Technologies

This chapter provides an overview of the various transition mechanisms that are available. IPv6 and IPv4 will coexist for many years, and there are a wide range of techniques that make coexistence possible and provide an easy transition. It is important to make the right choices and find the best migration path. There is not an easy one-size-fits-all strategy. The migration path has to be adjusted to the individual requirements of each organization and network.

The available techniques that support you in your transition are separated into three main categories:

Dual-stack techniques
: Allow IPv4 and IPv6 to coexist in the same devices and networks

Tunneling techniques
: Allow the transport of IPv6 traffic over the existing IPv4 infrastructure

Translation techniques
: Allow IPv6-only nodes to communicate with IPv4-only nodes

These techniques can and likely will be used in combination with one another. The migration to IPv6 can be done step-by-step, starting with single hosts or subnets. You can migrate your corporate network or parts of it while your ISP still runs only IPv4, or your ISP can upgrade to IPv6 while your corporate network still runs IPv4. This first section describes the techniques available today for each of these categories. RFC 4213, "Basic Transition Mechanisms for IPv6 Hosts and Routers," describes the dual-stack technique and configured tunneling. There has been a lot of change in this area; transition mechanisms that were key technologies at some point (such as 6to4) have been replaced by newer mechanisms, such as 6rd in the case of 6to4. New technologies have been defined and even more are soon to come. This is the fulfillment of the promise that the IETF will develop new and adjusted transition mechanisms to provide solutions for enterprises as the need becomes obvious.

Dual-Stack

A *dual-stack node* has complete support for both protocol versions. This type of node is often referred to as an *IPv6/IPv4 node*. In communication with an IPv6 node, such a node behaves like an IPv6-only node; in communication with an IPv4 node, it behaves like an IPv4-only node. Implementations may allow you to enable or disable one of the stacks, so this node type can have three modes of operation. When the IPv4 stack is enabled and the IPv6 stack is disabled, the node behaves like an IPv4-only node. When the IPv6 stack is enabled and the IPv4 stack disabled, it behaves like an IPv6-only node. When both the IPv4 and IPv6 stacks are enabled, the node can use both protocols. An IPv6/IPv4 node has at least one address for each enabled protocol version. It uses IPv4 mechanisms to be configured for an IPv4 address (static configuration or DHCP) and uses IPv6 mechanisms to be configured for an IPv6 address (static configuration, SLAAC, or DHCPv6).

DNS is used with both protocol versions to resolve names and IP addresses. An IPv6/IPv4 node needs a DNS resolver that is capable of resolving both types of DNS address records. The DNS A record represents IPv4 addresses, and the DNS AAAA (referred to as quad-A) record represents IPv6 addresses.

Depending on how a service is reachable (over IPv4 or IPv6 or both), DNS may return only an IPv4 or only an IPv6 address, or both. Default address selection mechanisms and profiles will have to ensure that connections can be established efficiently in any case. Hopefully, both the DNS resolver on the client and an application using DNS will have configuration options that let us specify orders or filters of how to use the addresses (i.e., preferred protocol settings). Happy Eyeballs is a specification that has been developed to optimize connection setup in a dual-stacked world. Note that the DNS resolver may run over an IPv4 or IPv6 network to resolve either type of address record.

For a detailed discussion of IPv6 DNS and Happy Eyeballs, refer to Chapter 5.

A *dual-stack network* is an infrastructure in which both IPv4 and IPv6 forwarding is enabled on routers. The advantage of dual-stack is that you run both protocols in native mode. Once the infrastructure runs dual-stack you can start to migrate applications from IPv4 to IPv6 as they become available. And your traffic shifts from IPv4 to IPv6 smoothly. No tunneling and no translation is involved. This is best for performance, scalability, and efficiency. And there is no need to design, test, and deploy temporary transition mechanisms that need to be removed later on.

For this technique you must perform a full network upgrade to run the two separate protocol stacks. All tables (e.g., routing tables) are kept simultaneously with routing protocols being configured for both protocols. For security, you need two concepts, one for IPv4 and one for IPv6, as you have two inroads into your network. For network management, on some operating systems you may still have separate commands depending on the protocol (e.g., *ping* for IPv4 and *ping6* for IPv6), and it takes more memory and CPU power. But with state-of-the-art hardware, this should not be an issue and the advantages of dual-stack outweigh the disadvantages by far.

Dual-stack is a good way to go for IPv6 deployment. But it requires that IPv4 addresses are available for all hosts. If that is not the case, IPv6-only options such as NAT64, DS-Lite, MAP, or 464XLAT have to be used. These methods decouple edge network growth from IPv4 availability. They are discussed later in this chapter.

Another aspect to keep in mind is that dual-stack makes troubleshooting problems more complicated. For instance, did an application that has problems with IPv6 attempt to connect via IPv4 instead of IPv6 and fail? How do you have to adjust your troubleshooting approaches to test and figure that out? Your helpdesk and IT support staff also need to understand how to use specific tools for IPv4 and IPv6 so they can rule out one protocol versus the other. So from an operational and support perspective, it may cost more to run a dual-stack network. This is one of the main reasons more and more enterprises consider migrating to an IPv6-only infrastructure as soon as possible. But more about this in Chapter 9.

Tunneling Techniques

Tunneling mechanisms can be used to deploy an IPv6 forwarding infrastructure while the overall IPv4 infrastructure is still the basis and either should not or cannot be modified or upgraded.

Tunneling is also called *encapsulation*. With encapsulation, one protocol (in our case, IPv6) is encapsulated in the header of another protocol (in our case, IPv4) and forwarded over the infrastructure of the second protocol (IPv4). The process of encapsulation has three components:

- Encapsulation at the tunnel entry point
- Decapsulation at the tunnel exit point
- Tunnel management

So tunneling can be used to carry IPv6 traffic by encapsulating it in IPv4 packets and tunneling it over the IPv4 routing infrastructure. For instance, if your provider still has

an IPv4-only infrastructure, tunneling allows you to have a corporate IPv6 network and tunnel through your ISP's IPv4 network to reach other IPv6 hosts or networks. Or you can deploy IPv6 islands in your corporate network while the backbone is still IPv4. IPv6 packets traveling from one IPv6 island to another can traverse the backbone encapsulated in IPv4 packets. General tunneling techniques and the encapsulation of IPv6 packets in IPv4 packets are defined in several RFCs, such as RFC 2473, "Generic Packet Tunneling in IPv6 Specification," and RFC 4213, "Basic Transition Mechanisms for IPv6 Hosts and Routers." We differentiate two general types of tunneling:

Manually configured tunneling of IPv6 over IPv4
: IPv6 packets are encapsulated in IPv4 packets to be carried over IPv4 routing infrastructures. These are point-to-point tunnels that need to be configured manually.

Automatic tunneling of IPv6 over IPv4
: IPv6 nodes can use different types of addresses, such as 6to4, 6rd, or ISATAP addresses, to dynamically tunnel IPv6 packets over an IPv4 routing infrastructure. These special IPv6 unicast addresses carry an IPv4 address in some parts of the IPv6 address fields, which can be used to determine the IPv4 address of the destination or the tunnel endpoint, respectively.

How Tunneling Works

The concepts discussed in this section apply to tunneling in general. The next two paragraphs discuss the difference between configured tunnels and automatic tunneling. Figure 7-1 shows two IPv6 networks connected through an IPv4-only network.

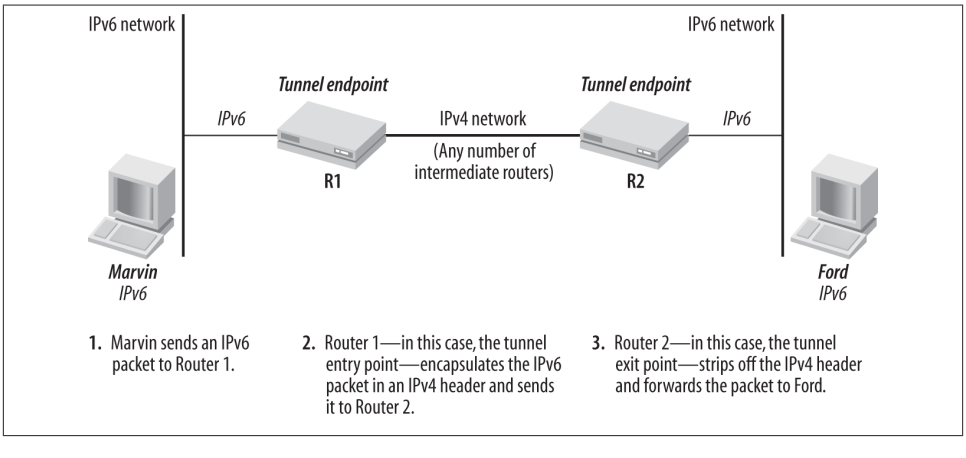

Figure 7-1. Encapsulation and tunneling

Host Marvin is on an IPv6 network and wants to send an IPv6 packet to host Ford on another IPv6 network. The network between router R1 and router R2 is an IPv4-only

network. Router R1 is the *tunnel entry point*. Marvin sends the IPv6 packet to router R1 (step 1 in Figure 7-1). When router R1 receives the packet addressed to Ford, it encapsulates the packet in an IPv4 header and forwards it to router R2 (step 2 in Figure 7-1), which is the *tunnel exit point*. Router R2 decapsulates the packet and forwards it to its final destination (step 3 in Figure 7-1). Between R1 and R2, any number of IPv4 routers is possible.

A tunnel has two endpoints: the tunnel entry point and the tunnel exit point. In the scenario in Figure 7-1, the tunnel end points are two routers, but the tunnel can be configured in different ways. It can be set up router-to-router, host-to-router, host-to-host, or router-to-host. Depending on which scenario is used, the tunnel entry and exit point can be either a host or a router.

The steps for the encapsulation of the IPv6 packet are the following:

1. The entry point of the tunnel decrements the IPv6 hop limit by one, encapsulates the packet in an IPv4 header, and transmits the encapsulated packet through the tunnel. If necessary, the IPv4 packet is fragmented.
2. The exit point of the tunnel receives the encapsulated packet. It checks whether the source of the packet (tunnel entry point) is an acceptable source (according to its configuration). If the packet was fragmented, the exit point reassembles it. Then, the exit point removes the IPv4 header and processes the IPv6 packet to its original destination.

Figure 7-2 shows the encapsulation of an IPv6 packet in an IPv4 packet.

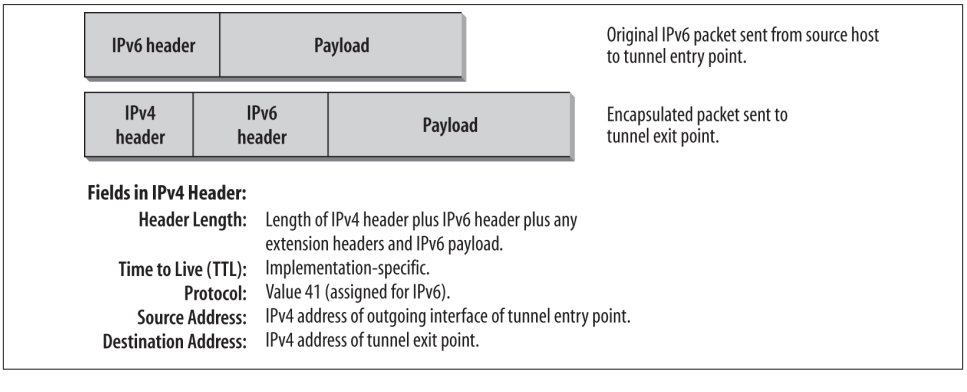

Figure 7-2. Encapsulation

The following fields in the IPv4 header are interesting to note: the Total Length field in the IPv4 header contains the length of the IPv4 header plus the length of the IPv6 packet, which is treated as the payload. If the encapsulated packet has to be fragmented, there will be corresponding values in the Flags and Fragment Offset fields. The value of the

Time to Live (TTL) field depends on the implementation used. The Protocol Number is set to 41, the value assigned for IPv6. Thus, if you want to analyze your tunneled IPv6 traffic, you can set a filter in your analyzer to display the packets containing the value 41 in the Protocol Number field. The IPv4 Source address is the address of the outgoing interface of the tunnel entry point. It should also be configurable for cases where automatic address selection may produce different results over time (multiple addresses/interfaces). The IPv4 Destination address is the IPv4 address of the tunnel exit point. The IPv6-over-IPv4 tunnel is considered a single hop. The Hop Limit field in the IPv6 header is therefore decremented by one. This hides the existence of a tunnel to the end user, and is not detectable by common tools such as *traceroute*. Figure 7-3 shows an encapsulated IPv6 packet in the trace file.

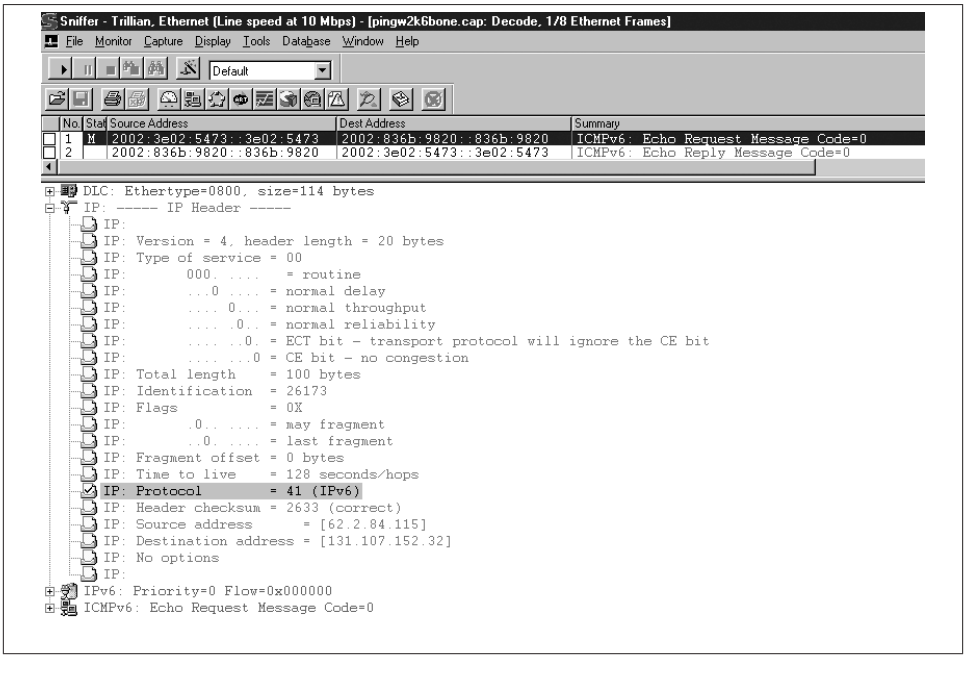

Figure 7-3. Encapsulation in the trace file

The Ethertype is set to 0800, the value for IPv4. The TTL in the IPv4 header is set to 128. The Protocol field shows value 41 for IPv6, which identifies this packet as an encapsulated packet. The Source address `62.2.84.115` is the IPv4 address of the tunnel entry point. The Destination address is the IPv4 address of a 6to4 relay router (explained later in this chapter) in the Internet, the tunnel exit point. This router can forward IPv6 packets to an IPv6 network. Compare these IPv4 addresses with the IPv6 Source and Destination addresses (which can be seen in the highlighted summary line above the detail screen). Use your Windows calculator to find out that the IPv6 Source and

Destination addresses have the 6to4 prefix of `2002` plus the IPv4 address in hexadecimal notation in the low-order 32 bits. This is an example of a host-to-host automatic tunnel because we were actually pinging the 6to4 router.

If an IPv4 router from within the tunnel generates an ICMPv4 error message, the router sends the message to the tunnel entry point because that host is the source of that packet. If the packet contains enough information about the original encapsulated IPv6 packet, the tunnel entry point may send an ICMPv6 message back to the original source of the packet.

When the tunnel exit point receives an IPv4 datagram with a protocol value of 41, it knows that this packet has been encapsulated. Before forwarding a decapsulated IPv6 packet, the tunnel endpoint must verify that the tunnel Source address is acceptable. Thus, unacceptable ingress into the network can be avoided. If the tunnel is a bidirectionally configured tunnel, this check is done by comparing the Source address of the encapsulated packet with the configured address of the other side of the tunnel. For unidirectionally configured tunnels, the tunnel must be configured with a list of source IPv4 address prefixes that are acceptable. By default, this list is empty, which means that the tunnel endpoint has to be explicitly configured to allow forwarding of decapsulated packets. In the case of fragmentation, the tunnel exit point reassembles the packets and removes the IPv4 header. Before delivering the IPv6 packet to the final destination, it checks to see if the IPv6 Source address is valid. The following Source addresses are considered invalid:

- All multicast addresses (`ff00::/8`)
- The loopback address (`::1`)
- All IPv4-compatible IPv6 addresses (`::/96`), excluding the unspecified address for Duplicate Address Detection (`::/128`)
- All IPv4-mapped IPv6 addresses (`::ffff:0:0/96`)

Both tunnel endpoints need to have a link-local IPv6 address. The IPv4 address of that same interface may be the interface identifier for the IPv6 address. For example, a host with an IPv4 address of `192.168.0.2` may have a link-local address of `fe80::192.168.0.2/64.`

The specification contains rules that apply tunnel Source address verification and ingress filtering (RFCs 2827 and 3704) in general to packets before they are decapsulated. If further security mechanisms are desirable, a tunneling scheme with authentication can be used—for example, IPsec (preferable) or Generic Routing Encapsulation (GRE) with a preconfigured secret key (RFC 2890). Since the configured tunnels are set up manually, setting up the keying material is not a problem.

Automatic Tunneling

Automatic tunneling allows IPv6/IPv4 nodes to communicate over an IPv4 infrastructure without the need for tunnel destination preconfiguration. The tunnel configuration information is extracted from an IPv4 address embedded in an IPv6 address. In a previous specification (RFC 2893, obsoleted by RFC 4213), the tunnel endpoint address was determined by an IPv4-compatible Destination address. RFC 4213 removes the description of automatic tunneling and IPv4-compatible addresses and refers to 6to4 (discussed later in this chapter), which does not use IPv4-compatible IPv6 addresses. 6to4 has its own IPv6 address format, which includes the IPv4 address of the tunnel endpoint in the prefix and therefore allows for automatic tunneling. These days, 6to4 is not a recommended tunnel mechanism anymore; it has been replaced by 6rd. But more about this in the sections 6to4 and 6rd later in this chapter.

Configured Tunneling (RFC 4213)

Configured tunneling is IPv6-over-IPv4 tunneling where the IPv4 tunnel endpoint addresses are determined by configuration information on the tunnel endpoints. All tunnels are assumed to be bidirectional. The tunnel provides a virtual point-to-point link to the IPv6 layer using the configured IPv4 addresses as the lower layer endpoint addresses. The administrative work to manage configured tunnels is higher than with automatic tunnels, but for security reasons it may be desirable, as it provides more possibilities to control the forwarding path of IPv6 packets.

RFC 4213 discusses the configuration and issues to be taken care of, such as determination of valid tunnel endpoint addresses (ingress filtering), how to deal with ICMPv4 or ICMPv6 messages, how to optimize tunnel MTU sizes, fragmentation, the header fields, Neighbor Discovery (ND) over tunnels, and security considerations.

IPv6/IPv4 hosts connected to network segments with no IPv6 routers can be configured with a static route to an IPv6 router in the Internet at the other side of an IPv4 tunnel; this enables communication with a remote IPv6 world. In this case, the IPv6 address of an IPv6/IPv4 router at the other end of the tunnel is added into the routing table as a default route. Now all IPv6 Destination addresses match the route and can be tunneled through the IPv4 infrastructure. This default route has a mask of zero and is used only if there are no other routes with a more specific matching mask.

Encapsulation in IPv6 (RFC 2473)

RFC 2473 specifies the model and the generic mechanisms for encapsulation with IPv6. Most of the rules discussed in this chapter about tunneling in IPv4 apply to tunneling in IPv6. The main difference is that in tunneling in IPv6, the packets are encapsulated in an IPv6 header and sent through an IPv6 network. The packet being encapsulated can be an IPv4 packet, an IPv6 packet, or any other protocol. The tunnel entry

point prepends the IPv6 header and, if needed, one or a set of Extension headers in front of the original packet header. Whatever the tunnel entry point prepends are called the *Tunnel IPv6 headers*. Figure 7-4 shows the Tunnel IPv6 headers in the packet view.

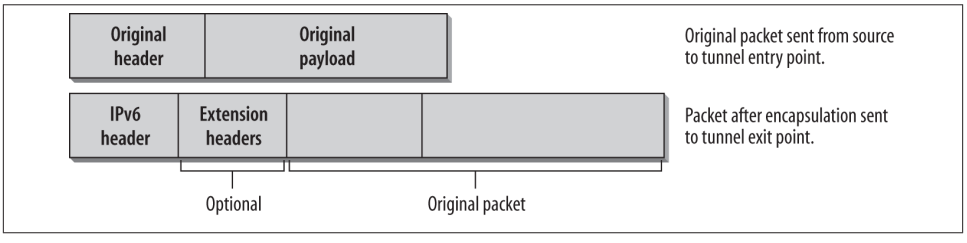

Figure 7-4. Tunnel IPv6 headers from the packet view

In the IPv6 header applied by the tunnel entry point, the Source address is the address of the tunnel entry point node, and the Destination address is the address of the tunnel exit point node. The source node of the original packet can be the same node as the tunnel entry point. The original packet, including its header, becomes the payload of the encapsulated packet. The header of the original packet is treated according to standard forwarding rules. If the header is an IPv4 header, the TTL field is decremented by one. If it is an IPv6 header, the Hop Limit field is decremented by one. The network between the tunnel entry point and the tunnel exit point is thus virtually just one hop, no matter how many actual hops there are in between.

The Tunnel IPv6 header is processed according to the IPv6 protocol rules. Extension headers, if added, are processed as though the packet were a standard IPv6 packet. For example, a Hop-by-Hop Options header would be processed by every node listed in the Hop-by-Hop Options field. A Destination Options header would be processed by the destination host—i.e., the tunnel exit point. All these options are configured on the tunnel entry point. An example of the use of a Destination Options header is the configuration of a Tunnel Encapsulation Limit Option (RFC 2473). This option may be used when tunnels are nested. One hop of a tunnel can be the entry point of another tunnel. In this case, we have *nested tunnels*. The first tunnel is called the *outer tunnel*, and the second tunnel is called the *inner tunnel*. The inner tunnel entry point treats the whole packet received from the outer tunnel as the original packet and applies the same rules as shown in Figure 7-6. The only natural limit to the number of nested tunnels is the maximum IPv6 packet size. Every encapsulation adds to the size of the tunnel IPv6 headers. This would allow for something around 1,600 nested tunnels, which is not realistic. Also, consider the case in which the packet has to be fragmented. If it has to be fragmented again because the additional tunnel IPv6 headers have increased the packet size, the number of fragments is doubled. So a mechanism was needed to limit the number of nested tunnels. It is specified in RFC 2473 and is called the Tunnel

Encapsulation Limit Option. This option is carried in a Destination Option header and has the format shown in Figure 7-5.

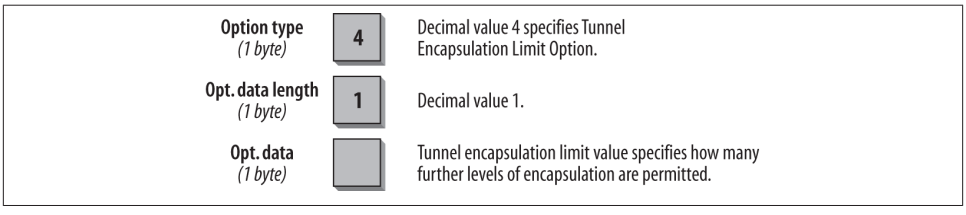

Figure 7-5. Format of the Tunnel Encapsulation Limit Option

The Option Type field has 1 byte and the decimal value 4, specifying the Tunnel Encapsulation Limit Option. The Option Data Length field has the decimal value 1, specifying the length of the following Option field. In this case, the Option field has a size of 1 byte and contains the actual value for the Tunnel Encapsulation Limit Option. The value in this field specifies how many further levels of encapsulation are permitted. If the value is zero, the packet is discarded and an ICMP Parameter Problem message is sent back to the source (the tunnel entry point of the previous tunnel). If the value is nonzero, the packet is encapsulated and forwarded. In this case, a new Tunnel Encapsulation Limit Option has to be applied with a value of one less than the limit received in the packet being encapsulated. If the packet received does not have a tunnel encapsulation limit, but this tunnel entry point has one configured, the tunnel entry point must apply a destination options header and include the configured value.

Loopback encapsulation should be avoided. Loopback encapsulation happens when a node encapsulates a packet originating from itself and destined to itself. IPv6 implementations should prevent this by checking and rejecting configurations of tunnels where both the entry and exit points belong to the same host. Another undesirable situation is a *routing-loop nested encapsulation*. This situation happens if a packet from an inner tunnel reenters an outer tunnel from which it has not yet exited. This can be controlled only by a combination of the original packet's hop limit and the configuration of tunnel encapsulation limits.

Let's have a closer look at a Tunnel IPv6 Header in Figure 7-6.

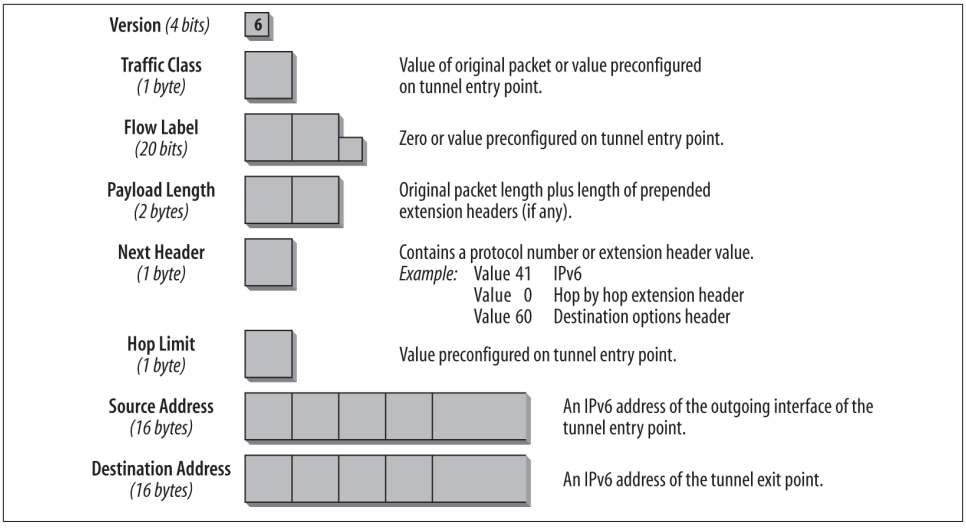

Figure 7-6. The Tunnel IPv6 header

The fields of a standard IPv6 header were discussed in Chapter 3. Interesting values here are the following: the values for Traffic Class, Flow Label, and Hop Limit can be preconfigured on the tunnel entry point. The Payload Length has the value for the packet length of the original packet plus the size of any Extension headers prepended by the tunnel entry point. The Source and Destination Addresses of the Tunnel IPv6 header contain the IPv6 addresses of the tunnel entry and exit points, respectively. Note that a host configured as a tunnel entry point must support fragmentation of packets that it encapsulates. Encapsulated packets may exceed the Path MTU of the tunnel. Because the tunnel entry point is considered the source of the encapsulated packet, it must fragment it if needed. The tunnel exit point node will reassemble the packet. If the original packet is an IPv4 packet with the Don't Fragment Bit set, the tunnel entry point discards the packet and sends an ICMP Destination Unreachable message with the code "fragmentation needed and DF set" back to the source of the packet.

Tunneling Mechanisms

The next sections describe a set of tunneling mechanisms available today. They are to be seen as a set of tools in your IPv6 toolbox. Analyze your environment and your requirements to find the optimal tools or combination of tools that meet your goals.

6to4

RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds," specifies a mechanism for IPv6 sites to communicate with each other over the IPv4 network without explicit tunnel setup. This mechanism is called *6to4* and is one of the automatic tunneling mechanisms.

The wide area IPv4 network is treated as a unicast point-to-point link layer, and the native IPv6 domains communicate via 6to4 routers, also referred to as 6to4 gateways. Note that only the 6to4 router needs to be 6to4 aware. No changes have to be made to the hosts within the 6to4 network.

This was intended as a transition mechanism used during the period of coexistence of IPv4 and IPv6. It will not be used as a permanent solution. 6to4 will also not be one of the strategic tunnel mechanisms. There are some disadvantages in the design that will be discussed later in this section. We still discuss the mechanism as it has not been deprecated, because there is a newer mechanism called *6rd*, which is based on 6to4 and is implemented in production networks. 6rd resolves some of the disadvantages of 6to4 and is also discussed later in this chapter. So back to 6to4.

The IPv6 packets are encapsulated in IPv4 at the 6to4 gateway. At least one globally unique IPv4 unicast address is required for this configuration. The IANA has assigned a special prefix for the 6to4 scheme: `2002::/16`. Figure 7-7 shows the format of the 6to4 prefix in detail.

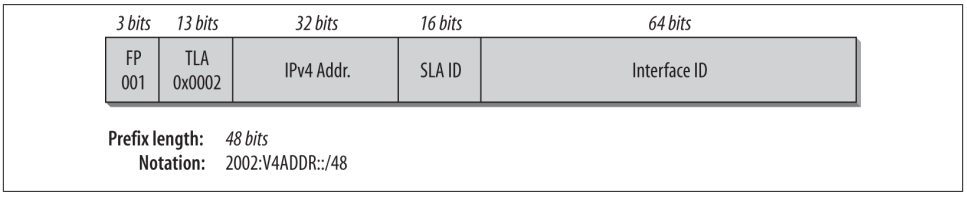

Figure 7-7. Format of the 6to4 prefix

The 32 bits after the prefix `2002::/16` are the IPv4 address of the gateway in hex representation. This leaves you with 80 bits of address space for your internal network. 16 bits are used for the local network addressing, so you can create 65,536 networks! The remaining 64 bits are used for the interface identifier of the nodes on your network; that is, 2^{64} nodes per subnet. It looks like getting familiar with the extended address space has some advantages. Now all the hosts on your network can communicate with other 6to4 hosts on the Internet.

In Figure 7-7, there are terms such as FP (Format Prefix), TLA (Top Level Aggregator), and SLA (Site Level Aggregator). They come from an older IPv6 address architecture specification (RFC 2374). At the time when 6to4 was specified, RFC 2374 was still valid.

When a node in a 6to4 network wants to communicate with a node in another 6to4 network, no tunnel configuration is necessary (automatic tunneling). The tunnel entry point takes the IPv4 address of the tunnel exit point from the IPv6 address of the

destination. To communicate with an IPv6 node in a remote IPv6 network (which is not a 6to4 network), you need a *6to4 relay router*. The relay router is a router configured for 6to4 and IPv6. It connects your 6to4 network to the native IPv6 network. It announces the 6to4 prefix of `2002::/16` into the native IPv6 network.

Figure 7-8 shows the 6to4 components and how they play together.

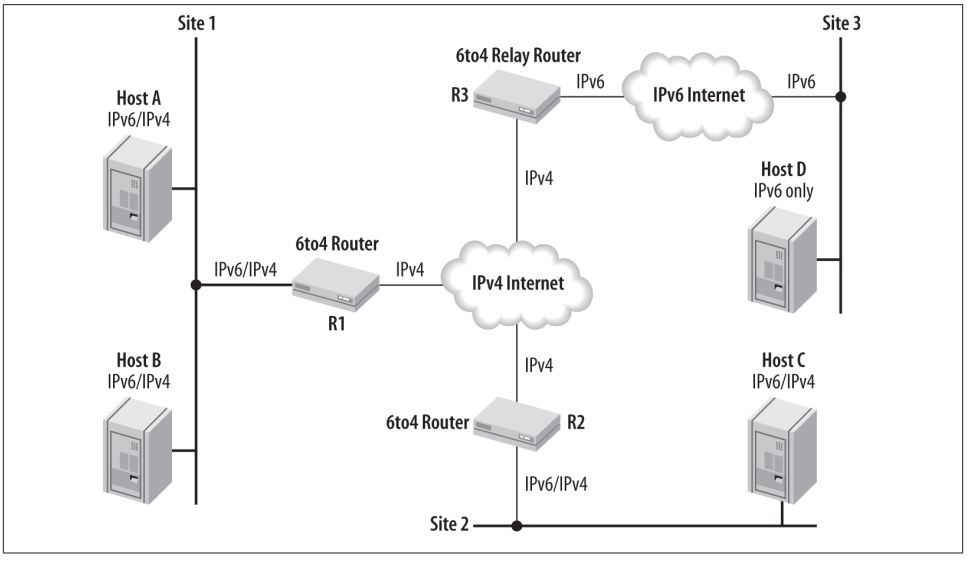

Figure 7-8. 6to4 components

The figure shows the different possible communication paths. Within site 1, hosts A and B can communicate using IPv6. To communicate with host C in site 2 (another 6to4 site), the packets are sent to router R1 in site 1. Router R1 encapsulates them in IPv4 and forwards them to Router R2 in site 2. Router R1 learns the IPv4 address of Router R2 from the IPv6 Destination address. Router R2 decapsulates the packet and forwards the original IPv6 packet to host C. To communicate with an IPv6-only host in the Internet, host A or B sends its IPv6 packets to Router R1. Router R1 encapsulates them in IPv4 and forwards them to the Relay Router R3. Router R3 decapsulates the packet and forwards the original IPv6 packet over the IPv6 routing infrastructure to host D.

Router R1 internally advertises the 6to4 prefix in its Router Advertisements (if configured to do so). The IPv6 hosts in site 1 can use the RA for Stateless Address Autoconfiguration of their IPv6 address. The prefix announced has the format `2002:IPv4-address-R1:subnet-ID::/64`.

To connect a 6to4 network with the IPv6 Internet, a convenient 6to4 relay can be evaluated and manually configured. The manual configuration has the advantage of

providing control over the relays used but creates more administrative work. In case the preconfigured relay is not available, another relay needs to be configured.

RFC 3068 defines a 6to4 relay router anycast address to simplify the configuration of 6to4 gateways that need a default route to find a 6to4 relay router on the Internet. IANA assigned an IPv4 6to4 Relay anycast prefix of `192.88.99.0/24`. The assigned anycast address corresponds to the first node in the prefix, e.g., `192.88.99.1`. The 6to4 routers have to be configured with a default route pointing to this anycast address. Using this address means that 6to4 packets are routed to the nearest available 6to4 relay router automatically. If one 6to4 relay goes down, you do not need to reconfigure your 6to4 gateway; packets will automatically be rerouted to the next available relay.

As mentioned before, 6to4 has some problems. Deployments showed that there are several cases where communication failures can occur. These failures can lead to long retry delays or to total communication failures for users trying to access services. The users are usually not aware of the fact that they use 6to4 and blame the applications and services for the failure. There were discussions going on to move 6to4 to historic status, but this would not eliminate the multitude of implementations out in the market. As 6to4 was one of the early and main transition technologies, it has been implemented widely. RFC 6343, "Advisory Guidelines for 6to4 Deployment," describes the observed issues and gives recommendations for 6to4 deployment. In fact, implementations on many hosts and consumer routers now de-preference or eliminate 6to4 such that websites that enable native IPv6 see very little 6to4 traffic in practice today. You can see this in the Google Statistics, where 6to4 and Teredo traffic is displayed in the red curve, which is down to zero basically since 2011. The green curve that increases constantly is native IPv6 traffic.

IPv6 Rapid Deployment—6rd

Let me give you some background on the story behind 6rd before we dive into the workings of it. I kept preaching that providers and vendors should not wait until customers ask for IPv6, because customers would usually not care about transport protocols, but rather about services. So why should they ask for IPv6? But then the exception happened in France. Back in 2007, customers of Free (a French ISP belonging to the Iliad Group) made a request to Free, initiated by Remi Despres. More than 20,000 people signed a petition that they would pay one Euro more per month for getting IPv6 services. Within the incredibly short period of five weeks, Free provided IPv6 Internet access to 1.5 million users. They started from zero and did the following:

1. Obtained a /32 IPv6 prefix from their Regional Internet Registry (RIR)
2. Added 6rd support to the software of their Freebox home-gateway (based on the widely available 6to4 code)
3. Provisioned PC-compatible platforms with a 6to4 gateway software

4. Modified it to support 6rd
5. Tested IPv6 operation with several operating systems and applications
6. Finished operational deployment, by providing downloadable software to their Freeboxes
7. Announced IPv6 Internet connectivity, at no extra charge, for all their customers wishing to activate it

IPv6 availability was limited in December 2007 to only one IPv6 link per customer site (with /64 site-prefix assignments). A few months later, after Free had detailed its achievement and plans to its RIR, and then obtained from it a /26 prefix, up to 16 IPv6 links per customer became possible (with /60 site-prefix assignments). At the IPv6 World Congress in 2011, Free announced that they would now enable IPv6 for all new users and after that for all users. This was a very significant announcement, as it was one of the first providers to provide IPv6 not just to users asking for IPv6, but to all users without asking.

For a long time, France has been leading in the percentage of IPv6-enabled Internet users, thanks to this. In 2013, Switzerland jumped up to number one, being the first country with a double digit user penetration rate (more than 10% of IPv6-enabled users). This was mainly possible thanks to Swisscom, who has also deployed 6rd. But let's continue the Free story first.

So why did Free not simply use 6to4? For reasons mentioned before in the 6to4 section there are certain limitations in 6to4 that prevented ISPs from using it. The first reason is the communication issues. Packets coming from a 6to4 network can easily reach nodes in other 6to4 networks or in the IPv6 Internet. But it is not always possible that packets coming from an IPv6 network can reach a 6to4 site. The reason for this is that somewhere on their path these packets must traverse a 6to4 relay router. And there is no guarantee or control that routes toward such a relay router exist from anywhere and that all such relays forward packets toward the complete IPv4 Internet. Also, even if the ISP operates one or several 6to4 relays and has them advertise the 6to4 prefix of `2002::/16` toward the IPv6 Internet, it may receive packets destined to an unknown number of other 6to4 ISPs. Furthermore, ISPs prefer to assign their customers addresses out of their public ISP range and not out of the 6to4 prefix. So Free has slightly modified 6to4 and made the following changes in order to improve the situation:

1. They replaced the standard 6to4 prefix with their IPv6 prefix that belongs to the ISP-assigned address space.
2. They chose to replace the 6to4 anycast address by another anycast address chosen by the ISP.
3. The ISP operates one or several 6rd gateways (which are upgraded 6to4 routers) at its border between its IPv4 infrastructure and the IPv6 Internet.

4. CPEs (Customer Premises Equipment) support IPv6 on the customer-site side and support 6rd (which is upgraded 6to4 functionality) on the provider side.

This creative and efficient deployment has been described in RFC 5569, if you are interested in more details. But as you can guess, this is not the end of the story.

The IETF took notice of this rapid deployment and today 6rd is an official protocol specification defined in RFC 5969. It has been adopted fast, and there are many implementations out there, and more and more providers are using it in production (such as Swisscom in Switzerland) or are considering it. To make it clear, this is an automatic tunneling mechanism that can be used by providers if their backbone does not support IPv6 natively yet. If the ISP doesn't want to wait until he can upgrade his backbone he can choose to use 6rd in order to tunnel customer IPv6 traffic over his IPv4-based backbone. But it should be a temporary solution. The final goal is to be native end-to-end.

Let us have a closer look at how 6rd is designed to work. And remember, many of the processes are similar to 6to4, so we describe the changes here.

Let's first define the terminology used in the RFC and in this section:

6rd Prefix
: An IPv6 prefix selected by the service provider. There is exactly one 6rd prefix for a given 6rd domain. A service provider can deploy 6rd with one or multiple 6rd domains.

6rd Customer Edge (CE)
: The customer edge router in a 6rd deployment. Sometimes also referred to as customer premises equipment (CPE). Usually has one WAN-side interface, one or more LAN-side interfaces, and a 6rd virtual interface.

6rd Delegated Prefix
: The IPv6 prefix calculated by the CE to be used within the customer network. It is built by combining the 6rd prefix with the IPv4 address of the CE received by IPv4 configuration methods. This is equivalent to the DHCPv6 delegated prefix described in RFC 3633 and in Chapter 5.

6rd Domain
: A set of 6rd CEs and BRs connected to the same virtual 6rd link. Each 6rd domain requires a separate 6rd prefix.

CE LAN side
: This interface serves the customer-facing side of the CE and is fully IPv6 enabled.

CE WAN side
: This interface serves the WAN side of the CE and is IPv4 only.

6rd Border Relay (BR)
: A 6rd-enabled router managed by the SP, sitting at the border between the SP's IPv4 network and the native IPv6 Internet (or another IPv6 network). It has at least an IPv4-enabled interface toward the SP network, a 6rd virtual interface acting as an endpoint for the 6rd tunnel, and an IPv6 interface connecting to the native IPv6 network.

6rd Virtual Interface
: Internal tunnel interface that encapsulates and decapsulates IPv6 packets. A typical CE or BR requires only one 6rd virtual interface, but no more than one per 6rd domain.

One of the most obvious changes is the fact that 6rd does not use the 6to4 prefix (`2002::/16`). It uses the service provider's IPv6 prefix. This way the operational domain is limited to the SP's network and he has direct control over it. The RFC states that from the customer perspective and the IPv6 Internet, 6rd can be seen as equivalent to native IPv6. Addresses assigned from the 6rd-delegated prefix are treated like native IPv6 in the default address selection rules (RFC 6724).

Figure 7-9 shows the layout of a 6rd network.

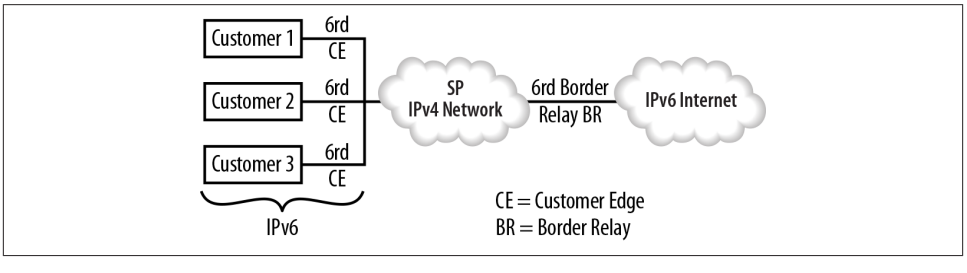

Figure 7-9. A 6rd network

A 6rd domain consists of 6rd customer edge routers (CE) and one or more 6rd border relays (BR). 6rd packets are encapsulated in an IPv4 header and therefore follow IPv4 routing within the SP's IPv4 network. The BR is traversed if the packet is destined to or coming from networks outside of the SP's network. 6rd works stateless, and thus, similar to 6to4, the BR can be reached using anycast for failover. In this case the BR's IPv4 address can be an anycast address shared within the 6rd domain. The BR will use this address as a source address in packets relayed to CEs. Since 6rd uses provider address space, no specific routes need to be advertised externally for 6rd to operate.

Let's have a look at how prefix delegation works.

The 6rd prefix is created by combining the IPv6 prefix defined by the SP with all or a part of the IPv4 address of the CE. From these two elements the 6rd delegated prefix is

automatically configured by the CE and is used in the same manner as a prefix obtained via DHCPv6 prefix delegation.

Refer to Chapter 5 for details on DHCPv6 and prefix delegation.

In 6to4, we have a well-known /16 prefix (`2002::/16`) followed by a 32-bit IPv4 address at a fixed location within the prefix. The 6to4 prefix is a /48 prefix. In 6rd, the IPv6 prefix and also the position and number of bits of the IPv4 address can vary. With 6rd the SP can adjust the size of the 6rd prefix. He can choose how many bits are used by the 6rd mechanism and how many bits are left to be delegated to the customer sites. To allow for Stateless Address Autoconfiguration, the prefix assigned to the customer site should be at least a /64. It would be preferable to assign an even shorter prefix, such as a /60 or even a /56, in order to allow the customer to further subnet his IPv6 network.

Figure 7-10 shows the format of the 6rd address.

n bits	o bits	m bits	128 - n - o - m
SP Prefix	IPv4 Address	Subnet ID	Interface ID

Figure 7-10. The format of the 6rd address

The length of the delegated 6rd prefix is the sum of the number of bits of the 6rd prefix (n) plus the number of bits of the IPv4 address (o). If the SP uses his /32 provider prefix and the full IPv4 address of the CE (32 bits), the customer will get a /64 prefix. If the SP wants to delegate a shorter prefix such as a /60 or /56, he can vary the number of bits in the prefix. Using less than the full 32 bits of an IPv4 address is possible, if a block of aggregatable IPv4 addresses is available for a 6rd domain (depending on the ISP's IPv4 address plan). So as an example, if the IPv4 addresses can be aggregated as 10.1.0.0/8, the prefix for 6rd will be a /56 (32 + 24 = 56). In this case the customer network has 8 bits for subnetting available (64 – 56 = 8).

Since 6rd delegated prefixes are selected algorithmically from an IPv4 address, changing the IPv4 address also changes the IPv6 prefix, which could be disruptive for the customer network. It is recommended to have either static IPv4 addresses on the CE or assign IPv4 addresses with long lifetimes.

While the protocol mechanism in 6rd allows for use within smaller allocations, it is much simpler operationally to support mapping of the full 32 bits of the IPv4 address into the 6rd prefix. This is what operators often prefer to do. The worry is that operators will use a /32 and give a customer only a /64, which is very problematic for any home network that has more than a single subnet. We expect that the home network in the future needs multiple subnets, for example, for guest plus home Wi-Fi support and to manage smart building components and multimedia devices.

This is the reason the developers are putting so much effort into changing the allocation policies. In the RIPE region as an ISP you can currently get a /29, which allows you to assign a /30 to be used for 6rd customers. ARIN currently has a special policy that allows for a /24 "separate allocation" with some special provisions associated. Other regions may have different rules. This is work and discussion in progress, so check with your local registry what the current rules are.

The 6rd CEs and BRs must be configured with the following elements:

IPv4 Mask Length
: The number of high-order bits that are identical across all CE IPv4 addresses within the 6rd domain. For example, if there are no identical bits, the IPv4 mask length is zero and the entire CE IPv4 address is used in the 6rd prefix (32 bits). If there are 8 identical bits, the IPv4 mask length is 8 and the 8 high-order bits from the IPv4 address are stripped off before constructing the 6rd delegated prefix.

6rd Prefix
: The 6rd prefix for the given 6rd domain.

6rd Prefix Length
: The length of the 6rd IPv6 prefix for the given 6rd domain.

6rd BR IPv4 address
: The IPv4 address of the 6rd BR for the given 6rd domain.

There is a 6rd DHCPv4 option with Option Code 212 and it looks as shown in Figure 7-11.

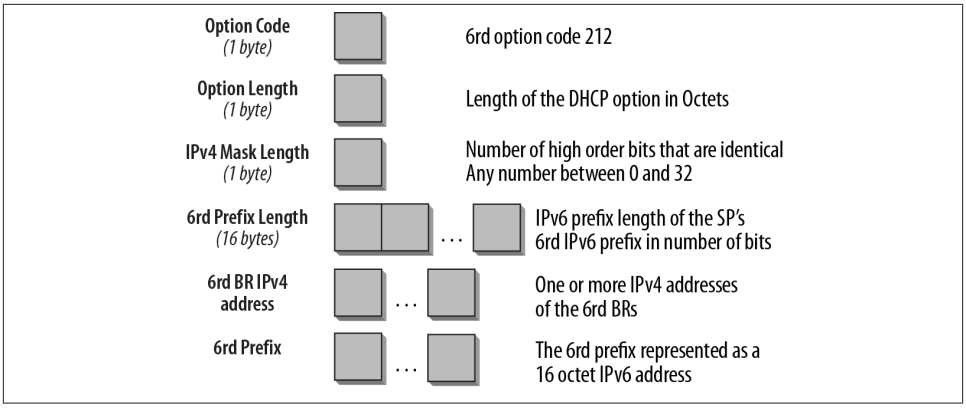

Figure 7-11. The format of the 6rd DHCPv4 option

The 6rd Option Code is set to 212. The Option Length field shows the length of the DHCP option in octets (22 octets with one BR IPv4 address). The IPv4 Mask Length field shows the number of high-order bits that are identical across all CE IPv4 addresses within a given 6rd domain. This can be a value between 0 and 32. The 6rd Prefix Length field shows the IPv6 prefix length of the SP's 6rd prefix in number of bits. The 6rd BR IPv4 Address field contains one or more IPv4 addresses of the 6rd Border Relay(s) for a given 6rd domain. And finally the 6rd Prefix field contains the 6rd prefix as a 16-octet IPv6 address. The number of bits in the prefix after the number of bits specified in the 6rd prefix length field must be set to zero by the sender and ignored by the receiver.

When 6rd is enabled, a typical CE router will create a default route to the BR, a black hole route for the 6rd delegated prefix, and routes for any LAN-side assigned and advertised prefixes.

ISATAP

The *Intra-Site Automatic Tunnel Addressing Protocol* (ISATAP) is designed to provide IPv6 connectivity for dual-stack nodes over an IPv4-based network. It treats the IPv4 network as one large link-layer network and allows those dual-stack nodes to automatically tunnel between each other. You can use this automatic tunneling mechanism regardless of whether you have global or private IPv4 addresses. ISATAP addresses embed an IPv4 address in the EUI-64 interface identifier. Note that all nodes in an ISATAP network need to support ISATAP. ISATAP is specified in RFC 5214.

Figure 7-12 shows the format of the ISATAP address.

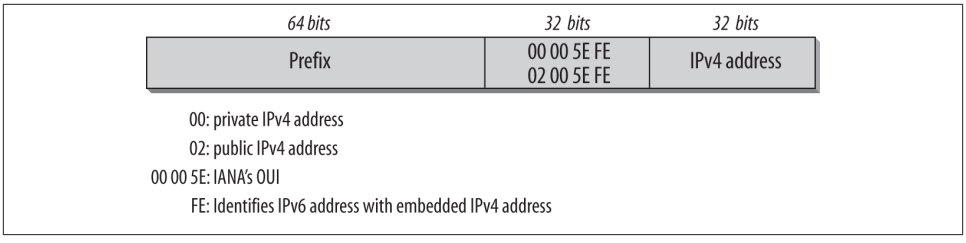

Figure 7-12. The format of the ISATAP address

The ISATAP address has a standard 64-bit prefix that can be link-local, site-local, a 6to4 prefix, or can belong to the global unicast range. The interface identifier is built using the IANA OUI `00-00-5E`, which follows the prefix. The following byte is a type field, and the value `fe` indicates that this address contains an embedded IPv4 address. The last four bytes contain the IPv4 address, which can be written in dotted decimal notation. The format of the address can thus be summarized as `64bitPrefix:5efe:IPv4address`. For instance, if you have an assigned prefix of `2001:db8:510::/64` and an IPv4 address of `62.2.84.115`, your ISATAP address is `2001:db8:510::200:5efe:3e02:5473`. Alternatively, you can write `2001:db8:510::200:5efe:62.2.84.115`. The corresponding link-local address would be `fe80::200:5efe:62.2.84.115`.

ISATAP interfaces form ISATAP interface identifiers from their IPv4 addresses and use them to create link-local ISATAP addresses. The Neighbor Discovery mechanisms specified in RFC 4861 are used (router and prefix discovery).

An ISATAP router is an IPv6 router that is also reachable over IP4 and performs the following:

- Advertises address prefixes to identify the logical ISATAP subnet on which ISATAP hosts are located.
- Forwards packets between ISATAP hosts on the logical ISATAP subnet and IPv6 hosts on other subnets (IPv4 subnets or IPv6 subnets).
- Acts as a default router for ISATAP hosts.
- Configures hosts on the logical subnet with router advertisements.

Using ISATAP, IPv6 hosts within an IPv4 intranet can communicate with each other. If they want to communicate with IPv6 hosts on the Internet, a border router must be configured; it can be an ISATAP router or a 6to4 gateway. The IPv4 addresses of the hosts within the site do not need to be public. They are embedded in the address with the standard prefix and are therefore unique and routable. Large numbers of ISATAP hosts can be assigned to one ISATAP prefix. If you deploy IPv6 on a segment in your corporate network, you configure one of the native IPv6 nodes with an ISATAP

interface, and it acts as a router between the native IPv6 segment and ISATAP hosts in the IPv4 segments.

ISATAP is an easy-to-deploy mechanism if you want to test and play quickly and have an IPv4-routed network only. It can also easily give access to the IPv6 Internet for a number of IPv6 users in an IPv4 network. But it is not recommended to use in production networks.

In an unmanaged IPv6 network, you should make sure to disable ISATAP on all hosts and block IPv6 traffic at the border for security reasons.

Teredo

6to4 makes IPv6 available over an IPv4 infrastructure using public IPv4 addresses. ISATAP enables deployment of IPv6 hosts within an IPv4 site regardless of whether it uses public or private IPv4 addresses. Teredo is designed to make IPv6 available to hosts through one or more layers of NAT by tunneling packets over UDP. It is specified in RFC 4380. There are extensions defined in RFC 6081 to support more types of NAT and RFC 5991 contains security extensions for Teredo.

Many Internet users, especially many home users, can access the Internet only through NATs (Network Address Translation). NATs create issues when tunneling IPv6 over an IPv4 infrastructure mainly for two reasons: first, NAT users have a private IPv4 address, and second, many NATs are configured to perform ingress filtering and do not allow many types of payload to go through. With tunneling, the IPv6 packet is the payload of IPv4. Mechanisms such as 6to4 often fail in these environments because they require a public IPv4 address. 6to4 can be used in NAT environments if the 6to4 router runs on the same box as NAT. In all other cases, other mechanisms have to be chosen. In our future IPv6 world, we will no longer need NATs, but for the coming transition time, we will have to deal with them. Therefore, IPv6 developers are working on mechanisms to allow users sitting behind NATs to access the IPv6 world by tunneling IPv6 packets in UDP. One of these mechanisms is Teredo.

The following terms are used with Teredo:

Teredo Service
: The transmission of IPv6 packets over UDP.

Teredo Client
: A node that has access to the IPv4 Internet and needs access to the IPv6 Internet.

Teredo Server
: A node that has access to the IPv4 Internet through a public IPv4 address and is used to provide IPv6 connectivity to Teredo clients.

Teredo Relay
: An IPv6 router that can receive traffic destined to Teredo clients and forward it using the Teredo service.

Teredo IPv6 Service Prefix
: An IPv6 addressing prefix used to construct the IPv6 address of Teredo clients. The global Teredo prefix assigned by IANA is `2001:0000::/32`.

Teredo UDP port
: The UDP port number at which Teredo servers are waiting for packets. The default value of this port is 3544.

Teredo Bubble
: A minimal IPv6 packet made of an IPv6 header and a null payload (payload type is set to 59, No Next Header). Teredo clients and relays use bubbles to create a mapping in a NAT.

Teredo Service port
: The port from which the Teredo client sends Teredo packets. This port is attached to one of the client's IPv4 addresses.

Teredo Server Address
: The IPv4 address of the Teredo server.

Teredo-mapped Address and Teredo-mapped Port
: A global IPv4 address and a UDP port that results from the translation of the IPv4 address and UDP port of a client's Teredo service port by one or more NATs. The client learns these values through the Teredo protocol.

Teredo IPv6 Client Prefix
: A global IPv6 prefix composed of the Teredo IPv6 service prefix and the Teredo server address.

Teredo Node Identifier
: A 64-bit identifier that contains the UDP port and IPv4 address at which a client can be reached through the Teredo service. A flag indicates the type of NAT through which the client accesses the IPv4 Internet.

Teredo IPv6 Address
: A Teredo IPv6 address obtained by combining a Teredo IPv6 client prefix and a Teredo node identifier.

Teredo Refresh Interval
: The interval during which a Teredo IPv6 address is expected to remain valid in the absence of "refresh" traffic. The interval depends on configuration parameters of the local NAT(s) in the path to the Teredo server. By default, clients assume an interval value of 30 seconds.

Teredo Secondary Port
: A UDP port used to send or receive packets in order to determine the appropriate value of the refresh interval, but not used to carry any Teredo traffic.

Teredo IPv6 Discovery Address
: An IPv4 multicast address used to discover other Teredo clients on the IPv4 subnet. The multicast address is `224.0.0.253`.

The Teredo service transports IPv6 packets as payload of UDP.

The Teredo design aims to provide robust access to IPv6 networks. This design creates some overhead. Teredo is only to be used if no other, more direct access is possible. For instance, if it is possible to implement a 6to4 or even a 6rd gateway on a NAT, this is the preferable solution.

A Teredo address has the format shown in Figure 7-13.

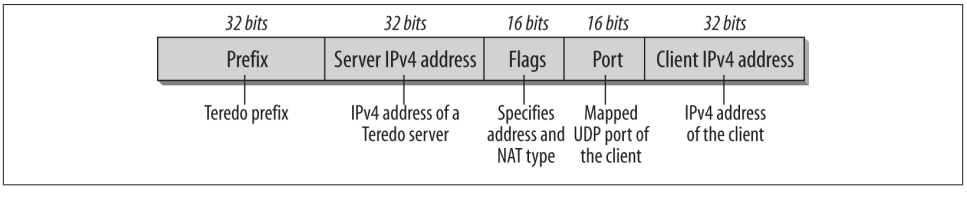

Figure 7-13. Format of the Teredo address

The Teredo service prefix has 32 bits and is `2001:0000::/32`. The Server IPv4 field has a length of 32 bits and contains the IPv4 address of the Teredo server. The Flags field has 16 bits and defines the address and the NAT type used. The 16-bit Port field contains the mapped UDP port of the Teredo Service on the client; the Client IPv4 field contains the mapped IPv4 address of the client. The bits in the Port and Client address field are all obfuscated. Each bit in the address and port number is reversed.

Figure 7-14 shows the Teredo communication.

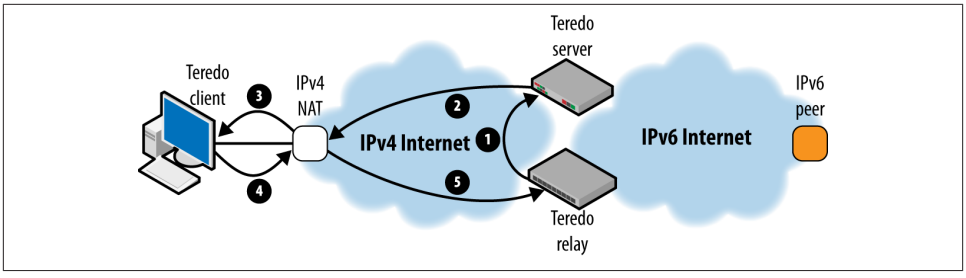

Figure 7-14. Teredo communication

A Teredo client must be preconfigured with the IPv4 address of its Teredo server. On booting, it sends a Router solicitation to the all-routers multicast address from its link-local IPv6 address. The Router solicitation is sent to the IPv4 address of the Teredo server over UDP. The Router advertisement coming back from the Teredo server contains the Teredo IPv6 Service prefix. The client builds its Teredo IPv6 address by combining the prefix with the reversed values for address and port.

When the Teredo server forwards packets from Teredo clients, it encapsulates the IPv6 packet in a UDP packet. It builds the IPv4 address and the UDP port for the destination from the destination IPv6 address. It uses its own IPv4 address as Source address and the Teredo UDP port (3544) as source port. The Teredo server's job is to forward packets from Teredo clients over UDP to the right Destination address and to forward packets for Teredo clients coming from outside to the right client internally.

The Teredo Relay is an IPv6 router announcing the Teredo Service prefix to the outside world using regular IPv6 routing mechanisms.

Teredo was one of the early transition mechanisms and like 6to4 is widely implemented. It is not recommended to use Teredo in production networks. Your Microsoft Windows clients often have Teredo activated by default (depending on OS version and patch level). Make sure, whatever operating system you are using, to verify that Teredo is turned off.

In an unmanaged IPv6 network, you should make sure to disable Teredo on all hosts and block IPv6 traffic at the border for security reasons.

Tunnel Broker

Tunnel Brokers can be seen as virtual IPv6 providers providing IPv6 Internet connectivity to users that already have an IPv4 connection to the Internet. The Tunnel Broker is specified in RFC 3053.

Figure 7-15 illustrates how the Tunnel Broker works.

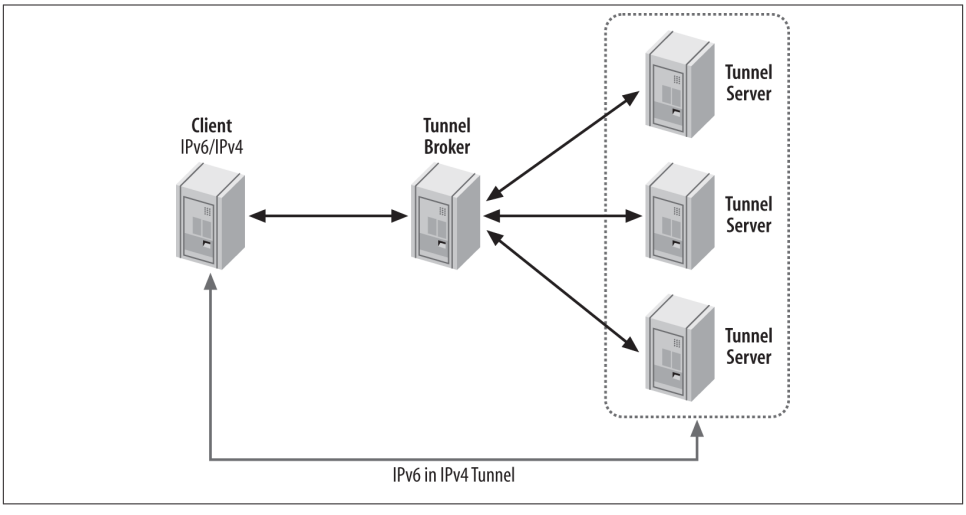

Figure 7-15. How the Tunnel Broker works

A user desiring an IPv6 connection registers with the Tunnel Broker. The Tunnel Broker manages the establishment, maintenance, and deletion of the tunnel on behalf of the user. The Tunnel Broker can share the data load across several *Tunnel Servers*. The Tunnel Broker sends the configuration information to a Tunnel Server when it wants to establish, change, or delete a tunnel. The Tunnel Broker also registers the addresses in DNS if it is configured to do so. A Tunnel Broker must be reachable with an IPv4 address. It can also have an IPv6 address, but it is not required. The communication between Tunnel Broker and Tunnel Server can run over either IPv4 or IPv6.

A Tunnel Server is a dual-stack router connected to the global Internet. When it receives configuration information from the Tunnel Broker, it establishes, changes, or deletes the server part of the tunnel.

The client is a dual-stack host or router connected to the Internet over IPv4. When it wants to register for an IPv6 connection with the Tunnel Broker, it should authenticate with standard procedures (e.g., with RADIUS). This way, unauthorized use of the tunnel service can be avoided. So the Tunnel Broker provides access control to the tunnel service. Once the client is authorized, it provides its IPv4 address, a name for the registration of its IPv6 address in DNS, and an indication of whether it is a host or a router. If the client is a router, it should send additional information about the number of IPv6 addresses that it wants to be served. The Tunnel Broker needs this information in order to assign an appropriate prefix to the client.

The Tunnel Broker fulfills the following tasks:

- Choosing a Tunnel Server as a tunnel exit point. If it has more than one option, it chooses based on preconfigured load-sharing criteria.
- Choosing a prefix for the client. The prefix can be any length. The most common values are /48 (site prefix), /64 (subnet prefix), or /128 (single host).
- Defining a lifetime for the tunnel.
- Registering the global IPv6 addresses it has assigned in DNS.
- Configuring the Tunnel Server.
- Sending the configuration information back to the client. This information includes the tunnel parameters and DNS names.

This concludes the tunnel configuration. The clients now have access to all IPv6 networks to which the Tunnel Server has access.

There are a number of ISPs that offer Tunnel Broker services. Often, users can register through the browser by filling out a form and receiving the configuration information displayed or sent by email. The client can now manually configure its tunnel entry point or use script files from the provider that automate the configuration process.

There are different lists on the Internet to find a suitable Tunnel Broker. Two well-known providers are Hurricane Electrics and SixXS. Wikipedia has a list of Tunnel Brokers.

The Tunnel Broker model is designed for smaller and isolated IPv6 networks and especially for single, isolated IPv6 hosts. It works only with public IPv4 addresses. If private addresses are used, another mechanism such as Protocol 41 Forwarding must be used.

IPv4/IPv6 coexistence by using VLANs

VLANs, which are quite common in enterprise networks, can be used to deploy IPv6 in a situation where IPv6-capable router and switch equipment is not available yet. The VLAN standard allows separate LANs to be deployed over a single bridged LAN. It uses virtual LAN tagging or membership information, which is inserted into the Ethernet frames. So to introduce IPv6 in such an environment, a parallel IPv6 routing infrastructure can be deployed, and the IPv6 links can be overlaid onto the IPv4 infrastructure by using VLAN technology. This setup doesn't require any changes to the IPv4 environment. Find a detailed description of this scenario and possible configurations in RFC 4554, "Use of VLANs for IPv4-IPv6 Coexistence in Enterprise Networks."

IPv6 in MPLS networks

MPLS (MultiProtocol Label Switching) technology was introduced originally to enhance the forwarding performance of Service Provider network cores. It is basically a tunneling mechanism where forwarding is done hop-by-hop based on locally relevant labels prepended to the IP packet. This means shorter lookups and easier hardware implementation. Since its inception, however, the capabilities of routing platforms have dramatically increased with respect to forwarding IP packets at line rate. At this point the value of MPLS does not reside as much in forwarding performance gains as it does in its ability to segment (MPLS-based VPNs) and organize traffic (Traffic Engineering) over a network core. These capabilities also made MPLS attractive to large enterprises that deploy their own network backbone.

The operation of MPLS depends on the existence of an IGP that ensures IP routing is properly set up in the backbone and a label exchange protocol. The label exchange protocol enables the P (Provider) and PE (Provider Edge) routers to define a label for each hop a packet would take from one edge of the MPLS core to another. As an IP packet enters the MPLS core, it is fitted with a locally relevant label that tells the next hop where the packet is going. The next-hop router (or MPLS switch as it might sometimes be called) swaps the label with a label relevant for the next hop and so on to the destination PE. The P router before the destination PE will strip the label and present the PE router with the IP packet that entered the MPLS core. The PE router will in turn IP route the packet further toward its final destination.

MPLS is a generic tunneling mechanism. So regardless of the control plane implementation (IGP plus label exchange), it can forward IPv6 packets just as easily as it would forward IPv4 packets. However, the options for transporting IPv6 over an IPv4 MPLS core were particularly easy to expand when the MPLS labels are complemented with labels specific to particular functionality relevant at the edge of the MPLS core. In other words, an additional label that tells the destination PE what type of IP packet this is or in what routing context the packet should be placed.

Most MPLS deployments today leverage MP-iBGP (MultiProtocol interior BGP; RFC 3107) to enhance functionality. Address families relevant to various protocols (such as IPv4 and IPv6) and various functionalities (VPNs, multicast, etc.) can be defined on the PE routers and the related traffic tagged with specific labels. MP-iBGP exchanges these labels between the PE routers, enabling them to recognize and properly process the tagged packets. These protocol or functional labels are stacked on top of the MPLS labels. Extending this functionality to IPv6 was easy. An IPv6 address family had to be identified and functionality similar to IPv4 (VPN, multicast, etc.) associated to the IPv6 address family. For example:

- Address Family Identifier (AFI) for IPv6 = 2
- Subsequent Address Family Identifier for transporting IPv6 traffic over MPLS = 4
- Subsequent Address Family Identifier for transporting VPN IPv6 traffic over MPLS = 128

For these reasons, IPv6 can easily and very effectively be transported over existing MPLS backbones. In general, IPv6 transport over an MPLS infrastructure can be done in three ways:

IPv6 over IPv6/MPLS core
: The MPLS core can be enabled directly for IPv6, meaning it runs IPv6 natively. It uses an IPv6 IGP and it exchanges labels for and over IPv6. In this case, IPv6 runs over the IPv6 MPLS core the same way as IPv4 runs over an IPv4 MPLS core. IPv4 can be tunneled over this IPv6-based infrastructure as well.

Layer 2 tunneling over MPLS
: Layer 2 tunnels can be set up across MPLS cores. These tunnels simplify the network architecture of the organizations using the MPLS backbone for connectivity. They will carry any Layer 3 protocol, including IPv6 between the PE endpoints.

IPv6 over IPv4/MPLS core
: This method is based on the distribution of IPv6 prefixes (along with the corresponding labels) between the Edge Label Switching routers using MP-iBGP4 over IPv4. The next hop is identified by an IPv4 address and the packet is forwarded over the MPLS core based on the labels for the IPv4 address of the target PE router (RFC 4798). Cisco calls their implementation of this mechanism 6PE (IPv6 Provider Edge Router). The same concept is used to enable IPv6 VPNs over an IPv4 MPLS core (RFC 4659).

It is important to note that transporting IPv6 over an IPv4 MPLS core is a very effective transition mechanism. The changes required consist only of configuration adjustments for the PE routers (assuming the functionality is supported). In this type of deployment, IPv6 benefits from all the value provided by the MPLS infrastructure, from line rate forwarding to traffic engineering.

6PE. The concept for 6PE is based on the hierarchical routing structure of MPLS shown in Figure 7-16. I do not aim to discuss general MPLS technology here; the goal is to show how MPLS can support an easy introduction of IPv6.

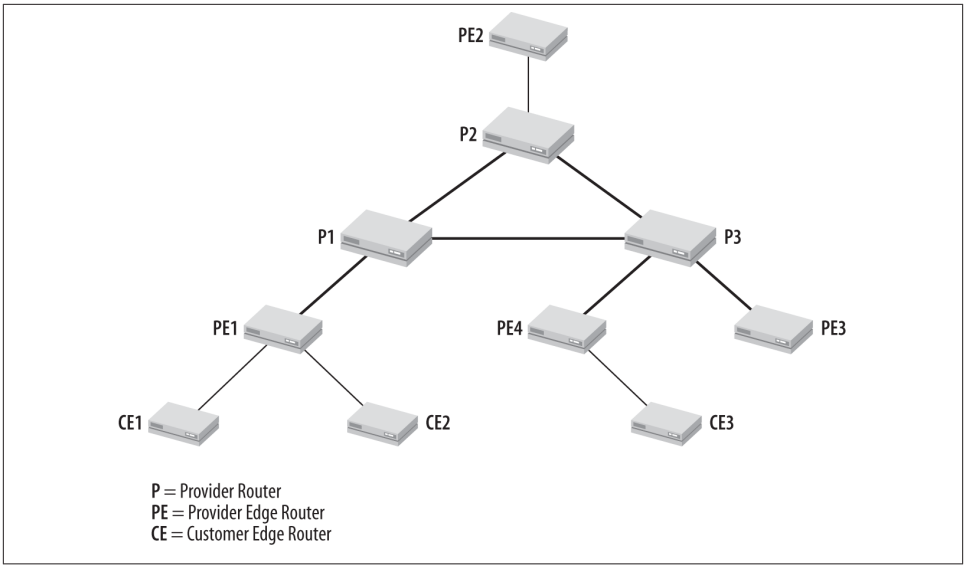

Figure 7-16. MPLS routing hierarchy

In the center of the MPLS network are the Provider routers (P). They switch the MPLS packets, which means that they do not process the Layer 3 header. At the edge of the core network are the Provider Edge routers (PE). They receive regular IP packets from the Customer Edge routers (CE), apply an MPLS label, and forward them to the Provider routers. MPLS packets are sent only between Provider Edge routers and Provider routers in Figure 7-16.

In this environment, routing works as follows:

- The PE and CE routers use the common routing protocols (RIP, OSPF, BGP, or static routing). The PE router learns the prefixes that it can reach through the CE routers through these routing protocols.
- Each PE router announces the prefixes learned from its CE routers over MP-iBGP to the other PE routers and inserts itself as next hop for these prefixes. If a global routing domain is established across the MPLS core for IPv6 (6PE) then MP-iBGP will use SAFI 4 when exchanging prefixes. Along with the prefixes, the PE advertises labels that should be used on packets sent to those prefixes.
- The MPLS core uses an IGP, typically IS-IS or OSPF, to establish reachability between the PEs. In conjunction with the label exchange protocol, Label Switched Paths (LSPs) can be set up between PEs.

Forwarding of IPv6 traffic works as follows:

- An LSP (Label Switched Path; think the equivalent of a tunnel) already exists between the ingress and egress PEs, established by the IPv4-based control plane in the MPLS core.
- MP-iBGP already informed the ingress PE (PE1) who the next-hop PE is (PE4 identified by its IPv4 address) and about the label that should be used for the IPv6 destination prefix.
- The label for the destination address prefix (CE3) is first appended by PE1 to the incoming IPv6 traffic received from CE1. The ingress PE1 also appends in front of that the label for the IPv4 address of the egress PE4 and sends the packet to the next hop P router.
- The P1 router swaps the top label (the one relevant for the egress PE4 IPv4 address) and forwards the packet to the next P router.
- The process continues hop by hop until the packet reaches the P router connected to the egress router. The P router eliminates the top label and forwards the packet to the PE router.
- The egress PE4 receives the packet with only one label; however, based on that label it will know what to do with it. If this is a 6PE deployment, it will drop the label and hand the IPv6 packet to the global IPv6 forwarding process
- The IPv6 packet is then natively forwarded to CE3 and further on to the destination.

If you are familiar with the concept of MPLS-VPNs, you can think of 6PE as one global VPN dedicated to transporting IPv6 between all PEs where it is enabled.

6VPE. Once we understand how 6PE works, and we are comfortable with MPLS-VPNs, then 6VPE will seem to be a very natural extension. For those who are not familiar with MPLS-VPN, the key concept to know is that of a Virtual Routing and Forwarding (VRF) instance. Across the MPLS backbone, if we define domains that should not share or see each other's traffic, we define Virtual Private Networks (VPNs). The PE routers must maintain that separation so the traffic for each VPN instantiated on a PE is routed and forwarded using dedicated tables that are maintained within a respective VRF (Virtual Routing and Forwarding).

In the case of 6VPE, we create MPLS-VPNs specifically for IPv6 or we dual-stack the existing VPNs. The control plane is very similar to 6PE. The MP-iBGP will exchange labels (using SAFI 128) that now not only are related to an IPv6 prefix but that are also related to the VRF to which that IPv6 prefix belongs. At the PEs, IPv6 interfaces are bound to a VRF and so the IPv6 traffic is maintained within the defined VPN. Forwarding is also similar to 6PE, the IPv6 packets are fitted with a top-level label (exchanged via the label exchange protocol) used for switching within the MPLS core and

a second label (exchanged via MP-iBGP) used to indicate to the receiving PE in which VRF the IPv6 packet should go.

Referring to Figure 7-12, unlike the 6PE scenario where all CEs are part of the same global routing domain, CE1 and CE3 belong to the red VRF while CE2 belongs to blue VRF. PE1 will maintain separate routing and forwarding instances for CE1 and CE2. Moreover, PE1 exchanges labels for the IPv6 prefixes belonging to each VRF. Forwarding is similar to 6PE:

- When CE1 sends a packet toward CE3, the packet enters PE1 through an interface associated to the red VRF.
- The packet is fitted with the label for the CE3 destination IPv6 prefix (also in the red VRF) and with the label for MPLS forwarding.
- PE1 sends the labeled packet to P1, and P1 swaps the top label and then forwards it to P2.
- P2 pops the top label and sends the packet to PE4.
- PE4 receives a labeled packet and, based on it, PE4 knows what VRF this packet belongs to and even where to send it within that VRF.
- PE4 strips the last label and lets the typical IP routing and forwarding functions within the red VRF to take it to CE3.

The use of 6PE or 6VPE depends on the service that is supported by the IPv6 traffic. If you simply want to offer IPv6 access to the Internet for all CEs, then 6PE will be a quick and easy solution. If you have multiple customers or organizations on your MPLS backbone who are separated in distinct VPNs, then 6VPE is the right solution for IPv6 enablement of these VPNs.

The fact that MPLS can be used to transport IPv6 packets over IPv4 does not mean that you should implement MPLS for this purpose. If you do not have an MPLS infrastructure in place, other tunneling mechanisms may be better suited to reach your goal. But if you already have MPLS, it is a great foundation.

If you would like to delve deeper into this, refer to the book *Deploying IPv6 Networks* by Ciprian Popoviciu, Patrick Grossetete, and Eric Levy-Abegnoli (Cisco Press). It is a practical guide to IPv6 concepts, service implementations, and interoperability with IPv4.

Locator ID Separation Protocol (LISP)

LISP (Locator ID Separation Protocol) is a new and innovative architecture, which was not purely designed to support IPv6. However, the ability to register IPv6 addresses and use IPv4 as a transport makes LISP a very useful tool for IPv6 transport over IPv4,

connecting IPv6 islands or attaching IPv6 network elements directly to the IPv6 Internet.

Many challenges with the current addressing structure of IP come from the fact that an IP address (IPv4 and IPv6) contains two different types of information. One is about the location of a device (in the subnet information, prefix) and the other about the identity of a device (host ID and interface ID, respectively). Every time you boot your notebook, you may get a different IP address; the prefix changes depending on which network you are in, and the host ID may change also (depending on the configuration). LISP enables this "level of indirection," by separating the IP addresses into two namespaces: *Endpoint Identifiers* (EID) and *Routing Locators* (RLOCs), which are assigned to a router, called *Tunnel Router*.

LISP creates an overlay network with the following benefits:

- Multihoming with ingress traffic engineering
- Address family independence (IPv4 and IPv6; MAC address registration on roadmap)
- High-scale virtualization and multitenancy support
- Datacenter mobility including session persistence with mobility events

LISP is an IETF standard defined in the RFCs 6830 to 6836 and RFC 7052. There is still ongoing work on LISP, which is documented in various drafts.

Figure 7-17 shows the basic architectural elements of LISP.

The EID (Endpoint Identifier) space is the hosts/servers identifier's IP address space. It could be IPv4 or IPv6. There are no special changes for the hosts/servers or the branch, campus, or datacenter network. The EID is the identity of the host.

The RLOC (Routing Locator) space is the transport's IP address space (identifying the location). It could be IPv4 or IPv6. There are no special changes for the transport network, which could be the Internet, a Service Provider, or a private WAN.

As LISP is a pure network-based solution, the only box that is aware of LISP is the Tunnel Router (xTR), which comes in two flavors: ingress Tunnel Router (iTR) and egress Tunnel Router (eTR). The iTR gets an IP packet from the branch, campus, or datacenter and encapsulates the original packet into an IPv4/IPv6 UDP packet with either two destination ports (UDP 4341 for data and UDP 4342 for control). The source and destination of this encapsulated packet is defined by the RLOC (transport IP address of the xTRs) of the iTR and the eTR. The eTR will receive an encapsulated packet and will do the decapsulation. The basic configs for the xTR is minimal and LISP is very lightweight to the control plane (LISP is data driven and follows a pull model).

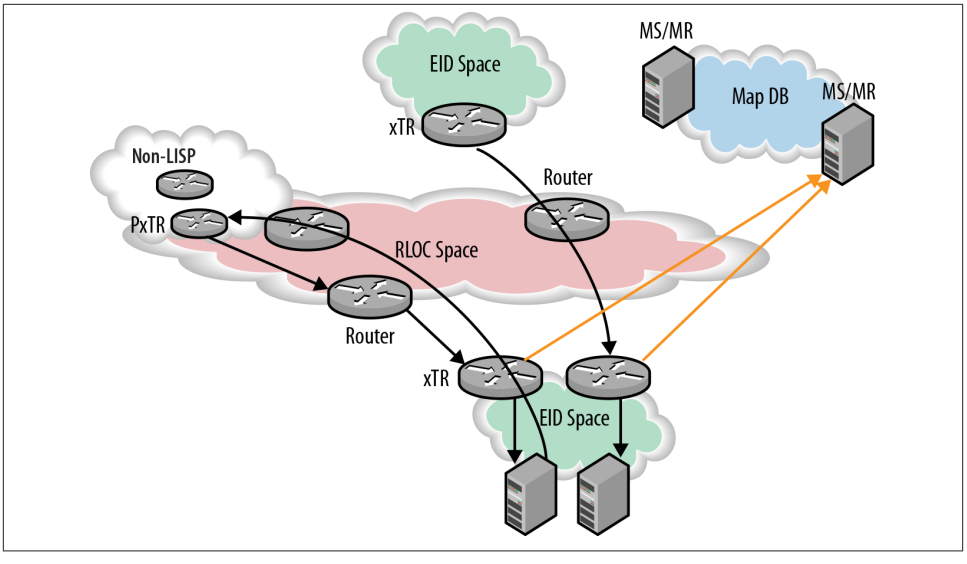

Figure 7-17. LISP architectural network elements

The *Mapping Database* consists of two functions: the *Map Resolver* (MR) and the *Map Server* (MS). The eTR will register the EIDs (either IPv4 or IPv6) with its RLOC addresses (either IPv4 or IPv6) to the Map Server (map-register message). The iTR will ask the MR, in case of any EID traffic, which destination RLOC to use for the destination EID (map-request message). The MR forwards the map-request to the MS and the MS checks its database and forwards the map-request to the eTR, which will then do a more advanced answer (map-reply message) to the iTR. The MR/MS can be configured on an xTR or can be a standalone router or server (like a BGP route-reflector). The mapping database is very scalable and can be compared with DNS for scalability and performance.

To make LISP incrementally deployable there is a special xTR called *Proxy Tunnel Router* (PxTR). The Proxy ingress Tunnel Router (PiTR) announces the EID space into the Internet or non-LISP WAN and campus, and the Proxy egress Tunnel Router (PeTR) is used to send encapsulated traffic to the non-LISP world. The trigger to use the PeTR is a negative map-reply message from the MR.

Many advantages come with LISP:

- Optimizes routing from push to pull
- Works as an overlay technology
- Uses stateless UDP encapsulation for optimized transport
- Calculates the source UDP LISP port from the original IP packet and helps with ECMP (Equal Cost Multiple Path) load balancing in the RLOC space

- Makes deployment very easy as it only touches the network equipment at the edge

These advantages come in addition to the use cases for multihoming with real load balancing in the ingress. It supports virtualization for VRF transport and unique mobility features.

But the most important part in this context is the capability to use IPv4 as a transport for IPv6. LISP can be used to connect IPv6 islands over an IPv4 network or deliver a scalable and lightweight solution to connect to the IPv6 Internet.

Figure 7-18 shows the concept for transport of IPv6 over IPv4.

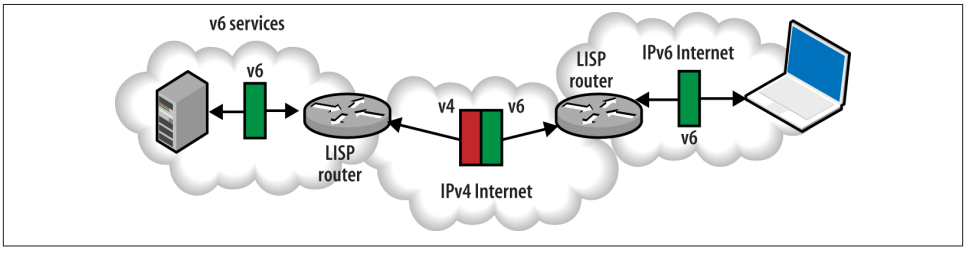

Figure 7-18. LISP IPv6 over IPv4

The LISP router on the left will do a map-register with the LISP mapping-database (not shown here). It will register the IPv6 address (IPv6 EID) of the IPv6 service with the IPv4 RLOC (Internet IP address) of the LISP router.

The Proxy Tunnel Router (PxTR; LISP router on the right) is connected to the IPv4 and IPv6 Internet. The PiTR will announce the IPv6 EID to the IPv6 Internet to attract the traffic for the IPv6 service. When a packet for the IPv6 service arrives on the PiTR it will do a map-request for the IPv6 EID. The LISP router on the left (eTR) will get the map-request from the MR/MS and will answer with its IPv4 RLOC. The PiTR will now create a map-cache with the IPv6 EID and the next-hop IPv4 RLOC. Now the packet from the IPv6 Internet will be encapsulated in IPv4 and forwarded over the IPv4 Internet. The eTR will then decapsulate the packet and deliver this natively as an IPv6 packet to the IPv6 service.

The return traffic from the IPv6 EID to the IPv6 Internet works similarly. The iTR gets the IPv6 packet and depending on the configuration, it will issue a map-request or it will send the packet directly to the PeTR (static map-cache). In case of a map-request, a negative map-reply message will be received from the MR including the PeTR IPv4 RLOC. The iTR can now encapsulate the IPv6 packet into IPv4 with its source IPv4 RLOC and the IPv4 RLOC of the PeTR. The PeTR gets the encapsulated LISP packet, strips off the IPv4 transport header and forwards the IPv6 packet natively.

These examples show the ease of LISP IPv6 over IPv4 transport. You can use the LISP control-plane or you can configure it statically as a direct map cache. So if you want to enable IPv6 over IPv4 with LISP, all it takes is: switch on LISP for xTR, configure your control-plane info (MR and MS), and define the IPv6 prefix that is to be transported over IPv4. Additionally, you can add a static map-entry to the iTR.

LISP is available on all Cisco routers, on Catalyst 6500/6800, and Nexus7000. The German CPE vendor AVM implemented LISP for IPv6 transport in their Fritz!Box, and there are a couple of open source versions (OpenLISP or LISPmob). There are also plans from other vendors to adopt LISP soon.

The solution from Figure 7-18 can be combined with other LISP use cases to create a real network architecture, such as adding virtualization for transporting different VRFs or increasing availability and load balancing with multihoming.

For further information on LISP, check out *http://www.lisp4.net* or *http://www.lisp6.net* (this link only works if your IPv6 connection works).

The LISP workgroup can be found at *http://www.tools.ietf.org/wg/lisp*.

Generic Routing Encapsulation

Another Tunneling mechanism that can be used is Generic Routing Encapsulation (GRE). GRE is specified in RFC 2784 and is designed to encapsulate any protocol in another protocol. The protocol being encapsulated—in our case IPv6—is called the Passenger Protocol. The protocol that is used to encapsulate—in our case IPv4—is called the Carrier Protocol.

The configuration of a GRE tunnel is manual. On both tunnel endpoints (the GRE routers), the IPv4 address of the tunnel peer is preconfigured. So for each route where IPv6 has to be tunneled, a tunnel must be configured separately. In a more complex network, this can lead to a high initial configuration effort. A GRE tunnel cannot traverse NATs. It is useful when multiple protocols have to be tunneled through the same tunnel.

Softwire Hub and Spoke Deployment Framework

This framework uses Layer 2 Tunneling Protocol version 2 (L2TPv2) and is basically L2TP tunneling. This works well in ISPs which have PPP deployed. In the "Hub and Spoke" solution space, a softwire is established to provide the home network with IPv4 connectivity across an IPv6-only access network, or IPv6 connectivity across an IPv4-only access network. A softwire is a tunnel that is created on the basis of a control protocol setup between softwire endpoints with a shared point-to-point or multipoint-to-point state.

Just like any tunnel mechanism, it can be used until the rest of the infrastructure can be updated to support native IPv6. It has been deployed by many ISPs, as it allows them to use a lot of their existing infrastructure. It is defined in RFC 5571.

Proto 41 forwarding

Some NAT implementations allow the configuration of IPv6 tunnels from inside of the private LAN to routers or tunnel servers in the Internet. This is a simple and helpful way to provide IPv6 nodes and IPv6 networks behind a NAT with access to the IPv6 Internet. This should only be used if no other mechanisms such as 6to4 or native IPv6 are possible.

A tunnel client (host or router) with a private IPv4 address and a connection to the Internet through an IPv4-only NAT box can use a Tunnel Broker or an IPv6 router to create an IPv6 tunnel. Many NAT boxes can be configured to forward packets based on the protocol value of 41 (for IPv6) in the IPv4 header. This provides an opportunity to rapidly deploy a huge number of IPv6 nodes and networks.

Most of the existing solutions for the transition to IPv6 rely on tunnels, assuming that the client endpoint is an IPv6-capable router. However, nowadays the installed base of IPv4-only NAT boxes/routers is still quite large, while most of the client operating systems already support IPv6.

SSH (Secure Shell) Tunnels

You won't find SSH Tunnels as an official IPv6 transition mechanism, but they can be very practical and offer useful solutions in different situations. This section describes what they are and how you can use them in an IPv6 environment.

The aim of two projects, the commercial OpenSSH and the closed source SSH, was to eliminate the use of unencrypted protocols such as Telnet, rlogin, and rsh. This section is by no means a complete overview of SSH, but it shows how SSH tunnels can be used as a simple transition mechanism for IPv4 to IPv6 and vice versa.

For a more precise look at SSH, we recommend *SSH—The Secure Shell, The Definitive Guide*, Second Edition, written by Daniel J. Barrett et al. (O'Reilly).

Both projects allow for a practice called *port forwarding*, which essentially allows TCP ports to be forwarded between machines. It is also loosely referred to as the Poor Man's VPN. In the scenario shown in Figure 7-19, we have an IPv4-only client connecting to a dual-stacked host running either version of SSH (both versions of SSH are IPv6 compliant).

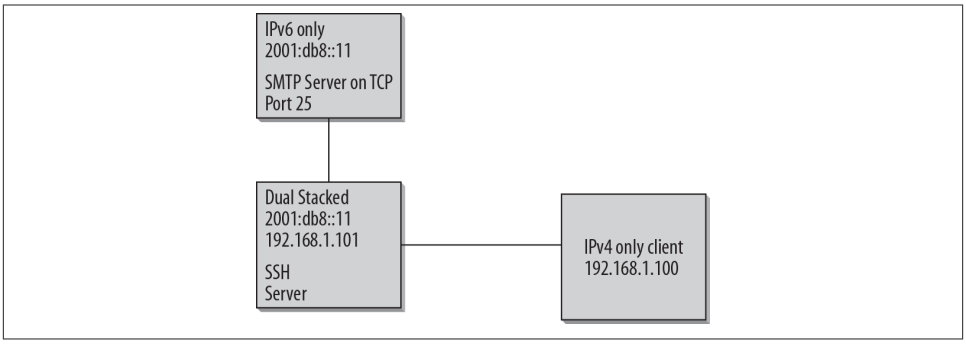

Figure 7-19. IPv4 client connects to IPv6-only server through a dual-stacked SSH host

The IPv4 client wishes to send mail via the IPv6-only server. With SSH's flexibility, this can be accomplished one of two ways, described next. The examples that follow show how to do this using the command-line SSH client available from both vendors, but this can also be easily accomplished with GUI tools (please check your vendor's documentation for your GUI tool).

Port forwarding TCP port 25 back to the client

Typing the following command on the SSH server allows a single IPv4 client to access the IPv6 mail server: `ssh -L 25:[2001:db8::11]:25` *user@192.168.1*`.101`. After typing in a password, this command will forward TCP port 25 from the IPv6 server back to the IPv4-only client. On the client side, a simple `telnet 127.0.0.1 25` shows that the data is actually initiated on the local host, then forwarded to the IPv6-only SMTP server via the SSH server. It may sound a bit complex, but it works very well.

Port forwarding TCP port 25 to the SSH server and allowing clients to connect to port 25 on the SSH server

This method, while a bit more complex on the initial setup, can ease administration issues; the connection only has to be initiated once, and then each client can connect to the SSH server. The following commands are typed on the SSH server: `ssh -g -L 25:[2001:db8::11]:25` *user@127.0.0*`.1`. After typing this on the server and logging in, you should be able to type `telnet 127.0.0.1 25` and get the SMTP prompt from the IPv6-only SMTP server. The difference here is the `-g` in the SSH command, which allows Gateway mode. In Gateway mode, clients other than localhost can connect to that port. Typing the command `telnet 192.168.1.101 25` at the IPv4-only workstation allows that client to connect to the IPv4 side of the dual-stacked SSH server, which relays the data to the IPv6-only SMTP server.

The flexibility of SSH tunnels allows for many other combinations, including the client being able to forward the port and allowing use of pregenerated keys for ease of

administration (no logging in required). Disadvantages of using SSH as a transition mechanism include being able to forward only TCP connections and a possibility of high processing overhead in forwarding many ports using the same machine.

IPv4 Residual Deployment via IPv6 (4rd)

4rd is a stateless automatic tunneling mechanism to transparently tunnel IPv4 packets over IPv6 networks. It is the reverse mechanism of 6rd. While IPv6 headers are too long to be mapped into IPv4 headers, so that 6rd requires encapsulation of full IPv6 packets in IPv4 packets, IPv4 headers can be reversibly translated into IPv6 headers in such a way that, during IPv6 domain traversal, UDP packets with checksums and TCP packets are valid IPv6 packets.

4rd is in draft status at the time of writing. The draft is called "IPv4 Residual Deployment via IPv6—a Stateless Solution (4rd)" (*draft-ietf-softwire-4rd-08*).

Network Address and Protocol Translation

NAT (Network Address Translation) and Protocol Translation are the most debated areas in IPv6 deployment scenarios. This section discusses the different forms of NAT that are available for the IPv4 address depletion problem and the integration of IPv6.

Network Address and Protocol Translation techniques offer transition mechanisms in addition to dual-stack and tunneling techniques. The goal is to provide transparent routing for nodes in IPv6 networks to communicate with nodes in IPv4 networks and vice versa.

While with tunneling, the original IPv6 packet remains untouched, simply encapsulated in an IPv4 header and then decapsulated again at the tunnel endpoint, a translated packet is modified at the translator, according to the rules defined in what is called Stateless IP/ICMP Translation Algorithm (SIIT; RFC 6145).

In the IPv4 world, Network Address and Port Translation (NAPT, usually simply called NAT) was defined many years ago to map between private addresses inside a network with a public address toward the outside world, in order to address the ongoing depletion of the public IPv4 address pool. This is a stateful technique, because the gateway needs to maintain state in order to route return packets correctly. Today, in our dual-stack world, this type of NAT is often called *NAT44*. In order to provide always-on connectivity to many devices, NAT not just maps devices and addresses, but also uses ports for each address to map multiple private addresses to one public address. There are 65,636 port numbers available for UDP and TCP each, many of which are unused. So NAT actually maps the internal private address and the port number to the outside public address and port number. This way it can map a large number of sessions for

each public address. This NAT usually sits at the customer edge and runs on the CPE (Customer Premises Equipment).

The fact that we have waited too long with the deployment of IPv6 and are now running out of IPv4 address space will force us to use this type of transition technology to deal with the exponential growth of the Internet. The problem is that if new Internet users only get IPv6 addresses, they will not be able to access the still predominantly IPv4-accessible web content. So the IETF working groups decided to define standard translation methods to prevent the industry from developing an ungovernable variety of nonstandard methods.

Before we go into the different types of NAT available today, let us have a look at the specification for IP and ICMP translation.

Stateless IP/ICMP Translation

For the case in which IPv4-only hosts want to communicate with IPv6-only hosts or vice versa, RFC 6145 defines how a protocol translator has to translate the IP and ICMP headers for both parties to understand each other. For example, you might have a new network segment and want to roll out native IPv6 hosts. With the implementation of a protocol translator, it is possible to set up the new IPv6-only network internally and have those IPv6-only clients access the standard IPv4 Internet or any other IPv4-only node.

For this discussion, we need to introduce a few terms. They are defined in RFC 6052, "IPv6 Addressing of IPv4/IPv6 Translators," which is a part of a series of documents about IPv4/IPv6 translation. It specifies how an individual IPv6 address is translated to an IPv4 address and vice versa, in cases where an algorithmic mapping is used.

Address translator
: A device that derives an IPv4 address from an IPv6 address or vice versa. This applies to devices that do IPv4/IPv6 translation, and also to other devices that manipulate addresses such as name-resolution proxies (e.g., DNS64 described later) and possibly other types of Application Layer Gateways (ALGs).

IPv4-converted IPv6 address
: An IPv6 address used to represent an IPv4 node in an IPv6 network. It is a variant of IPv4-embedded IPv6 addresses.

IPv4-embedded IPv6 address
: An IPv6 address in which 32 bits represent an IPv4 address.

IPv4/IPv6 translator
: A device that translates IPv4 packets to IPv6 packets and vice versa. The translation can be *stateless* (no per-flow state required) or *stateful* (per-flow state is created when first packet in a flow is received).

IPv4-translatable IPv6 address
: An IPv6 address assigned to an IPv6 node for use with stateless translation. It is a variant of IPv4-embedded IPv6 addresses.

Network-specific prefix
: An IPv6 prefix assigned by an organization for use in algorithmic mapping.

Well-known prefix
: The well-known prefix to be used for algorithmic mappings is `64:ff9b::/96`.

All IPv4-embedded addresses follow the same format described in a table in RFC 6052. A table, because they are composed of a variable-length prefix, the embedded IPv4 address, and a variable-length suffix (depending on the total length of the prefix, which can vary from 32 bits to 96 bits). The table in the RFC outlines all possible options.

The prefix can either be a network-specific prefix or the well-known translation prefix. For stateless translation, a network-specific prefix should be used. For stateful translation, an organization can choose between a network-specific prefix or the well-known prefix. The well-known prefix should be used in most cases, except if it deems appropriate for management and operational reasons, or in the scenario where the IPv6 Internet connects to an IPv4 network.

The companion RFC 6144, "Framework for IPv4/IPv6 Translation," describes eight different scenarios and outlines the requirements and rules for stateful or stateless translation. They are listed in the section "NAT64 scenarios" on page 268.

TCP and UDP headers generally do not need to be modified by the translator. One exception is UDP headers that need a checksum for IPv6 because a UDP checksum is required for IPv6. The same is true for ICMPv4 messages that need a checksum for ICMPv6. In addition to the checksum, ICMP error messages contain the IP header of the original packet in the payload that needs to be modified by the translator; otherwise, the receiving node cannot understand it. IPv4 options and IPv6 Routing headers, Hop-by-Hop Options headers, and Destination Option headers are not translated. Also, the translation techniques cannot be used for multicast traffic, because IPv4 multicast addresses cannot be mapped into IPv6 multicast addresses and vice versa.

Just as with dual-stack nodes, applications running on nodes that use IP/ICMP translation need a mechanism to determine which protocol version to use for communication with their peers.

Translating IPv4 to IPv6

An IPv4-to-IPv6 translator receives an IPv4 datagram. Because it has been configured to know the pool of IPv4 addresses that represent the internal IPv6 nodes, the translator knows that the packet needs translation. It removes the IPv4 header and replaces it with

an IPv6 header by translating all the information from the IPv4 header into the IPv6 header.

Path MTU Discovery is optional in IPv4 but mandatory in IPv6. If an IPv4 host does Path MTU Discovery by setting the Don't Fragment Bit in the header, Path MTU Discovery works even through the translator. The sender may receive Packet Too Big messages from both IPv4 and IPv6 routers. If the Don't Fragment Bit is not set in the IPv4 packet, an IPv6 translator has to ensure that the packet can safely travel through the IPv6 network. It does this by fragmenting the IPv4 packet if necessary, using the minimum MTU for IPv6, 1,280 bytes. IPv6 guarantees that 1,280-byte packets will be delivered without a need for further fragmentation. In this case, the translator always includes a fragment header to indicate that the sender allows fragmentation. Should this packet travel through an IPv6-to-IPv4 translator, the translator knows it can fragment the packet.

For a UDP packet with a zero checksum, the translator must calculate a valid checksum for IPv6. If a translator receives the first fragment of a fragmented UDP packet with a zero checksum, it should drop the packet and generate a system message specifying the IP address and port number. Further fragments should be silently discarded.

Translating ICMPv4 to ICMPv6 and vice versa

For all ICMPv4 messages, the translator has to compute a valid checksum because it is required with ICMPv6. In addition to this, the type values have to be translated and, for error messages, the included IP header also needs to be translated. Internet Group Management Protocol (IGMP) messages are single-hop messages and should not be forwarded over routers. Therefore, they do not require translation and are silently discarded.

The same translation rules apply to the translation of ICMPv6 messages to ICMPv4 messages, only in reverse order.

Translating IPv6 to IPv4

This process is not much different from the translation discussed previously. In this case, the translator knows that it has to translate from IPv6 to IPv4 based on the IPv4-mapped Destination address. It removes the IPv6 header and replaces it with an IPv4 header. The minimum MTU for IPv4 is 576 bytes; the minimum MTU for IPv6 is 1,280 bytes. If a translator receives a packet for an IPv4 network with a smaller MTU, it creates 1,280-byte packets and fragments them after translation.

NAT to Extend IPv4 Address Space

This section is not really about IPv6 transition. It explains the ways NAT is used today in order to extend IPv4 address space (and this is what NAT was originally designed for). These mechanisms will be used by ISPs all over the world because we waited too

long and the IPv4 address space is now exhausted. These mechanisms have a major impact on how users access IPv4 websites.

Carrier Grade NAT

As providers run out of IPv4 addresses and cannot cover Internet growth with IPv4, they have to deploy IPv6. But users want to be dual-stacked as they want to be able to access the IPv4 content on the Internet. So why not NAT the IPv4 part of the Internet connection? This lets users access IPv6 content over IPv6, but still get to IPv4 content over their NATed IPv4 connection.

This can be achieved by using what we call Carrier Grade NAT (CGN, also called LSN for Large Scale NAT), or *NAT444*. It means we add another layer of NAT to the NAT44 by adding a NAT44 inside the ISP's network. Traditional NAT44 is between the customer network and the ISP network. CGN is between the customer networks and the ISP network and allows the ISP to assign a private IPv4 address to the customers, not a public one. Or in other words, the traditional customer-side NAT now translates from private IPv4 inside to private IPv4 outside. Figure 7-20 shows this in a diagram.

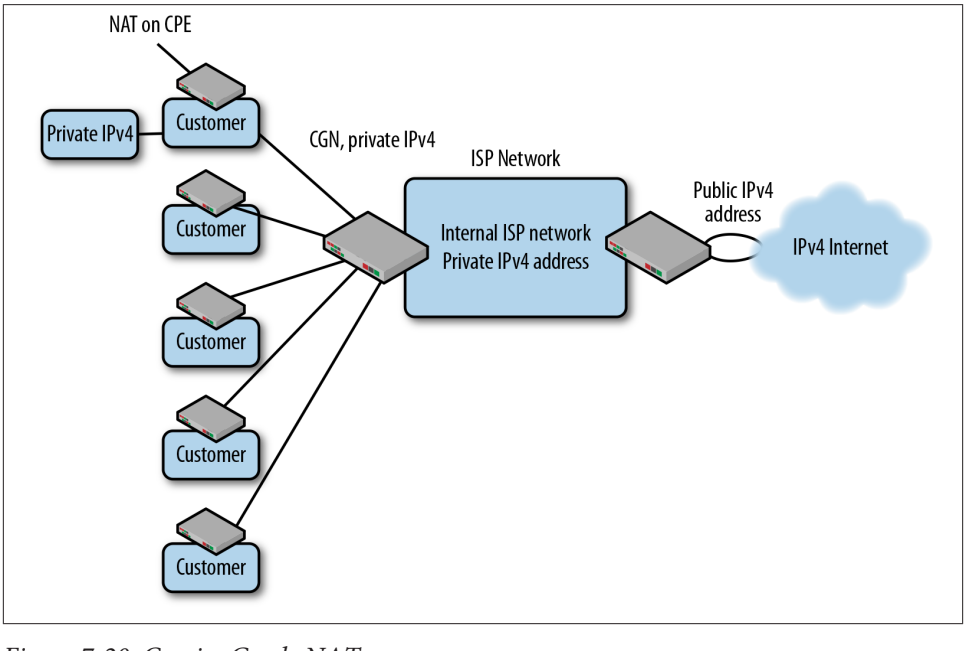

Figure 7-20. Carrier Grade NAT

On the bottom part we see the traditional NAT connecting a customer's privately addressed networks through NAT with the provider network. In this NAT444 scenario,

the translation is from private IPv4 to private IPv4. This allows the ISP to connect many customers through a single public IPv4 address on the outside.

Let's follow a packet. It originates inside the customer site. Its address is converted from private inside to a private address from within the CGN. When leaving the ISP network, it gets the public address assigned to the outside interface of the CGN. The packet goes through address translation and port mapping twice. This mechanism is called NAT444 because it only translates IPv4 to IPv4, with the goal to expand the address space. The advantage is that it can mostly be achieved with current equipment and implementations.

Experience will have to show how this scales with large numbers of users. The processing it takes for all the translations and mappings for a large number of users will have its limits that are yet to be determined. There may also be issues with overlapping private space, if an organization internally uses the same range as the provider within the CGN. Another issue might occur if customers connected to the same CGN want to send traffic to each other. Their packets may have to be routed to the outside and come back with a public IPv4 source address; otherwise, they may be filtered by traditional ACLs based on their private source address.

Where with traditional NAT, users in one site shared one public IPv4 address, with CGN multiple customers share one public IPv4 address. This can create some critical issues. For instance, if somebody successfully attacks the public IPv4 address, not just one customer, but all customers using that CGN IPv4 address may be affected. If one of the customers is a bad guy and blacklisted, all of the customers sharing the same address will be affected.

Another issue is the fact that all these customers share a fixed pool of ports and so only a limited number of ports per customer will be available to a probably increasing number of customers. When users run applications that use a large number of multiple simultaneous sessions, such as Google Maps or iTunes to name two examples, the CGN gateway may run out of port numbers and sessions. And the fact is that today's applications have an increasing appetite for parallel sessions. This leads to the fact that applications or services may not run well or even fail. For a user sitting behind a CGN this is usually not traceable. The user will assume that the website is not running or has issues.

From the perspective of a content provider, you want to offer your public services dual-stack in order to bypass CGNs and make sure your customers have a good experience when they visit your website. If your content is dual-stack, Internet users with IPv4 CGN access can come to your site using native IPv6 (if their provider offers IPv6 Internet access).

There is an interesting RFC, RFC 7021, "Assessing the Impact of Carrier-Grade NAT on Network Applications," which summarizes CGN tests performed by CableLabs, Time Warner Cable, and Rogers Communications. They independently tested the impacts of NAT444 on many popular Internet services using a variety of test scenarios, network topologies, and vendor equipment. The RFC identifies areas where adding a second layer of NAT disrupts the communication channel for common Internet applications.

NAT464

Another way to solve this problem can be to deploy IPv6-only between the customer edge and the provider network. This requires translation from IPv4 to IPv6 at the customer edge and translation again from IPv6 to IPv4 at the CGN. This reduces the need for IPv4 addresses on the provider side. Translation becomes more difficult as translation across protocols (from IPv4 to IPv6) is more complex than address translation within one protocol family. Also NAT444 is widely implemented and available while implementations for NAT464 are not so widespread at the time of writing. Figure 7-21 shows the NAT464 network.

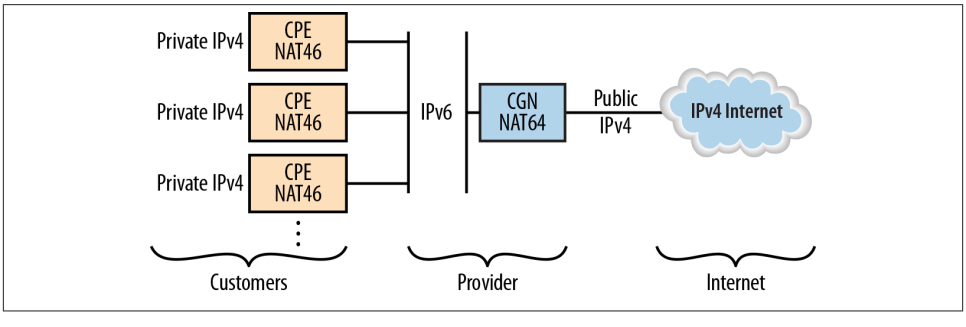

Figure 7-21. NAT464

In this case, the provider conserves IPv4 address space not by assigning private IPv4 addresses to the customer CPEs, but rather by using an IPv6-only network and translating the customer's IPv4 traffic to IPv6 with NAT46 and translating it back to IPv4 with NAT64 at the CGN. In both cases, with CGN and NAT464, multiple customers share one public IPv4 address.

The main disadvantage of this type of translation in general is, besides the fact that a NAT device is always a bottleneck, that when you have to translate IPv6 to IPv4, you lose all the advanced features of IPv6, because they cannot be translated to IPv4. So, for instance, if the packet has Extension headers, not all the information can be translated into IPv4 options. But in this scenario, where the applications are IPv4 and the tunnel is used to get from one IPv4 island to the next, there are probably no advanced IPv6 features to be lost.

DS-Lite

DS-Lite is another mechanism to allow for IPv6-only connection between the customer site and the CGN. Instead of translating from IPv4 to IPv6 and vice versa, such as in NAT464, the IPv4 packets are tunneled in IPv6 to the CGN. That is to say, one level of translation from IPv4 to IPv6 and IPv6 to IPv4, as in NAT464, is removed. DS-Lite is specified in RFC 6333. New terms in the specification are:

DS-Lite Basic Bridging BroadBand element (B4)
: B4 is a function implemented on a dual-stack capable node. This node can be either a directly connected node or a CPE (Customer Premise Equipment) that creates a tunnel to the AFTR (defined below).

DS-Lite Address Family Transition Router (AFTR)
: An AFTR is the combination of an IPv4-in-IPv6 tunnel endpoint and an IPv4-IPv4 NAT implemented on the same node. The AFTR can be provisioned with different NAT pools and serve different groups of clients with different address pools.

DS-Lite is shown in Figure 7-22.

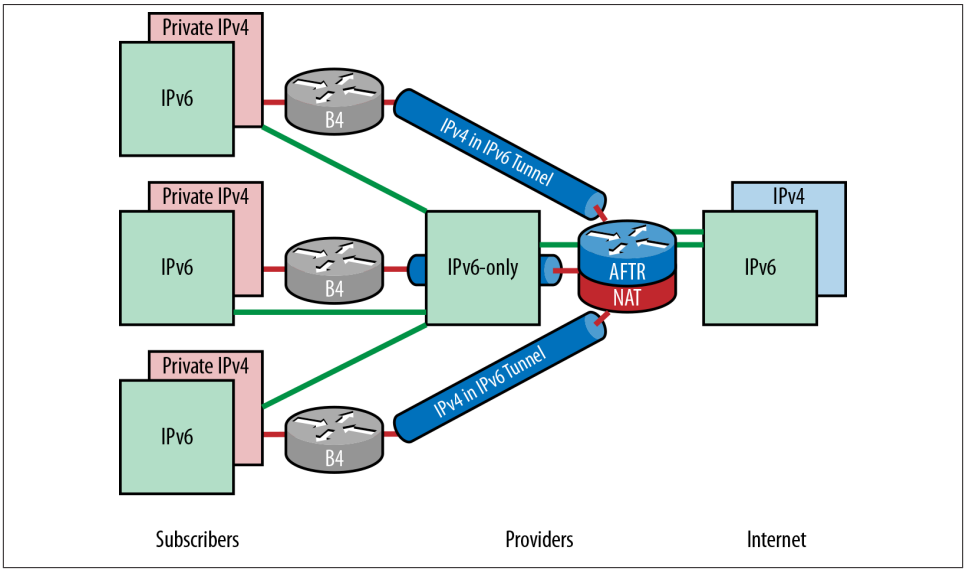

Figure 7-22. DS-Lite

The diagram shows that the private customer network is connected to the ISP private network through an IPv6 tunnel. The NAT maps the combination of IPv6 source address, IPv4 source address plus port to the outside IPv4 address plus port. Several customers are sharing one IPv4 public address. All IPv4 in IPv6 tunnels end on the AFTR. When customers communicate using IPv6, there are no tunnels, the AFTR is bypassed

and the traffic goes out natively. The trick here is to ensure that source addresses are unique. If many customers using private RFC 1918 addresses connect, their source address isn't distinguishable anymore. DS-Lite solves this problem by linking the IPv4 source address with the unique IPv6 address used for the tunnel.

Usually the CPE has DHCPv4 functionality handing out private IPv4 address space to hosts in the home network. It also advertises itself as a DNS server and should run a DNS proxy to resolve DNS queries from IPv4 hosts with the service providers DNS servers over the IPv6 network. In order to establish the tunnel to the AFTR, the B4 element is configured with the IPv6 address of the AFTR either through manual configuration or through DHCPv6. RFC 6334 defines a DHCPv6 DS-Lite option. A well-known IPv4 subnet address has been defined by the IANA to represent the B4 element. The range is `192.0.0.0/29`. `192.0.0.1` is reserved for the AFTR element and `192.0.0.2` is reserved for the B4 element.

DS-Lite removes one level of NAT compared to NAT444 or NAT464. The disadvantage is that single users cannot be identified by their IP address anymore. Currently DS-Lite only specifies IPv4 in IPv6 tunnels. Other types of encapsulation could be defined in the future. RFC 6619, "Scalable Operation of Address Translators with Per-Interface Bindings," for instance, defines a solution to use protocol translation during the migration period to IPv6 to deploy these mechanisms in a way that allows the support of a large user base without the need for a correspondingly large IPv4 address block.

RFC 6908, "Deployment Considerations for DS-Lite," refers to the scenarios for DS-Lite mentioned in the Appendix of RFC 6333 and describes problems that can arise when deploying DS-Lite and how they can be mitigated. The information and recommendations in this RFC are based on real-world experience and can be useful for operators.

There is an extension to DS-Lite underway (in draft status at the time of writing), called *Lightweight 4over6* (LW46). It moves the Network Address and Port Translation (NAPT) function from the centralized DS-Lite tunnel concentrator to the tunnel client located in the CPE. This removes the requirement for a Carrier Grade NAT function in the tunnel concentrator and reduces the amount of centralized state that must be held to a per-subscriber level.

NAT as an IPv6 Translation Mechanism

In the early days of IPv6 there was one main translation mechanism defined in RFC 2766, "NAT-PT, Network Address Translation—Protocol Translation." It was moved to historic with RFC 4966 because it was too complex. The SIIT (Stateless IP/ICMP Translation) RFC is still active and the specification is used in newer forms of translators, such as in NAT64.

The following sections describe a number of different translation techniques, some of them still in draft status. Depending on when you read this book, they may be published as an RFC.

Expect to see more types and flavors of translation mechanisms to appear. But note that the main recommendation is to not use them if possible and go native IPv6 wherever you can, especially in the enterprise network. Providers may have to use translation mechanisms, mainly due to IPv4 address shortage. As an Internet customer you want to know what kind of access your provider gives you, from an IPv4 and IPv6 perspective.

Stateless NAT64

Stateless NAT64 provides a translation mechanism that translates IPv6 headers into IPv4 headers and vice versa. It is based on RFC 6144, which defines a framework for IPv4/IPv6 translation and provides an overview and discussion of all possible scenarios. Due to the stateless character, this mechanism is very efficient. It supports end-to-end transparency and has a better scalability than stateful translation. Multiple translators can be deployed in parallel without the need to synchronize state between them.

For the stateless mechanism the translation information is carried in the address itself. To perform stateless translation, there must be a rule how an IPv6 address is translated to a corresponding IPv4 address and vice versa. A specific IPv6 address range represents the IPv4 systems in the IPv6 world. This range is manually configured on the NAT device. In the IPv4 world all the IPv6 systems have directly correlated IPv4 addresses that can be mapped to a subset of the service provider's IPv4 addresses. The IPv6 hosts are assigned IPv6 addresses through either manual configuration or DHCPv6. The IPv4-embedded IPv6 address format is described in section 2.2 of RFC 6052. The well-known prefix to use for an algorithmic mapping is `64:ff9b::/96`. Common implementations often allow you to configure your own prefix taken from your IPv6 address range.

With stateless NAT64, sessions can be initiated from both sides, from IPv4 to IPv6 and vice versa. The disadvantage is that it consumes an IPv4 address for each IPv6 device that needs translation. So it is not a solution to address IPv4 address depletion. It can be used to provide public servers with an IP address for both protocols. But to aggregate many IPv6 users to a single IPv4 address, stateful NAT64 has to be used. Another limitation of stateless NAT64 is that only IPv4 options that have direct counterparts in IPv6 are translated, and it does not translate IPv6 Extension headers except for the Fragmentation header. The best use case for stateless NAT64 is probably to provide access to an IPv6 server to IPv4 clients.

Stateful NAT64 and DNS64

This scenario is used where users only have IPv6 addresses but need to connect to IPv4 networks and to the IPv4 Internet. One or more public IPv4 addresses are assigned to the translator to be shared among the IPv6 clients. When stateful NAT64 is used with DNS64, no changes are usually required in the IPv6 client or the IPv4 server. To have support for DNS64, use BIND 9.8.0. Stateful NAT64 is specified in RFC 6146 and DNS64 in RFC 6147.

Figure 7-23 shows how this works.

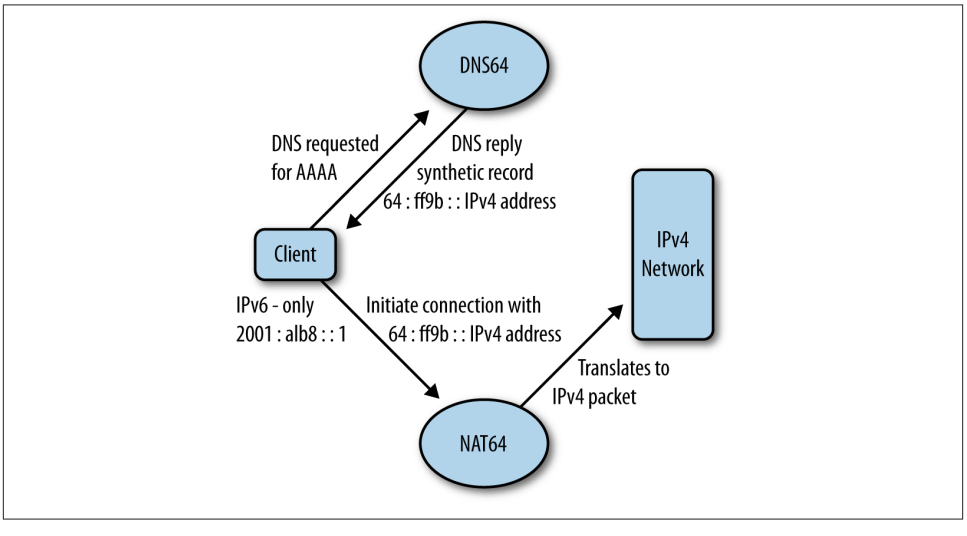

Figure 7-23. Stateful NAT64 and DNS64

The IPv6 client sends a DNS AAAA request to the DNS64 server for a certain domain name. If the name server has an AAAA record, it will pass the information and the client will connect over IPv6. If the DNS64 server does not have a AAAA record, because it is an IPv4-only service, it finds the corresponding A record and creates a synthetic record. The name server uses the well-known prefix of `64:ff9b::/96` or the specific prefix chosen and configured for this purpose and inserts the IPv4 address learned from the A record into the 32 low-order bits of the IPv6 address. So if the A record was `203.10.100.2`, the IPv6 address will be `64:ff9b::203.10.100.2`.

When the client initializes a connection to this address it will be routed through the NAT64 gateway, which will use an IPv4 address from its pool with an associated port number, creating a mapping entry for the two addresses. It will then translate the IPv6 header into an IPv4 header using the translation mechanisms described in RFC 6145, and send it to the destination IPv4 address, learned from the IPv6 address.

Stateful NAT64 only supports IPv6-initiated connections and is therefore suited for IPv6-only hosts to provide access to an IPv4 host. The advantage is that many IPv6-only hosts can connect to the IPv4 Internet through a single IPv4 public address. If an IPv4 device needs to speak to an IPv6-only device, the translation must be configured manually.

This has been tested and works well for general Internet access. Problems arise when IPv4 addresses are embedded in applications or when IPv4 literals are used. RFC 7050 defines a method how clients can discover a NAT64 prefix if they can't query a DNS64 server. Application developers should stick to using FQDNs (Fully Qualified Domain Names) in applications instead of IP addresses. A variety of vendors have implemented stateful NAT64 and large mobile providers are doing trials. In the mobile world this mechanism may be preferred as it uses less power on the mobile client (battery) than a dual-stack client.

NAT64 scenarios

As listed in RFC 6144, NAT64 can translate eight different scenarios. The following list shows which type of NAT64 supports the translation:

An IPv6 network to the IPv4 Internet
: Both stateless and stateful NAT64 can support this scenario.

The IPv4 Internet to an IPv6 network
: For this scenario to work with stateful NAT64, a DNS Application Level Gateway (ALG) would have to be used. This was deprecated with RFC 4966. Stateless NAT64 can be used for this scenario as it supports connections initiated by IPv4 nodes.

The IPv6 Internet to an IPv4 network
: Stateless NAT64 does not work with this scenario because stateless NAT64 only supports 1:1 address translation. The IPv4 address space could only support a small subset of the IPv6 address space. But IPv6-initiated connections can be supported through stateful NAT64. A network-specific prefix will be assigned to the translator that assigns the hosts IPv4-converted IPv6 addresses. Static AAAA records can be put into DNS to represent these IPv4-only hosts.

An IPv4 network to the IPv6 Internet
: For this scenario to work with stateful NAT64, a DNS Application Level Gateway (ALG) would have to be used. This was deprecated with RFC 4966. This scenario is not considered viable as these requirements will only occur in a later deployment state of the IPv6 Internet. For this scenario, other techniques should be considered.

An IPv6 network to an IPv4 network
: For this scenario, both networks are within the same organization. This scenario is the same as scenario one from a translation perspective. So both stateless and stateful translation can be used.

An IPv4 network to an IPv6 network
: For this scenario, both networks are within the same organization. This scenario is the same as scenario two from a translation perspective. So the same rules apply here.

The IPv6 Internet to the IPv4 Internet and vice versa
: Due to the huge difference in size between the two address spaces, there is no viable translation technique to handle unlimited IPv6 address translation.

If you are interested in operational experience with NAT64 in combination with CGN or as a server frontend (FE) mechanism, please refer to RFC xxxx (*draft-ietf-v6ops-nat64-experience-10.txt*). It is a report of operational deployment and testing of a NAT64 service between an IPv6-only mobile network and the larger IPv4 Internet as well as a NAT64 service in an IDC environment. This testing includes the use of NAT64 CGN and NAT64 FE; its coexistence with more traditional NAT44; reliability, availability, and maintainability issues; the transparency or lack of it regarding source addresses; quality of experience; MTU issues; and ULA-related issues.

RFC 6889, "Analysis of Stateful 64 Translation," analyzes how stateful NAT64 solves the issues that led to the deprecation of NAT-PT (RFC 2766).

464XLAT

464XLAT is not a separate transition mechanism. RFC 6877 describes an architecture that combines stateful translation in the core with stateless translation at the edge of the network. This provides IPv4 connectivity for IPv4-only applications across an IPv6-only network.

To discuss 464XLAT, we have to introduce two terms:

PLAT
: PLAT is a provider-side translator (XLAT) that complies with RFC 6146 on stateful NAT64. It translates N:1 global IPv6 addresses to global IPv4 addresses and vice versa.

CLAT
: CLAT is the customer-side translator (XLAT). It complies with RFC 6145 on IP/ICMP Translation Algorithms. It translates 1:1 private IPv4 addresses to global IPv6 addresses and vice versa. The CLAT can run on a router or on an end device such as a mobile phone. It performs IP routing to forward packets through the stateless translation.

464XLAT provides easy-to-use transition services as it requires no new protocols. It encourages the deployment of IPv6-only networks that are less expensive to operate than dual-stack networks. It is also needed in cases where there are no IPv4 addresses available anymore, but IPv6-only hosts need access to IPv4 applications. So it decouples network growth at the edge from IPv4 address availability.

Figure 7-24 shows the 464XLAT architecture.

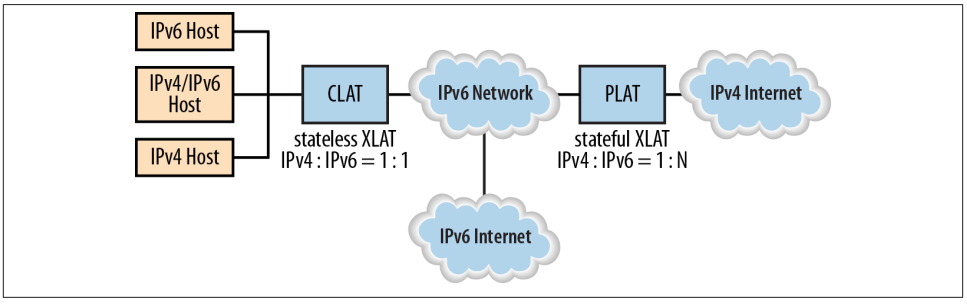

Figure 7-24. 464XLAT architecture

On the left side of the diagram, you see three different types of clients: an IPv6-only host, a dual-stack host with a private IPv4 address, and an IPv4-only host, also with a private IPv4 address. The IPv6 hosts can reach the IPv6 Internet directly without translation. IPv6 hosts can reach global IPv4 hosts through the PLAT (NAT64). The IPv4 hosts can reach global IPv4 hosts via stateless translation on the CLAT and stateful translation on the PLAT.

The 464XLAT address format follows the IPv4-embedded IPv6 address format described in the table in section 2.2 of RFC 6052, "IPv6 Addressing of IPv4/IPv6 Translators." The CLAT needs a /64 IPv6 prefix for the uplink interface and a /64 prefix for each downlink interface and a dedicated /64 prefix for the purpose of sending and receiving statelessly translated packets.

An IPv6-only host that wants to discover a NAT64 prefix sends a DNS query for a AAAA record for the domain `ipv4only.arpa` (RFC 7050) to its DNS server. If there is a NAT64 prefix, the host gets a DNS reply with a synthesized AAAA record. This AAAA record contains the IPv6 prefix plus the 32 bits of the IPv4 address for `ipv4only.arpa`. The CLAT uses the prefix to send packets to the translator.

This architecture supports IPv4 in the client-server model. It is not designed for IPv4 peer-to-peer communication or inbound IPv4 connections. It is based on IPv6 transport and supports native IPv6 communication. The advantage is that it also works for IPv4 applications that contain literals, as only the IP header is translated and the payload of the packet is encapsulated in the translated header.

MAP

A new mechanism has been developed called *MAP*. It comes in two flavors, both specifying mechanisms to map IPv4 to IPv6. Draft *"draft-ietf-softwire-map,"* called MAP-E, specifies the encapsulation of IPv4 packets in IPv6 including the address mapping with independence between IPv6 and IPv4 addresses. The other draft, *"draft-ietf-*

softwire-map-t," specifies the MAP-T mechanism, which uses the same address- and port-mapping algorithm and offers the same functionality as MAP-E, but does translation instead of encapsulation. Both mechanisms serve the purpose of delivering IPv4 services across an IPv6-only infrastructure, a situation and requirement which will become more and more common in the future.

The big advantage of MAP compared with CGN, NAT464, and DS-Lite is that it requires no central stateful translator on the service provider's network. This allows providers to deploy native IPv6 and share the scarce IPv4 address resources with much less overhead.

MAP is shown in Figure 7-25.

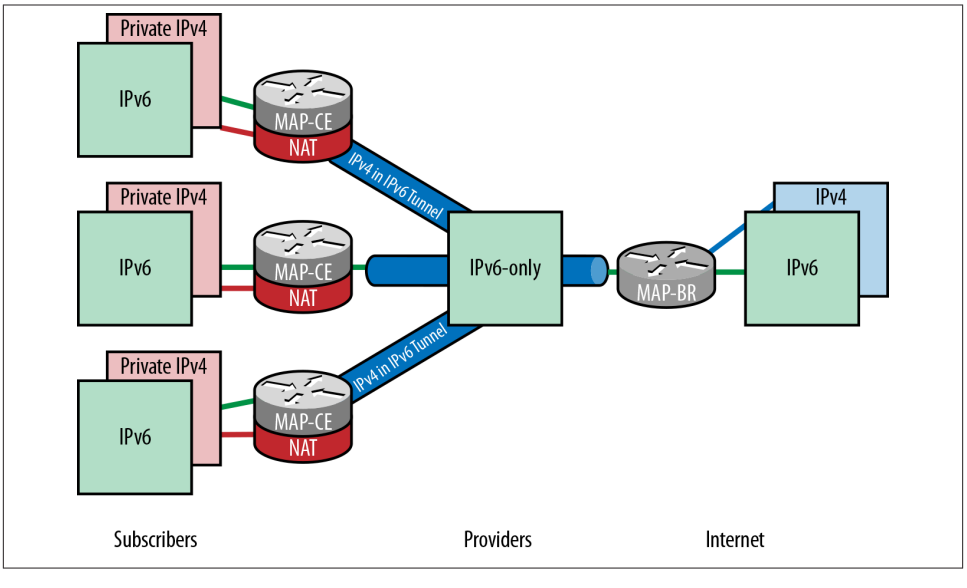

Figure 7-25. MAP

As you can see in the figure, the translation is moved to the CPE. All traffic through the provider network is now IPv6-only. This removes the need for a stateful CGN; the IPv6 traffic is forwarded by the Border Router (BR). The BR handles the traffic from a given MAP domain and is reachable via anycast.

MAP uses an IPv6-address-to-IPv4-address mapping with port-mapping algorithm. Specific bits in the IPv6 address space are used to represent both IPv4 addresses and ports. With MAP-T, the IP header is translated; with MAP-E, the IPv4 datagram is encapsulated in an IPv6 header. NAT44 as used with MAP differs slightly from traditional NAT44, as it allows assigning a port range to each of the CPEs sharing the same public IPv4 address. This address and port combination is then translated into the IPv6

address space on the MAP CPE. This stateless address mapping between IPv4 and IPv6 addresses removes the need for a large stateful translator in the provider network.

All nodes in a MAP domain must be provisioned with a set of parameters, which are used to implement the MAP functions. They can be configured manually or provisioned by DHCP. There are three rules, the *Basic Mapping Rule* (BMR), the *Default Mapping Rule* (DMR), and the *Forwarding Mapping Rules* (FMR). The mapping rules define the forwarding behavior for a MAP domain and make up the *Mapping Rule Table* (MRT), which serves as a routing table for the BR and CE.

MAP-E is going to be a standards track RFC, while MAP-T is going to be an informational or experimental RFC. There is also a draft specifying DHCPv6 options for address mapping, so a DHCP server can provision the information needed for the address-mapping algorithm (*draft-ietf-softwire-map-dhcp-07*).

NPTv6 and NAT66

A frequently asked question is whether there is or will be a counterpart to NAT (in this case, people usually refer to what we call NAT44) in IPv6. Given the fact that NAT was originally designed as a temporary solution to the IPv4 address depletion problem, you would expect a simple answer: no, because IPv6 is the long-term solution to the address exhaustion problem. And by design, the developers of IPv6 had end-to-end IPv6 networks in mind with no NAT.

Now, unfortunately people got very used to NAT in general and there are some use cases where some form of IPv6 NAT could be useful. So we can't get around discussing NAT for IPv6. Let us have a closer look at NPTv6.

IPv6-to-IPv6 Prefix Translation (NPTv6)

In the introduction to RFC 6296, "IPv6-to-IPv6 Network Prefix Translation," the IETF clearly states that they do not recommend the use of NAT technology for IPv6. So why do they publish a specification? One of the reasons is that they learned from the experience with IPv4 NAT. They did not publish a specification for NAT44 for the same reason: they did not think it was a good idea. This led to the fact that vendors started to implement their proprietary version of NAT44, which again led to the situation we face today, with multiple flavors of NAT44. This adds a lot to the complexity. So the IETF decided that even though they do not want to encourage NAT in IPv6 networks, it would be better to still have a specification, so that at least everybody who was doing it used the same mechanism.

NPTv6 is a stateless IPv6-to-IPv6 network-prefix translation mechanism. It provides address independence to the network. Address independence means that addresses used inside the local network do not need to be renumbered in case the global prefix changes (due to a provider change, for instance). Sessions can be initiated from both sides

(internal or external). You can also connect multiple NPTv6 Translators and also use them in multihoming scenarios. Section 2 in RFC 6296 describes some use cases.

There is a major difference between NPTv6 and NAT44, stemming from the fact that NPTv6 is not designed to save addresses. So the address-mapping is a 1:1 mapping and there is no need to modify port numbers and rewrite transport layer headers. In this respect, NPTv6 is less complex than traditional NAT44, but some issues still remain. For instance, it does not work with the IPsec Authentication Header, which provides protection for the IP header. Applications that transmit IP addresses in the payload may also stumble. The deployment of NPTv6 may also require configuring split DNS, as internal hosts want to resolve names for internal services to internal addresses, while external nodes need to obtain external addresses for the services. Some people perceive NAT44 to be a security feature due to the fact that it lets you hide your internal topology. NPTv6 does not hide your topology because the address mapping is 1:1. Only the prefix is translated and it must be a prefix of the same size. So if you want to translate a /48 internally, you also need a /48 externally. But you may choose to only translate a subset of your prefix, such as one or several /64 out of your /48 (for instance, the client subnets that need Internet access).

Another option to have independence in your address space is to apply for Provider Independent (PI) address space. But not all organizations qualify to receive a PI allocation. If you do qualify you have to ensure that your ISP(s) are willing to install specific routes for your prefix. This can be especially difficult or troublesome if you have locations in different geographical regions. It is not guaranteed that you will get a U.S. ISP to route a PI prefix that was allocated from RIPE (Europe region).

NAT66

NAT66 is the equivalent to NAT44. The difference to NPTv6 is that this translator is stateful. An IPv6 address on one interface is translated to a new IPv6 address on another interface of the router or firewall. Return traffic has to come the same way back, as the device keeps a state table of all the translations. With NAT66 the internal and external prefixes do not need to be of the same size. Whether this is a good idea or not is your choice. Most will probably choose to deal with prefixes of the same size anyway, for simplicity reasons. There are implementations from different vendors on the market. NAT66 has the same problems as NAT44 and there will be many workarounds needed to make all applications work through it. Going back to our design recommendations, this means carefully evaluate whether you really need and want this and only do this if there is no simpler solution.

Other Translation Techniques

There are additional translation mechanisms, which I describe in this section.

Bump-in-the-Host

Bump-in-the-Host (BIH) is a host-based IPv4-to-IPv6 protocol translation mechanism that allows a class of IPv4-only applications that work through NATs to communicate with IPv6-only peers. It is defined in RFC 6535. The host on which applications are running may be connected to IPv6-only or dual-stack access networks. BIH hides IPv6 and makes the IPv4-only applications think they are talking with IPv4 peers by local synthesis of IPv4 addresses. This document obsoletes RFC 2767 (Bump-in-the-Stack) and RFC 3338 (Bump-in-the-API).

Transport Relay Translator

The Transport Relay Translator (TRT; see RFC 3142) is a translation mechanism to be used in an IPv6-only network on the transport layer. It sits in the IPv6 network and allows communication between IPv6 nodes and IPv4 nodes. Every communication of an IPv6 client with an IPv4 application needs to go through the Relay Translator. In case of a TCP connection, the relay terminates the connection to the client and makes a new TCP connection on the other side to the IPv4 application. Internally, the translator translates between the two sessions. In case of a UDP connection, the translator simply translates and forwards the packet.

All translation techniques should be used only if there is no other choice. The overview in this chapter aims to give an idea of the variety of mechanisms to enable coexistence and smooth transition. The most important goal the developers had in mind was to provide mechanisms to give customers the possibility to move to an IPv6 network as soon as possible. The sooner you have an IPv6-dominant network, the better, because maintaining one protocol is always less costly than maintaining two.

Load Balancing

If you have either IPv4 servers and applications and want to give access for IPv6 users, or the other way around, there are two potential options that cover this scenario:

- Use stateless NAT64 (or NAT46, respectively)
- Use load balancers

The stateless NAT is easy to implement and does not require much resources. The load balancer choice is a common one and a good short-term solution, since load balancers are essentially mandatory pieces of equipment for the frontend in data centers anyway. Different load balancer vendors offer high-performance load balancers that support

many different mechanisms and can be used in most scenarios. But make sure to test your specific planned scenario for performance under load and in dual-stack mode if that is going to be used.

Comparison

Now that you have an overview of the available techniques, I'll summarize them by listing advantages and disadvantages. This summary should help you determine which way to go and which combinations to choose.

Dual-Stack

This technique is easy to use and flexible. It is your best option. Hosts can communicate with IPv4 hosts using IPv4 or communicate with IPv6 hosts using IPv6. When everything has been upgraded to IPv6, the IPv4 stack can simply be disabled or removed. Whenever you can, deploying dual-stack hosts and routers offers the greatest flexibility in dealing with islands of IPv4-only applications, equipment, and networks. Dual-stack is also the basis for other transition mechanisms. Tunnels need dual-stacked endpoints, and translators need dual-stacked gateways.

Disadvantages of this technique include the following: you have two separate protocol stacks running, so you need additional CPU power and memory on the host. All the tables are kept twice: one per protocol stack. Generally, all applications running on the dual-stack host must be capable of determining whether this host is communicating with an IPv4 or IPv6 peer. In a dual-stack network, you need to have a routing protocol for each version of IP. If you are using dual-stack techniques, make sure that you have firewalls in place that protect not only your IPv4 network, but also your IPv6 network, and remember that you need separate security concepts and firewall rules for each protocol.

It also makes troubleshooting problems more complicated. For instance, did an application that has problems with IPv6 attempt to connect via IPv4 instead of IPv6 and fail? How do you have to adjust your troubleshooting approaches to test and figure that out? Your helpdesk and IT support staff also need to understand how to use specific tools for IPv4 and IPv6 so they can rule out one protocol versus the other. So from an operational and support perspective, it may cost more to run a dual-stack network. This is one of the main reasons many enterprises consider to migrate to an IPv6-only infrastructure as soon as possible.

Tunneling

Tunneling allows you to migrate to IPv6 just the way you like. There is no specific upgrade order that needs to be followed. You can even upgrade single hosts or single subnets within your corporate network and connect separated IPv6 clouds through

tunnels. You don't need your ISP to support IPv6 in order to access remote IPv6 networks because you can tunnel through their IPv4 infrastructure. And you don't need to upgrade your backbone first. As long as your backbone is IPv4, you can use tunnels to transport IPv6 packets over the backbone. If you have an MPLS infrastructure, you have the best foundation for using this to tunnel IPv6 packets as long as you do not want to upgrade the backbone routers to support IPv6 natively.

The disadvantages are known from other tunneling techniques used in the past. Additional load is put on the router. A general rule is that stateless tunnels are preferred over stateful tunnels. The tunnel entry and exit points need time and CPU power for encapsulating and decapsulating packets. They also represent single points of failure. Troubleshooting gets more complex because you might run into hop count or MTU size issues, as well as fragmentation problems. Management of encapsulated traffic (e.g., per-protocol accounting) is also more difficult due to encapsulation. Tunnels also offer points for security attacks. Find more information on security issues in Chapter 6.

RFC 7059, "A Comparison of IPv6-over-IPv4 Tunnel Mechanisms," provides a nice overview of currently available tunnel mechanisms, including considerations to make when choosing a suitable tunnel mechanism.

Translation

Translation should be used only if no other technique is possible and should be viewed as a temporary solution until one of the other techniques can be implemented. The disadvantages are that translation between IPv4 and IPv6 does not support the advanced features of IPv6, such as Extension headers and end-to-end security. It poses limitations on the design topology because replies have to come through the same NAT router from which they were sent. The NAT router is a single point of failure, and flexible routing mechanisms cannot be used. All applications that have IP addresses in the payload of the packets will stumble. The advantage of this method is that it allows IPv6 hosts to communicate directly with IPv4 hosts and vice versa. In certain cases NAT may help to deploy IPv6-only networks in an early stage. Pros and cons have to be considered, but this may be a viable solution.

Now that you have mastered the IPv6 basics and your integration options, it is time to put it all in place and start the planning of your transition. Refer to Chapter 9 for putting all the pieces together and understanding the planning process. It also contains some guidelines on IPv6 addressing concepts.

You may also refer to my companion book, *Planning for IPv6* (O'Reilly), to get more details on planning and the design considerations that are important.

References

Here's a list of the most important RFCs and drafts mentioned in this chapter. Sometimes I include additional subject-related RFCs for your personal further study.

RFCs

- RFC 2185, "Routing Aspects of IPv6 Transition," 1997
- RFC 2473, "Generic Packet Tunneling in IPv6 Specification," 1998
- RFC 2529, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels," 1999
- RFC 2553, "Basic Socket Interface Extensions for IPv6," March 1999
- RFC 2663, "IP Network Address Translator (NAT) Terminology and Considerations," 1999
- RFC 2784, "Generic Routing Encapsulation (GRE)," 2000
- RFC 3022, "Traditional IP Network Address Translator (Traditional NAT)," 2001
- RFC 3053, "IPv6 Tunnel Broker," 2001
- RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds," 2001
- RFC 3068, "An Anycast Prefix for 6to4 Relay Routers," 2001
- RFC 3107, "Carrying Label Information in BGP-4," 2001
- RFC 3142, "An IPv6-to-IPv4 Transport Relay Translator," 2001
- RFC 3162, "RADIUS and IPv6," 2001
- RFC 3484, "Default Address Selection for Internet Protocol version 6 (IPv6)," 2003
- RFC 3489, "STUN—Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," 2003
- RFC 3493, "Basic Socket Interface Extensions for IPv6," 2003
- RFC 3542, "Advanced Sockets Application Program Interface (API) for IPv6," 2003
- RFC 3582, "Goals for IPv6 Site-Multihoming Architectures," 2003
- RFC 3715, "IPsec-Network Address Translation (NAT) Compatibility Requirements," 2004
- RFC 3901, "DNS IPv6 Transport Operational Guidelines," 2004

- RFC 3964, "Security Considerations for 6to4," 2004
- RFC 3971, "Secure Neighbor Discovery," 2005
- RFC 3972, "Cryptographically Generated Addresses (CGA)," 2005
- RFC 4007, "IPv6 Scoped Address Architecture," 2005
- RFC 4029, "Scenarios and Analysis for Introducing IPv6 into ISP Networks," 2005
- RFC 4038, "Application Aspects of IPv6 Transition," 2005
- RFC 4057, "IPv6 Enterprise Network Scenarios," 2005
- RFC 4159, "Deprecation of 'ip6.int,'" 2005
- RFC 4177, "Architectural Approaches to Multihoming for IPv6," 2005
- RFC 4191, "Default Router Preferences and More-Specific Routes," 2005
- RFC 4192, "Procedures for Renumbering an IPv6 Network without a Flag Day," 2005
- RFC 4213, "Basic Transition Mechanisms for IPv6 Hosts and Routers," 2005
- RFC 4215, "Analysis on IPv6 Transition in Third Generation Partnership Project (3GPP) Networks," 2005
- RFC 4218, "Threats Relating to IPv6 Multihoming Solutions," 2005
- RFC 4219, "Things Multihoming in IPv6 (MULTI6) Developers Should Think About," 2005
- RFC 4241, "A Model of IPv6/IPv4 Dual Stack Internet Access Service," 2005
- RFC 4282, "The Network Access Identifier," 2005
- RFC 4380, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," 2006
- RFC 4554, "Use of VLANs for IPv4-IPv6 Coexistence in Enterprise Networks," 2006
- RFC 4659, "BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN," 2006
- RFC 4779, "ISP IPv6 Deployment Scenarios in Broadband Access Networks," 2007
- RFC 4787, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP," 2007
- RFC 4798, "Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)," 2007
- RFC 4852, "IPv6 Enterprise Network Analysis—IP Layer 3 Focus," 2007
- RFC 4966, "Reasons to Move the Network Address Translator—Protocol Translator (NAT-PT) to Historic Status," 2007
- RFC 5157, "IPv6 Implications for Network Scanning," 2008

- RFC 5181, “IPv6 Deployment Scenarios in 802.16 Networks,” 2008
- RFC 5214, “Intra-Site Automatic Tunnel Addressing Protocol (ISATAP),” 2008
- RFC 5220, “Problem Statement for Default Address Selection in Multi-Prefix Environments,” 2008
- RFC 5375, “IPv6 Unicast Address Assignment Considerations,” 2008
- RFC 5569, “IPv6 Rapid Deployment on IPv4 Infrastructures (6rd),” 2010
- RFC 5571, “Softwire Hub and Spoke Deployment Framework with Layer Two Tunneling Protocol Version 2 (L2TPv2),” 2009
- RFC 5572, “IPv6 Tunnel Broker with the Tunnel Setup Protocol,” 2010
- RFC 5902, “IAB Thoughts on IPv6 Network Address Translation,” 2010
- RFC 5969, “IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)—Protocol Specification,” 2010
- RFC 5991, “Teredo Security Updates,” 2010
- RFC 6036, “Emerging Service Provider Scenarios for IPv6 Deployment,” 2010
- RFC 6052, “IPv6 Addressing of IPv4/IPv6 Translators,” 2010
- RFC 6081. “Teredo Extensions,” 2011
- RFC 6144, “Framework for IPv4/IPv6 Translation,” 2011
- RFC 6145, “Stateless IP/ICMP Translation,” 2011
- RFC 6146, “Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers,” 2011
- RFC 6147, “DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers,” 2011
- RFC 6164, “Using 127-Bit IPv6 Prefixes on Inter-Router Links,” 2011
- RFC 6177, “IPv6 Address Assignment to End Sites,” 2011
- RFC 6180, “Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment,” 2011
- RFC 6250, “Evolution of the IP Model,” 2011
- RFC 6269, “Issues with IP address sharing,” 2011
- RFC 6296, “IPv6-to-IPv6 Network Prefix Translation,” 2011
- RFC 6302, “Logging Recommendations for Internet-Facing Servers,” 2011
- RFC 6333, “Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion,” 2011
- RFC 6334, “Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Option for Dual-Stack Lite,” 2011

- RFC 6343, "Advisory Guidelines for 6to4 Deployment," 2011
- RFC 6434, "IPv6 Node Requirements," 2011
- RFC 6459, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)," 2012
- RFC 6535, "Dual-Stack Hosts Using "Bump-in-the-Host" (BIH)," 2012
- RFC 6540, "IPv6 Support Required for All IP-Capable Nodes," 2012
- RFC 6586, "Experiences from an IPv6-only Network," 2012
- RFC 6619, "Scalable Operation of Address Translators with Per-Interface Bindings," 2012
- RFC 6791, "Stateless Source Address Mapping for ICMPv6 Packets," 2012
- RFC 6830, "The Locator/ID Separation Protocol (LISP)," 2013
- RFC 6831, "The Locator/ID Separation Protocol (LISP) for Multicast Environments," 2013
- RFC 6832, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites," 2013
- RFC 6833, "Locator/ID Separation Protocol (LISP) Map-Server Interface," 2013
- RFC 6834, "Locator/ID Separation Protocol (LISP) Map-Versioning," 2013
- RFC 6835, "The Locator/ID Separation Protocol Internet Groper (LIG)," 2013
- RFC 6836, "Locator/ID Separation Protocol Alternative Logical Topology (LISP +ALT)," 2013
- RFC 6866, "Problem Statement for Renumbering IPv6 Hosts with Static Addresses in Enterprise Networks," 2013
- RFC 6877, "464XLAT: Combination of Stateful and Stateless Translation," 2013
- RFC 6879, "IPv6 Enterprise Network Renumbering Scenarios, Considerations, and Methods," 2013
- RFC 6883, "IPv6 Guidance for Internet Content Providers and Application Service Providers," 2013
- RFC 6886, "NAT Port Mapping Protocol /NAT-PMP), 2013
- RFC 6888, "Common Requirements for Carrier-Grade NATs (CGNs)," 2013
- RFC 6889, "Analysis of Stateful 64 Translation," 2013
- RFC 6908, "Deployment Considerations for Dual-Stack Lite," 2013
- RFC 6911, "RADIUS Attributes for IPv6 Access Networks," 2013
- RFC 6921, "Design Considerations for Faster-Than-Light (FTL) Communication," April 1, 2013

- RFC 7021, "Assessing the Impact of Carrier-Grade NAT on Network Applications," 2013
- RFC 7040, "Public IPv4-over-IPv6 Access Network," 2013
- RFC 7050, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis," 2013
- RFC 7051, "Analysis of Solution Proposals for Hosts to Learn NAT64 Prefix," 2013
- RFC 7059, "A Comparison of IPv6-over-IPv4 Tunnel Mechanisms," 2013
- RFC 7084, "Basic Requirements for IPv6 Customer Edge Routers," 2013
- RFC 7157, "IPv6 Multihoming without Network Address Translation," 2014
- RFC 7225, "Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol (PCP)," 2014

Drafts

Drafts can be found at *http://www.ietf.org/ID.html*. To locate the latest version of a draft, refer to *https://datatracker.ietf.org/public/pidtracker.cgi*. You can enter the draft name without a version number and the most current version will come up. If a draft does not show up, it was possibly deleted. If it was published as an RFC, the RFC number will be displayed. *http://tools.ietf.org/wg* is also a very useful site. More information on the process of standardization, RFCs, and drafts can be found in the Appendix A.

Here's a list of drafts I refer to in this chapter, as well as interesting drafts that relate to the topics in this chapter:

"Mapping of Address and Port with Encapsulation (MAP)"
draft-ietf-softwire-map-10

"Mapping of Address and Port using Translation (MAP-T)"
draft-ietf-softwire-map-t-05

"DHCPv6 Options for Configuration of Softwire Address and Port Mapped Clients"
draft-ietf-softwire-map-dhcp-07

"Lightweight 4over6: An Extension to the DS-Lite Architecture"
draft-ietf-softwire-lw4over6-08

"IPv4 Residual Deployment via IPv6 - a Stateless Solution (4rd)"
draft-ietf-softwire-4rd-08

"NAT64 Deployment Options and Experience"
draft-ietf-v6ops-nat64-experience-10

CHAPTER 8
Mobile IPv6

In the past, we were used to making phone calls from home or from the office. Public pay phones allowed us to make phone calls while on the road. Today, the use of mobile phones is common and we make phone calls from almost anywhere and in any life situation. The use of notebook computers, wireless networks, and portable devices is expanding, and we can imagine having smart devices and using them from wherever we are. If these devices are to use IP as a transport protocol, we need Mobile IP to make this work. We expect our device to remain connected when we move around and change our point of attachment to the network, just as we are used to roaming from one cell to the next with our mobile phones today. For example, suppose you have a tablet with an 802.11 (wireless) interface and a UMTS (Universal Mobile Telecommunication System) interface. In your hotel room, you are connected to the network through your wireless interface; when you leave your room and go out to the street, you switch automatically to UMTS without losing your connection. All the applications, such as your Skype session or your voice call running on your tablet, don't drop. Isn't this cool? This section about Mobile IP explores the mechanisms needed and shows how IPv6 is ready for this challenge.

With IPv4 and IPv6 alike, the prefix (subnet address) changes depending on the network to which we are attached. When a mobile node changes its point of access to the network, it needs to get a new IP address, which disrupts its TCP or UDP connections. RFC 5944, "IP Mobility Support for IPv4," describes Mobile IP concepts and specifications for IPv4. Using Mobile IP with IPv4 has certain limitations, though, which make it unsuitable for the requirements in a global network. One reason is the limited address space. If we even imagine only smartphones having an IP address, the number of addresses required globally to cover the number of devices far exceeds the IPv4 address space available. The other reason is that IPv6 and the use of Extension headers offers the possibility to optimize routing in a mobile world, and this is really necessary if we talk about mobility for large masses of devices. The fact that IPv6 uses Neighbor Discovery (instead of ARP

like IPv4) makes IPv6 more independent of the Link layer. Mobile IPv6 takes the experience from Mobile IPv4 and makes use of the advanced features of IPv6.

This chapter describes how Mobile IPv6 works and how it is suited to provide the foundation for mobile services of tomorrow. First, I explain the most important terms that will be used throughout the chapter, then I provide an overview of the functionality, and after this I dive into the technical details of the protocol: the new headers, messages, options, processes, and communications. So take a deep breath and read on.

Overview

Mobile IPv6 is a protocol that allows a mobile node to move from one network to another without losing its connections. It is specified in RFC 6275.

Most Internet traffic uses TCP connections. A TCP connection is defined by the combination of IP address and port number of both endpoints of the communication. If one of these four numbers changes, the communication is disrupted and has to be reestablished. If a mobile node connects to a different network, it needs a new IP address. Mobile IP addresses the challenge of moving a node to a different connection point without changing its IP address by assigning the interface of the mobile node a new additional IP address. The mobile node now still knows its *home address*, which is static and does not change, and it is therefore used to identify the TCP connection. The new IP address is called the *Care-of address*. It changes depending on the network to which the node is currently attached. So this works within homogeneous networks (if the node moves from an Ethernet segment to another Ethernet segment) but also in heterogeneous networks (if the node moves from an Ethernet segment to a wireless LAN or UMTS).

In a wireless network, we are familiar with the *handover*, the event where a device moves from one access point to another. This is a handover on the Link layer. Mobile IPv6 solves the handover issue on the Network layer and maintains connections to applications and services if a device changes its temporary IP address.

Mobile IPv6 Terms

Here are the definitions of some terms that are used throughout the chapter.

Home address
: A global unicast address assigned to a mobile node. It is used as the permanent address for this node and is within the mobile node's home link. Regular routing mechanisms deliver packets to the home address of the mobile node.

Home subnet prefix
: The IP subnet prefix corresponding to a mobile node's home address.

Home link
: The link on which a mobile node's home subnet prefix is defined.

Mobile node
: A node that can change its point of attachment from one link to another while still being reachable via its home address.

Correspondent node
: A peer node with which a mobile node is communicating. The correspondent node may be either mobile or stationary.

Foreign subnet prefix
: Any IP subnet prefix other than the mobile node's home subnet prefix.

Foreign link
: Any link other than the mobile node's home link.

Care-of address
: A global unicast address for the mobile node while it is in a foreign network (away from home). The subnet prefix of the Care-of address is the foreign subnet prefix. A mobile node may have multiple Care-of addresses. The one being registered with its home agent is the primary Care-of address.

Home agent
: A router on a mobile node's home link with which the mobile node has registered its current Care-of address. While the mobile node is away from home, the home agent intercepts packets on the home link destined to the mobile node's home address, encapsulates them (IPv6 encapsulation), and tunnels them to the mobile node's registered Care-of address.

Binding
: The association of the home address of a mobile node with a Care-of address for that mobile node, along with the remaining lifetime of that association.

Registration
: The process during which a mobile node sends a Binding Update to its home agent or a correspondent node, causing a binding for the mobile node to be registered. The registration with the correspondent node is called Correspondent Registration.

Binding authorization
: A registration with a correspondent node needs to be authorized to allow the recipient to ensure that the sender has the right to specify a new binding.

Return Routability Procedure
: A procedure that authorizes registrations by the use of a cryptographic token exchange.

Keygen token
: A number supplied by a correspondent node in the Return Routability Procedure to enable the mobile node to compute the necessary binding management key for authorizing a Binding Update.

Nonce
: Random numbers used internally by the correspondent node in the creation of keygen tokens related to the Return Routability Procedure. The nonces are not specific to a mobile node and are kept secret within the correspondent node.

Nonce index
: Used to indicate which nonces have been used when creating keygen token values without revealing the nonces themselves.

How Mobile IPv6 Works

Figure 8-1 shows the components of Mobile IPv6 and how they interact.

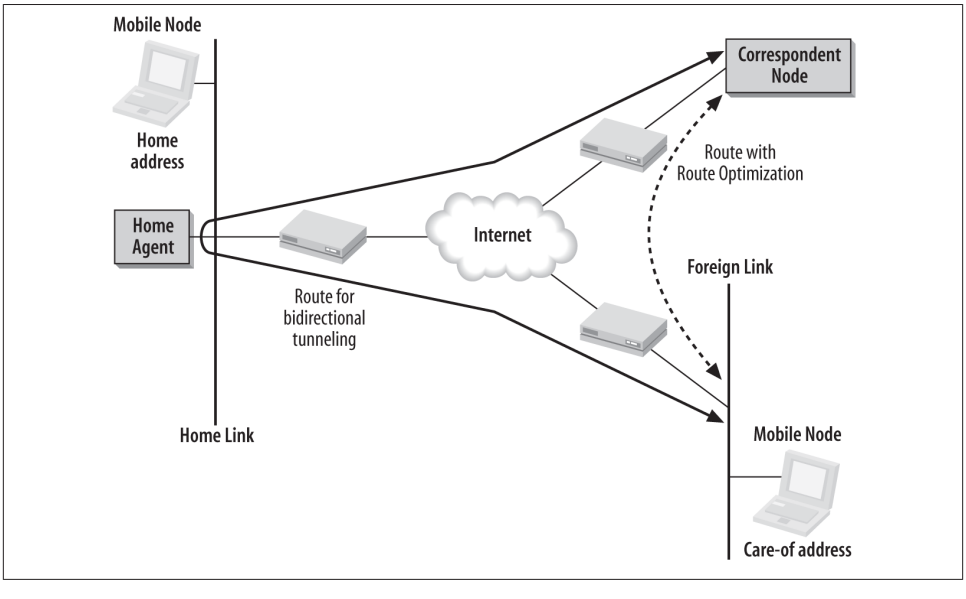

Figure 8-1. Overview Mobile IPv6

The home address is the IPv6 address within the home link prefix of a *Mobile Node* (*MN*). As long as the mobile node is at home, it receives packets through regular IP routing mechanisms and behaves like any other regular IP host. When the mobile node is away from home on a foreign link, it has an additional Care-of address. It receives the Care-of address through regular IPv6 mechanisms such as SLAAC (Stateless Address Autoconfiguration) or DHCPv6 when connecting to the new link.

The association of a home address and a Care-of address is called a *binding*. While away from home, the mobile node registers its Care-of address with a router on its home link, its *Home Agent* (*HA*). To register its Care-of address, the mobile node sends a binding update message to the home agent. The home agent responds with a binding acknowledgment. Every node communicating with a mobile node is called a *Correspondent Node* (*CN*). Mobile nodes can also send registrations to the correspondent node directly (called a correspondent registration). A correspondent node can also be a mobile node.

There are two ways to communicate for a correspondent node and a mobile node:

Bidirectional Tunneling
: Packets from the correspondent node are sent to the home agent, which encapsulates them in IPv6 and sends them to the Care-of address of the mobile node. Packets from the mobile node are sent through the reverse tunnel to the home agent that forwards them to the correspondent node through regular routing mechanisms. This mode does not require any Mobile IPv6 support on the correspondent node and works without correspondent registration.

Route Optimization
: With Route Optimization, the communication between mobile node and correspondent node can be direct without going through the home agent. This is one of the main advantages of Mobile IPv6 over Mobile IPv4, where Route Optimization is not possible. Route Optimization requires that the mobile node registers its Care-of address with the correspondent node (called Correspondent Registration) and that this binding is authorized through the Return Routability Procedure (discussed in the section Return Routability Procedure later in this chapter). The correspondent node uses a special Routing header (Type 2) when it sends packets to the mobile node directly. The mobile node uses the Home Address option (defined for Mobile IPv6) when sending packets to the correspondent node. The whole process is described in more detail later in the chapter.

The advantage of Route Optimization is that the shortest available path can be used between correspondent node and mobile node. The packets do not have to go through the home agent. This not only ensures shorter communication paths but also reduces the load on the home agent and the home link. This becomes very important when we talk about high numbers of mobile nodes constantly moving around, for instance in a VoIP (Voice over IP) scenario or when moving around with your smartphones.

Mobile IPv6 also supports the option to have multiple home agents, and the mobile node can learn about reconfiguration of its home link or a change of IP address of its home agent through Dynamic Home Agent Address Discovery. If the prefix of its home link changes, the mobile node uses the Mobile Prefix Discovery mechanism to learn about the new prefix.

The following sections describe the protocol and new messages, options, and flags in more detail. After this, I dive into the communication flows that have just been described in an overview. Some people prefer to learn this way; other people prefer to learn about the processes and flows first and then about the technical details. Please read the sections in the order that best fits your preference.

The Mobile IPv6 Protocol

This section describes the components, messages, and options for Mobile IPv6.

Mobility Header and Mobility Messages

The Mobility Header (MH) has been defined for Mobile IPv6. It is an Extension header used by mobile node, correspondent node, and home agent. It is used in all messages that are related to establishing and maintaining bindings.

A Mobility Header is specified by the Next Header value 135 in the preceding header and has the format shown in Figure 8-2.

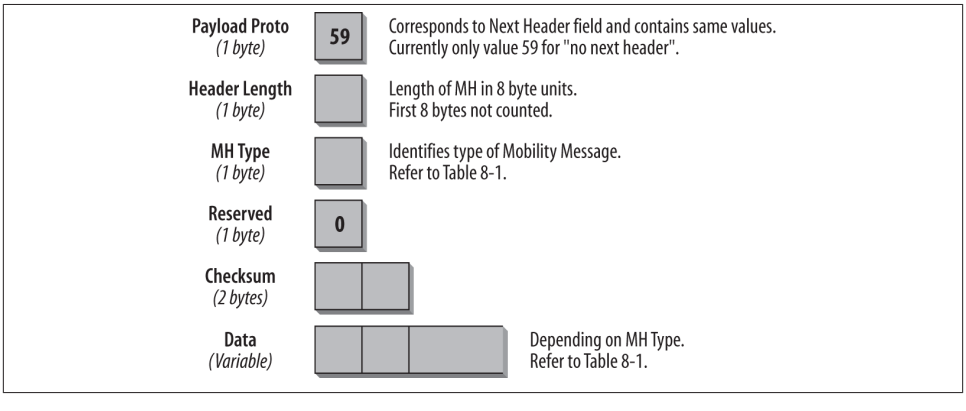

Figure 8-2. Format of the Mobility Header

The Payload Proto field corresponds to the Next Header field and identifies the following header. It can therefore contain the same values as the Next Header field. The current specification sets the value in this field to 59 decimal, which means "no next header." It is designed to be used for future extensions. The Header Length field contains the length of the Mobility Header in 8-byte units. The first 8 bytes are not counted. The length of the Mobility Header is always a multiple of 8 bytes. The Checksum field contains the checksum for the Mobility Header. It is calculated based on a pseudoheader and follows the rules defined in RFC 2460. The addresses used in the pseudoheader are the Source and Destination address in the IPv6 header. If the Mobility message contains a Home Address Destination option, the home address is used for the calculation of the

checksum. The MH Type field identifies the type of Mobility message. The messages defined are listed in Table 8-1. The Data field is variable; it depends on the type of message.

Table 8-1 is an overview of the Mobility messages.

Table 8-1. Mobility message types

Value	Message type	Description	RFC
0	Binding Refresh Request	Sent by CN requesting the MN to update its binding.	RFC 6275
1	Home Test Init	Sent by the MN to initiate the Return Routability Procedure and request a Home keygen token from a CN. Sent to the CN through the tunnel via HA.	RFC 6275
2	Care-of Test Init	Sent by the MN to initiate the Return Routability Procedure and request a keygen token from a CN. Sent to the CN directly.	RFC 6275
3	Home Test Message	Response to a Home Test Init message (type 1). Sent from the CN to MN. Contains a cookie and a Home keygen token for the authorization in the Return Routability Process. Sent through the tunnel via HA.	RFC 6275
4	Care-of Test Message	Response to Care-of Test Init message (type 2). Sent from CN to MN. Contains cookie and a Care-of keygen token for the authorization in the Return Routability Procedure. Sent to the MN directly.	RFC 6275
5	Binding Update	Sent by MN to notify a change of its Care-of address. This message is explained in more detail later in the chapter.	RFC 6275
6	Binding Ack	Sent as acknowledgment for receipt of a Binding Update message. This message is explained in more detail later in the chapter.	RFC 6275
7	Binding Error	Sent by CN to signal an error related to mobility, such as an inappropriate attempt to use the Home Address Destination option without an existing binding. The status field can have the following values: 1 = unkown binding for Home Address Destination option 2 = unrecognized MH type value	RFC 6275
8	Fast Binding Update	Identical to Binding Update message, only with slightly different processing rules.	RFC 5568
9	Fast Binding Ack	Sent as acknowledgment for receipt of a Fast Binding Update message.	RFC 5568
10	Fast Neighbor Advertisement	Deprecated.	RFC 5568
11	Experimental Mobility Header	Defined for testing of new message types without the risk to collide with standardized values.	RFC 5096
12	Home Agent Switch	Sent from HA to MN. Forces MN to configure a new HA.	RFC 5142
13	Heartbeat	Sent by MAC and LMA to verify status of reachability (Proxy Mobile IPv6).	RFC 5847
14	Handover Initiate	Sent between access routers to initiate the process of handover of MNs.	RFC 5568
15	Handover Ack	Sent by access routers to acknowledge the receipt of the Handover Initiate message.	RFC 5568
16	Binding Revocation	Sent by a mobility node to terminate a binding: 1 = Binding Revocation Indication 2 = Binding Revocation Ack	RFC 5846

Value	Message type	Description	RFC
17	Localized Routing Initiation	Sent by LMA or MAG to initiate localized routing (Proxy Mobile IPv6).	RFC 6705
18	Localized Routing Ack	Sent by LMA or MAG as response to Localized Routing Initiation.	RFC 6705

To help you understand the binding, the next section explores the Binding Update and the Binding Acknowledgment messages in more detail.

You can find all message and option types as well as status codes at *http://www.iana.org/assignments/mobility-parameters*.

The Binding Update Message

The Binding Update message is used by the mobile node to inform the home agent or a correspondent node about a new Care-of address. The message is also used to extend the lifetime of an existing binding.

The Binding Update message is of MH type 5 and has the format shown in Figure 8-3.

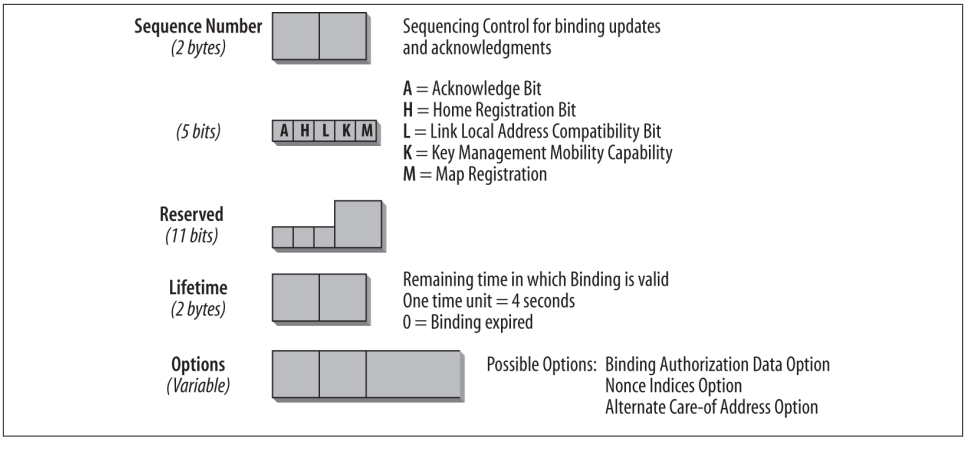

Figure 8-3. Format of the Binding Update message

The Sequence Number is used by the receiving node for sequencing Binding Updates. The sending node uses it to verify whether the Binding Acknowledgments received correspond to its Binding Updates. The Acknowledge bit (A-Bit) is set by the mobile node if it expects an acknowledgment in answer to its Binding Update. The Home Registration bit (H-Bit) is set by the mobile node to request the receiver to act as home agent for this node. This is possible only if the receiver is on the home link of the mobile

node. The Link-Local Address Compatibility bit (L-Bit) is set if the home address has the same Interface Identifier as the link-local address of the mobile node. The Key Management Mobility Capability bit (K-Bit) is valid only in Binding Updates sent to the home agent. IPsec Security Associations should survive the move of the mobile node to another network. If that is the case, the K-Bit is set. If that is not possible, the K-Bit is set to 0. Correspondent nodes ignore the K-Bit. The Lifetime shows for how long the binding for the Care-of address is valid (in four-second units). If the Lifetime is set to 0, the receiver must delete the entry in its Binding Cache. In this case, the mobile node must be on its home link, and the Care-of address is the same as the home address.

The M-Bit shown in Figure 8-3 has additionally been created to identify Local Binding Updates sent to a local home agent called a *Mobility Anchor Point* (MAP). This new node is used to improve Mobile IPv6 handover performance, to obtain efficient routing between the mobile node and correspondent nodes within the same geographical area, and to achieve location privacy. The mechanism is defined in RFC 4140, "Hierarchical Mobile IPv6," and is explained in more detail at the end of this chapter. When the M-Bit is set, the H-Bit cannot be set and vice versa.

A Binding Update can have the following options:

- Binding Authorization Data option (this option is mandatory in Binding Updates sent to a correspondent node)
- Nonce Indices option
- Alternate Care-of Address option

The Binding Acknowledgment

The Binding Acknowledgment is sent to confirm receipt of a Binding Update. It has to be sent if the A-Bit is set in the Binding Update. If the A-Bit is not set (which means the sender of the Binding Update does not require an acknowledgment), the Binding Acknowledgment is sent only if there is a problem in the Binding Update. If the receiver accepts the Binding Update and the A-Bit was not set, no acknowledgment is sent.

The Binding Acknowledgment is of MH type 6 and has the format shown in Figure 8-4.

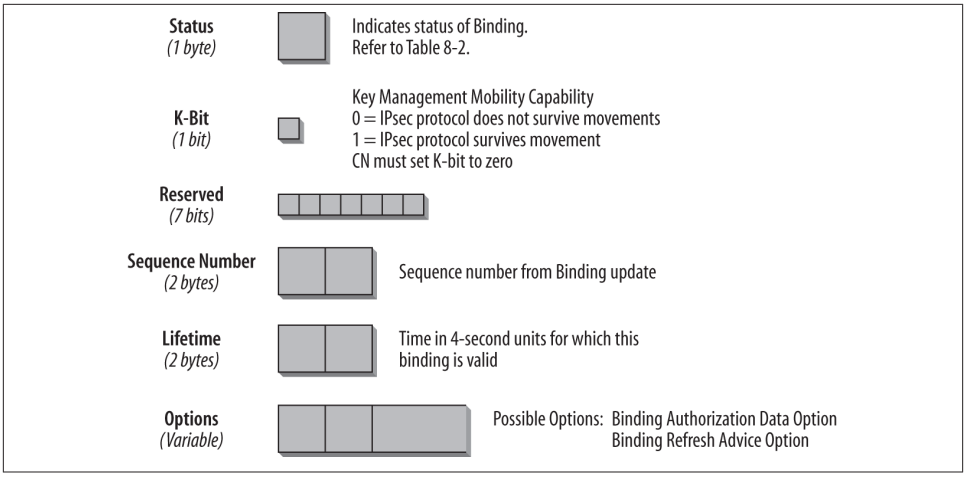

Figure 8-4. Format of the Binding Acknowledgment

The status field indicates the status of the Binding Update. Table 8-2 shows the status values. Values in the range of 0 to 127 indicate that the Binding Update has been accepted. Values above 128 indicate that the Binding Update has not been accepted.

Table 8-2. Status values in the Binding Acknowledgment

Value	Description
0	Binding Update accepted
1	Accepted but prefix discovery necessary
128	Reason unspecified
129	Administratively prohibited
130	Insufficient resources
131	Home Registration not supported
132	Not home subnet
133	Not home agent for this mobile node
134	Duplicate Address Detection failed
135	Sequence Number out of window
136	Expired home nonce index
137	Expired care-of nonce index
138	Expired nonces
139	Registration type change disallowed

These are the status values defined in RFC 6275. There are more status values from other specifications such as NEMO or Proxy Mobile IPv6 (discussed in the sections "NEMO" on page 308 and "Proxy Mobile IPv6" on page 310).

Find the full list of status codes at *http://www.iana.org/assignments/mobility-parameters*.

The K-Bit is the Key Management Mobility Capability bit (see the description earlier in the section "The Binding Update Message" on page 290). This bit is of importance only in bindings between mobile node and home agent. Correspondent nodes ignore this bit. The Sequence Number in the Binding Acknowledgment is copied from the Sequence Number field in the Binding Update. It is used by the mobile node in matching this Binding Acknowledgment with an outstanding Binding Update. The Lifetime shows in 4-second units for how long the binding for the Care-of address is valid. During the time indicated here, either the home agent or the correspondent node keeps the entry for this binding in its Binding Cache. In a Binding Acknowledgment that indicates that the binding has not been accepted (value of 128 or higher), the Lifetime is not specified.

A Binding Acknowledgment can have the following options:

- Binding Authorization Data option (this option is mandatory in Binding Acknowledgments sent by a correspondent node)
- Binding Refresh Advice option

The Binding Revocation

In certain cases it may be necessary to inform an MN that it can no longer receive Mobile IPv6 services for its home address. For this purpose a Binding Revocation mechanism has been defined in RFC 5846. It uses a new Mobility Header (type 16; refer to Table 8-1). This revocation mechanism can be used for Mobile IPv6 as well as for Proxy Mobile IPv6 (RFC 5213; described in the section "Proxy Mobile IPv6" on page 310).

New terms have been defined in this specification:

Initiator
: The mobility node that initiates the Binding Revocation process. It sends a Binding Revocation message to its peers (e.g., home agent, local mobility anchor, or mobile access gateway).

Responder
: The mobility node that receives the Binding Revocation message and responds with a Binding Revocation Acknowledgment (e.g., mobile node, mobile access gateway, or local mobility anchor).

To revoke a binding, the HA sends a *Binding Revocation Indication* (BRI) message to the MN. To indicate the binding, the HA includes the home address in the message (in

the Type 2 Routing header). In the case of dual-stack Mobile IP (DSMIPv6) the IPv4 home address option may be included also. If the MN has multiple Care-of addresses registered, the HA includes the Binding Identifier (BID) option in the revocation message, to identify which binding is revoked. When an MN receives a Binding Revocation Indication message with its home address in the Type 2 Routing header, it responds with a Binding Revocation Acknowledgment message.

Similarly, in the case of Proxy Mobile IPv6, the local mobility anchor is sending the revocation message to the mobile access gateway. In this case the local mobility anchor includes the mobile node home network prefix option and the mobile node identifier option.

Mobility Options

A mobility message can contain zero, one, or more options. These options are included in the Variable Data field of the Mobility header. This architecture is very flexible, as options are inserted only if needed and additional options can easily be defined in the future.

The presence of options is indicated in the Header Length field of the Mobility header. They have the known TLV format (Type 1 byte, Length 1 byte, Value variable).

Table 8-3 contains an overview of the currently defined options for mobility messages.

Table 8-3. Mobility options

Value length	Name	Description
Type 0	Pad1	Used to insert one padding byte. This option has a special format; it contains only a Type field, and no fields for length and data.
Type 1	PadN	Used to insert two or more padding bytes.
Type 2 Length 2	Binding Refresh Advice	Indicates the remaining time until the MN should send a new Home Registration to HA. Only valid in Binding Acks sent from the HA in response to a Home Registration. The interval must be shorter than the Lifetime value in the Binding Acknowledgment. A time unit is four seconds.
Type 3 Length 16	Alternate Care-of address	Contains an address to use as the Care-of address for the binding rather than using the Source address of the packet as the Care-of address. Only in Binding Update messages.
Type 4 Length 4	Nonce Indices	Has two additional fields besides the Type and Length field. The Home Nonce Index field tells the CN which nonce value to use when producing the Home Keygen Token. The Care-of Nonce field indicates the value for generating the Care-of Keygen Token. Valid only in the Binding Update message sent to a CN, and only when present together with a Binding Authorization Data option.

Value length	Name	Description
Type 5 Length variable	Binding Authorization Data	Contains a cryptographic value that can be used to determine that the message in question comes from the right authority. Rules for calculating this value depend on the authorization procedure used. Must always be the last option in the MH. Only valid in Binding Update and Binding Acknowledgment. Used for the Return Routability process. In this case, the calculation of the Authenticator value is based on Care-of address of MN, IPv6 address of CN, and data from the MH.
Type 201 Length 16	Home Address	Contains the home address of MN. Sent by MN when away from home to indicate its home address to the receiver. Carried in a Destination Options header. Must be inserted after Routing header and before Fragment, AH, or ESP header (if present).

With the exception of the Binding Authorization Data option, these options can appear in arbitrary order. The Home Address option is an exception, as it is carried in a Destination Options header and not in the Mobility Header (MH).

Again, there is a full list of mobility options including options from other specifications at *http://bit.ly/1nabH2o*.

RFC 4283, "Mobile Node Identifier Option for Mobile IPv6 (MIPv6)," extends the original specification to allow MIPv6 nodes (HA, CN, MN) to use identifiers other than an IP address. It defines an option with a subtype number to specify the identifier type. The identifier type can be a Network Access Identifier (NAI; see RFC 4282), an International Mobile Station Identifier (IMSI), or an application/deployment specific opaque identifier. Additional identifier types will be specified in the future.

Routing Header Type 2

A new Routing header has been defined for Mobile IPv6. This Extension header allows the data exchange between the Care-of address of a mobile node and a correspondent node without being routed through the home agent. In other words, it is used when communication is performed with Route Optimization after a successful Return Routability Procedure.

RFC 6275 defines the Type 2 Routing header. It allows, among other things, the configuration of specific rules for Mobile IPv6 packets on firewalls.

When a correspondent node sends an IPv6 datagram to a mobile node using Route Optimization, the Destination address field in the IPv6 header contains the Care-of address of the mobile node. The Routing header Type 2 inserted contains the home address of the mobile node. The Routing header Type 2 can only contain one unicast address. IPv6 nodes that process these Routing headers must verify that the IPv6 address contained corresponds to the home address of the mobile node.

The format of the Routing header Type 2 is shown in Chapter 3 (Figures 3-6 and 3-7). The Header Extension Length field has the value 2; this header does not have a variable length, as it only contains one address. In the Routing Type field, the value 2 is indicated, and the Segments Left field is set to 1 for one address. The Home Address field carries the home address of the mobile node. How this Routing header Type 2 is used and processed is described later in this chapter.

ICMPv6 and Mobile IPv6

This section describes two new ICMPv6 message pairs and some modifications to Neighbor Discovery (ND) that have been made to support Mobile IPv6.

Home Agent Address Discovery

The Home Agent Address Discovery mechanism is used by the mobile node to determine the address of its home agent on its home link. For this purpose the mobile node uses the two new ICMPv6 message pairs and the home agents list, which is a list maintained by each home agent.

ICMPv6 Home Agent Address Discovery messages

This message pair consists of a Home Agent Address Discovery Request and Reply message. As the name implies, the mobile node uses these messages to find its home agent on the home link dynamically. Normally, mobile nodes are configured statically with a home agent address. In the case where a home agent is renumbered or goes down and is replaced by another home agent with a different IP address, dynamic discovery of the home agent address may be a useful mechanism.

The mobile node sends a Discovery message to the Home Agent Anycast address (Anycast ID decimal 126, hexadecimal `0x7E`; see Chapter 2) on its home link. The Source address field in the IPv6 header carries the Care-of address of the mobile node.

The home agents on the home link that are configured for the Home Agent Anycast address respond with a Home Agent Address Discovery Reply message.

The Discovery message has a type value of 144; the Reply message has a type value of 145. The Code field is always set to 0. The Identifier field is inserted by the mobile node and copied over by the home agent for the Reply. This allows to identify corresponding messages. The Reply carries a Home Address field, which can carry one or more home agent addresses. This address or list of addresses is generated from the home agents list (described next).

For a detailed description of the header fields in an ICMPv6 message, please refer to Chapter 4.

Home agents list

Every home agent needs to maintain a home agents list. In this list, every router must be listed that is connected to the same link and provides home agent services. A router advertises itself as a home agent by setting the H-Bit in the Router Advertisement. A router maintains a home agents list for each link on which it acts as a home agent. The list is updated through Router Advertisements and contains the following information:

- Link-local address of the HAs on the link. This address is learned from the Source address field in the IPv6 header of Router Advertisements that have the H-Bit set (home agent bit).
- One or more global IPv6 unicast addresses for these HAs. These addresses are learned from the Prefix option in the Router Advertisements.
- Remaining Lifetime for this HA entry. When the Lifetime expires, the HA has to be deleted from the list.
- Preference for this HA. Higher values means higher preference. This value is learned from the Home Agent Preference field in the Home Agent Information option in a Router Advertisement (if present). If not present, this value is set to 0. The HA uses this preference to sort the HA list when sending out a Home Agent Address Discovery Reply message.

The HA sending out a Home Agent Address Discovery Reply must list all HAs on the link, sorted by preference. Only the global IPv6 unicast addresses of the home agents are contained in the home agent address field of the Home Agent Address Discovery Reply message. The reply must not be larger than 1,280 bytes. Sorting by preference ensures that HAs with a high priority are listed in this packet.

Mobile Prefix Solicitation

The Mobile Prefix Solicitation message is sent by a mobile node away from home to determine changes in the prefix configuration of its home link (i.e., home network renumbering). The HA answers the Solicitation message with a Prefix Advertisement. Based on this reply, the mobile node can adjust its home address.

The mobile node sends an ICMPv6 Mobile Prefix Solicitation message to its HA. The IPv6 header carries the Care-of address as Source address. A Home Address Destination option is inserted. IPsec headers are supported and should be used. The Mobile Prefix

Solicitation can carry options that have to follow the format described in Chapter 4 (RFC 4861). Currently, there are no specific options defined. The type value field of a Mobile Prefix Solicitation message is set to 146; the Code field is set to 0.

The HA replies with an ICMPv6 Mobile Prefix Advertisement message to the Care-of address of the mobile node. The HA can also send out unsolicited Advertisements at regular intervals. The Advertisement carries a Routing header Type 2. The reply is sent to the Source address of the Solicitation message. If the Advertisement is unsolicited, it is sent to the Care-of address of registered mobile nodes. The type value for the Advertisement is set to 147. If it is the answer to a Solicitation, the Identifier is copied over from the Solicitation. The Advertisement contains the Prefix option described in Chapter 4. It carries all prefixes that should be used by the mobile node to configure its home addresses.

Changes in Neighbor Discovery (ND)

Some changes and new options have been defined in ND for use with Mobile IPv6.

Modified Router Advertisement format

As already mentioned in Chapter 4, there is a new flag in the Router Advertisement. The M-Flag and the O-Flag are followed by the H-Flag, which allows a router to advertise that it acts as a home agent on this link.

Modified Prefix option

In order to build an updated HA list based on Router Advertisements, a mobile node must know the global unicast address of the routers. A regular Router Advertisement lists only the link-local address of the router. For this purpose, the Prefix option has been modified. The Prefix option now carries an additional flag, the R-Flag (router address). When this flag is set, it indicates that the Prefix option field does not contain a Prefix, but rather a global IPv6 unicast address for the router.

New Advertisement Interval option

The Advertisement Interval option is used in Router Advertisements. It indicates the interval at which the router will send unsolicited multicast Router Advertisements. The option follows the TLV format (Time, Length, Value) and has a type value of 7. The Advertisement Interval field has 4 bytes and carries the time in milliseconds between unsolicited Router Advertisements. The mobile node uses this information in its Movement Detection Algorithm (described later in this chapter).

The Neighbor Discovery specification sets a minimum interval of three seconds for unsolicited multicast Router Advertisements. Mobile nodes are dependent on learning as fast as possible when moving to a new network to create new Care-of addresses accordingly and send out Binding Updates. They detect their movement to a new

network based on Router Advertisements from routers they don't know yet. As a consequence, routers supporting Mobile IPv6 must be configurable with a shorter Router Advertisement Interval. Alternatively, mobility information from lower layers (i.e., Layer 2) can be used to aid the mobile node in achieving faster movement detection.

New Home Agent Information option

The Home Agent Information option is used in Router Advertisements and follows the TLV format. The type value is 8.

The HA Preference field has a length of 2 bytes. In a Router Advertisement, the HA can use this field to indicate which level of preference should be associated to it. A higher value means a higher preference. When this option is not set, the HA has a preference of 0. This field can be used by the HA to dynamically adjust to different situations, e.g., to the number of mobile nodes currently connected or based on how many resources are available to serve additional mobile nodes. Alternatively, the preference can be configured manually.

The Home Agent Lifetime field also has a length of 2 bytes. It indicates the lifetime for this HA in seconds. The default value corresponds to the value in the basic Router Advertisement header. The maximum possible value is 18.2 hours. A value of 0 is not accepted. The HA Lifetime field only relates to the HA service of this router, so it can be present only in a Router Advertisement with the H-Bit (home agent) set.

Mobile IPv6 Communication

This section discusses Mobile IPv6 terms and goes into more details on the communication processes.

Binding Cache

Every correspondent node and home agent maintains a Binding Cache for each of its global IPv6 addresses. It lists all mobile nodes for which it has a binding. If it wants to send data to a certain destination, it first searches its Binding Cache, and after this, the Destination Cache for an address.

A Binding Cache entry carries the following information:

- Home address of the mobile node for the entry. This field is used to determine the Destination address when sending a packet.
- Care-of address for the mobile node indicated by the Home Address field in this Binding Cache entry.
- Lifetime value for the binding.

- A flag indicating whether this entry is a home registration entry. Present only on a node that acts as a home agent.
- Maximum value of the Sequence Number field of all previous Binding Updates received for this home address.
- Information on the use of this entry.

Binding Update List

Every mobile node maintains a Binding Update List. The list has an entry for each Binding Update the mobile node has sent to its home agent(s) and to correspondent nodes for which the Lifetime has not expired. If it has sent more than one Binding Update, only the last message with the highest Sequence Number is listed.

A Binding Update List carries the following information:

- IPv6 address of the node to which the Binding Update has been sent
- Home address for which the Binding Update has been sent
- Care-of address that was indicated in this Binding Update
- Lifetime that was indicated in the Binding Update
- Remaining Lifetime for the Binding
- Highest used Sequence Number for this Binding
- Time when Binding Update was sent
- State of any retransmissions needed
- A flag indicating whether further Binding Updates have to be sent to this destination

Return Routability Procedure

The Return Routability Procedure is designed to allow a correspondent node to detect whether the mobile node is reachable at its Care-of address as well as at its home address. Only when this has been successfully proven can *Route Optimization* (i.e., direct communication between correspondent node and mobile node) be used. The fact that the mobile node can be reached at both addresses indicates that it really is on the foreign link and has a valid registration for the home address. This reduces (but does not eliminate) the risk that this Binding Update is a security attack. Only after a successful Return Routability test does the correspondent node accept Binding Updates from the mobile node and send datagrams to the Care-of address of the mobile node directly.

The message flow for the Return Routability Procedure consists of the following steps (for MH types, refer to Table 8-1):

1. MN sends a Home Test Init message (MH type 1) via HA to the CN. This message carries a Home Init Cookie. This way the CN learns the home address of the MN.
2. MN sends a Care-of Test Init message (MH type 2) to the CN. This message carries a Care-of Init Cookie. It is sent to the CN directly (not through HA). This way, the CN learns the Care-of address of the MN.
3. CN replies to the Home Test Init message with a Home Test message (MH type 3) sent via HA. It carries the Home Init Cookie, the Home Keygen Token, and the Home Nonce Index. The MN can now generate a Home Keygen Token.
4. CN replies to the Care-of Test Init message with a Care-of Test message (MH type 4) sent to the MN's Care-of address. It carries the Care-of Init Cookie, the Care-of Keygen Token, and the Care-of Nonce Index. The MN can now generate a Care-of Keygen Token.

Once the mobile node has received the Home Test and the Care-of Test messages, the Return Routability Procedure has been accomplished. The mobile node hashes the two tokens and creates a 20-byte Management Key, which is obviously also known to the correspondent node that generated the two tokens in the first place. This key will be used by the mobile node to secure the Binding Update to the correspondent node. Upon a successful security check, the correspondent node can accept the Binding Update since the mobile node has proven that it is reachable on the home and Care-of addresses contained in the Binding Update.

RFC 4225, "Mobile IP Version 6 Route Optimization Security Design Background," outlines the security considerations and choices that were made when the Return Routability Procedure was defined. The goal of this informational document is to help implementers of MIPv6 understand the design choices and to help people who design mobility of multihoming solutions to avoid some common security pitfalls. The security problems and possible countermeasures are discussed in detail.

Home Agent Operation

When the mobile node is away from home, the HA must intercept all packets destined to the mobile node and tunnel them to the Care-of address of the mobile node. It uses Proxy Neighbor Discovery to do so.

Proxy Neighbor Discovery

In order to intercept packets destined to the mobile node on the home link, the HA must pretend to be the mobile node. The HA sends Neighbor Advertisements to the all-nodes multicast address, providing its own link-layer address as link-layer address for the home address of the mobile node. The ND message has the following information:

- The Source address in the IPv6 header of the Neighbor Advertisement is the address of the HA.
- The Target Address field in the ND message carries the IPv6 address of the mobile node.
- The ND Advertisement contains a Target Link-Layer Address option carrying the link-layer address of the HA.
- The Router flag (R-Flag) must be set to 0.
- The Override flag (O-Flag) must be set. All nodes on the link will update their Neighbor Caches and store the link-layer address.

Now the HA receives all packets on this link that are destined to the IPv6 address of the mobile node. The HA acts as a proxy for the mobile node. It must inspect all Neighbor Solicitations it receives and verify whether the Target Address field corresponds to a Home Registration entry in its Binding Cache. If so, it replies with a Neighbor Advertisement indicating its own link-layer address as the link-layer address for the mobile node. This procedure also defends the mobile node's home address from other home link nodes trying to configure that same address (i.e., Duplicate Address Detection).

For more details on Neighbor Discovery and the formats of the ND messages and processes mentioned, refer to Chapter 4.

Bidirectional tunneling

To forward packets destined to the home address of the mobile node, the HA uses an IPv6 tunnel. It inserts an additional IPv6 header called the Tunnel header. The Source address in the Tunnel header is the IPv6 address of the HA. The destination address is the primary Care-of address of the mobile node. The mobile node processes the Tunnel header and forwards the decapsulated packet internally to the upper-layer protocols and applications.

In order to receive multicast packets when away from home, the mobile node must register for these group memberships. There are two ways to accomplish this:

- The mobile node can register with local routers on the Home Link using its Care-of address. In this case, it can receive multicast packets directly. These memberships will not survive if the mobile node moves to another foreign network.
- The mobile node can register for multicast group memberships on its Home Link by sending MLD registrations to its HA, which in turn will forward multicast

packets to the mobile node using the tunnel. This will always work no matter how many times the mobile node changes the network.

The following packets are not forwarded to the mobile node:

- Packets sent to the link-layer address of the mobile node. These packets are answered with an ICMPv6 Destination Unreachable message.
- Packets sent to the site-local address of the mobile node. (Site-local addresses have been deprecated after the publication of the Mobile IPv6 specification; this will be removed in a future version of this RFC.)
- Multicast packets sent to a link-local, site-local, or organization-local scope.

Packets sent through the Reverse Tunnel from the mobile node to the HA are decapsulated by the HA and forwarded to their destinations through regular routing mechanisms.

When the HA itself sends data to the mobile node, it behaves like a regular correspondent node, which means it does not use the tunnel, but inserts a Routing header Type 2, which carries the home address of the mobile node.

Mobile Node Operation

As long as the MN is at home, no Mobile IPv6 mechanisms are necessary. If the MN is away from home, it uses its home address as well as its Care-of address. For each communication, it must choose which address to use. Applications and processes above the IP layer usually communicate using the home address of the MN.

If a communication has to survive a move of the MN to another network, the home address must be used. As soon as the MN has a communication with a correspondent node for which there is a Binding, the communication can be routed directly. If there is no Binding, all data will be tunneled through the home agent. For certain and specifically for new communications, the MN can also choose to use its Care-of address without Mobile IPv6 functionality, just as a regular unicast address. When the MN communicates with local nodes in a foreign network—e.g., for Neighbor Discovery—it should communicate directly and not use the Home Address Destination option.

The choice of the best communication path and the corresponding address depends on the requirements of the application, and that is where the decision has to be made. This definition is not part of the Mobile IPv6 specification.

Route Optimization in detail

When a mobile node away from home communicates with a correspondent node for which it has a Binding, it uses the process called Route Optimization.

The MN goes through the following steps: it checks its Binding Update List for an entry of its home address for this correspondent node. This verifies whether the correspondent node can process the Home Address Destination option. Then, it checks the Binding Update List for the following:

- Whether the Source address it wants to use corresponds to the home address in the entry.
- Whether the Destination address it wants to use corresponds to the address of the correspondent node in the entry.
- Whether its current Care-of address is listed as Care-of address in this entry.
- Whether the Binding is valid and the Lifetime is greater than zero.

If all these requirements are met, the mobile node knows that the correspondent node has a valid Binding Cache entry. A packet sent from the MN to this correspondent node contains the following information:

- The Source address field in the IPv6 header contains the Care-of address.
- The packet carries a Destination Options header with a Home Address option containing the home address of the mobile node.

The correspondent node receiving this packet copies the home address from the Destinations Options header into the Source address field of the IPv6 header before processing the packet to upper-layer protocols and applications. To the upper layers and the application, it looks as though the packet was sent from the home address of the mobile node. When the correspondent node wants to send data to an MN, it checks its Binding Cache for an entry for the destination. If there is such an entry, it inserts a Routing header Type 2.

When the correspondent node replies, the addresses are used as follows:

- The Destination address in the IPv6 header contains the Care-of address of the mobile node.
- The packet contains a Routing header Type 2. The address field in the Routing header contains the home address of the mobile node.
- The mobile node exchanges the address in the Destination address field of the IPv6 header with the home address copied from the Routing header, reduces the Segments Left field in the Routing header by one to zero, and processes the packet internally to upper-layer protocols and applications. Again, to the upper layers, it looks as though the packet had been sent to the home address of the mobile node.

Figure 8-5 shows the communication between the MN and the CN, as well as the specific headers associated with Route Optimization.

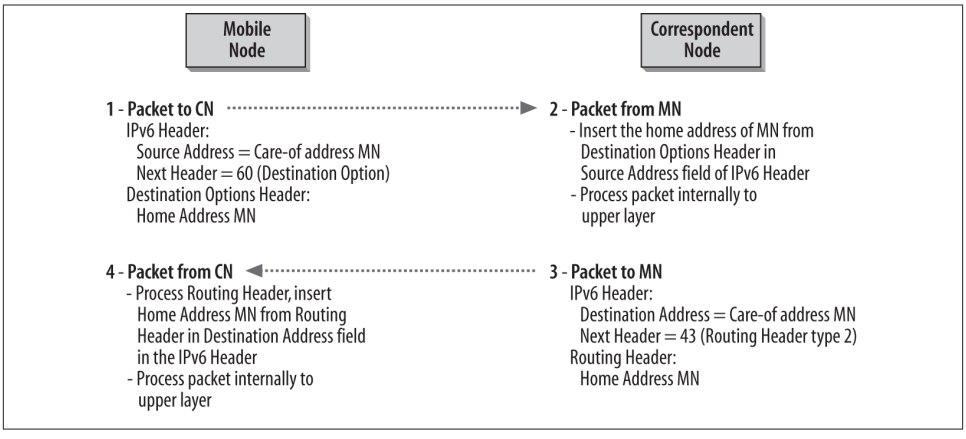

Figure 8-5. Header information with Route Optimization

This figure illustrates the processes described previously. The main goal of Mobile IPv6 is for an MN to keep connectivity to services and applications while moving from one network to another. The goal of Route Optimization is to allow for direct routing between MN and the correspondent node. With the use of Destination options and Routing Type 2 header, both nodes can process the packets internally as though they were in direct communication with the MN on its home link. So to the application, it looks as though the mobile node is on its home link.

This explains why Mobility with IPv6 is much more scalable and well-suited for widespread mobility. The Extension header architecture allows for Route Optimization. Imagine millions of mobile nodes communicating through their home agents to reach their correspondent nodes. The home agent would be a bottleneck, a single point of failure, and the home link unnecessarily overloaded. In many cases, the route from the mobile node to the correspondent node is much shorter than the route going through the home agent.

Communication with bidirectional tunneling

If the MN wants to communicate with a correspondent node for which it does not have a Binding, it uses the Reverse Tunneling mechanism. In this case, the packet is sent through the tunnel via the home agent. The Source address in the original packet carries the home address of the MN and the correspondent node's address as a Destination address. This packet is encapsulated in another IPv6 header carrying the Care-of address of the MN in the Source address field and the IPv6 address of the home agent in the Destination Address field. The home agent processes the first header and forwards the original packet to the correspondent node. Figure 8-6 illustrates the header information.

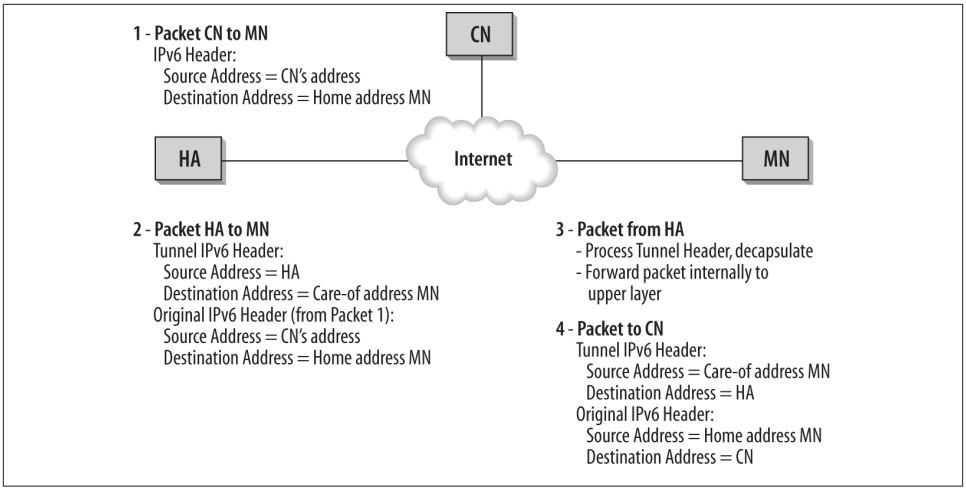

Figure 8-6. Header information with bidirectional tunneling

Movement Detection

How does the MN detect that it has moved to another network? Movement Detection is based on the process of Neighbor Unreachability Detection (NUD; described in Chapter 4). Using NUD, the MN detects when its default router is no longer available. In this case, the MN tries to find a new default router. It performs Duplicate Address Detection (DAD) for its link-local addresses, chooses a new default router based on the Router Advertisements, and builds new Care-of addresses based on the Router prefixes advertised. When the new addresses are initialized, it performs a Binding Update with its home agent first and then with all correspondent nodes for which it has Bindings.

The fact that new routers advertise new prefixes is not necessarily a sign that the MN is in a new network. There may be a new router or a prefix change in the current network. Procedures have to be defined to prevent an MN from unnecessarily updating all Bindings when it has not moved to another network. The following procedures have been defined so far:

- If the MN receives RAs from new routers with new prefixes but its default router is still reachable, it will not perform any Binding Updates. It uses NUD to detect whether its default router is still available.
- RAs can carry an Advertisement Interval option. This allows the MN to detect whether this router is still available based on the comparison of the interval in different RAs.
- If the default router does not reply to a Neighbor Solicitation, the MN should perform a Multicast Router Solicitation.

RFC 5568, "Mobile IPv6 Fast Handovers," describes an update to the Mobile IPv6 specification to reduce the handover latency. New ND messages types are defined for this purpose.

Returning home

When the MN detects that it is back on its home link, it sends a Binding Update to the home agent to inform it that it is back home and that the HA no longer needs to forward packets through the tunnel. This is called a de-registration Binding Update and only after having de-registered can the MN send and receive packets using its home address.

This Home Registration looks as follows:

- The A-Bit (Acknowledge) and the H-Bit (Home Registration) must be set.
- The Lifetime field is set to 0.
- The Care-of address must be the same as the home address.
- The Source address in the IPv6 header must be the home address of the MN.

Security

If the data flow between home agent and mobile node is not secured, there are many possibilities for attacks—for example, man-in-the-middle attacks, hijacking, or Denial of Service attacks.

To secure the tunnel between home agent and mobile node, an IPsec tunnel is configured. IPsec ESP is required for Mobile IPv6 messages. The Mobile IPv6 specification details this.

The following data flows have to be secured:

- Binding Update and Binding Ack between MN and HA
- Home Test Init and Home Test messages sent via HA during the Return Routability Procedure
- ICMPv6 messages between MN and HA used for prefix discovery

All control messages between mobile node and home agent need authentication, integrity, proper sequencing, and anti-replay protection. This protection requires a Security Association (SA) between home agent and mobile node. IPsec does not provide any means to control the sequence of messages. A correct sequence is given by the Sequence Number in Binding Update and Acknowledgment messages. Greater protection from replay attacks can be provided only when Internet Key Exchange (IKE) is used.

For a description of security terms and concepts, refer to Chapter 6.

Binding Updates between the mobile node and correspondent node are protected by the SA established during the Return Routability Procedure. Binding Updates between the mobile node and correspondent node must also be protected by the Binding Authorization Data option. This option includes a Binding Management Key, which is generated during the Return Routability Procedure.

A more detailed discussion of security aspects and mechanisms with Mobile IPv6 can be found in RFC 6275 ("Mobility Support in IPv6"), RFC 3776 ("Using IPsec to Protect Mobile IPv6 Signaling between mobile nodes and home agents"), and RFC 4877 ("Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture"), as well as in general security RFCs.

RFC 4285, "Authentication Protocol for Mobile IPv6," specifies an alternate mechanism to secure MIPv6 messages in 3GPP2 networks. It is an informational RFC not reviewed by the IETF and consists of a MIPv6-specific mobility message authentication option that can be added to MIPv6 signaling messages.

Extensions to Mobile IPv6

A number of extensions have been defined for Mobile IPv6 to make it more flexible and scalable. The following sections describe them.

NEMO

An extension to Mobile IPv6 called Network Mobility (NEMO) has been specified in RFC 3963. The NEMO Basic Support protocol enables mobile networks to attach to different points in the Internet. It allows session continuity for every node in the mobile network even as the Mobile Router changes its point of attachment to the Internet. It also allows every node in the Mobile Network to be reachable while moving around. The solution supports both mobile nodes and hosts that do not support mobility in the Mobile Network.

The processes and messages are basically the same as in Mobile IPv6, except that in this case, the mobile node is a *Mobile Router*. In the current NEMO specification, communication between nodes in the mobile network and correspondent nodes always goes through the home agent. Route Optimization has not been defined yet. Theoretically, nested mobility can be configured where a Mobile Router allows another Mobile Router to attach to its mobile network. This opens ways for many scenarios with high mobility.

One use case could be car communication, where the router in the car could be the Mobile Router. The future will show how we use these technologies, and the developments in the past couple years show that sometimes unexpected applications spring up overnight and are used en masse.

Hierarchical Mobile IPv6

With RFC 5380, "Hierarchical Mobile IPv6," the scalability of Mobile IPv6 is further extended. It is designed to significantly enhance performance of Mobile IPv6 and to reduce the number of messages a mobile node sends over a link in order to update its bindings with the home agent and the correspondent node. It also allows mobile nodes to hide their location from correspondent nodes and home agents when using route optimization.

Hierarchical Mobile IPv6 introduces a new node type, the *Mobility Anchor Point* (MAP). The MAP can be located anywhere in the hierarchical network of routers. It is essentially a local home agent in the geographical region of the mobile node. The mobile node now sends Binding Updates to the local MAP rather than to the home agent and correspondent nodes. By sending one Binding Update message to the MAP, all further traffic from the home agent and the correspondent nodes is rerouted to the new location of the mobile node. The correspondent node and home agent operation are not affected by this and therefore need no changes. Figure 8-7 illustrates the concept.

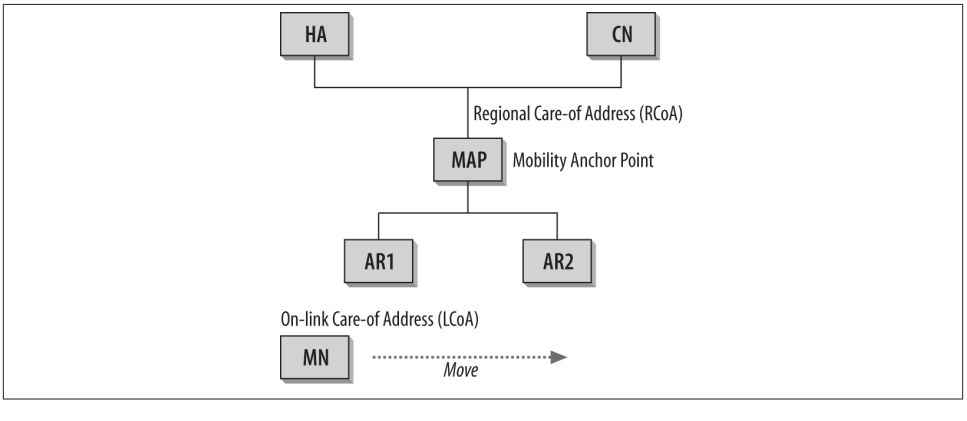

Figure 8-7. Hierarchical Mobile IPv6

When the mobile node enters a MAP domain, it receives Router Advertisements containing information on one or more local MAPs (MAP option). The *Regional Care-of Address* (RCoA) of the MAP corresponds to the Care-of address in the base MIPv6 specification. After registering with a MAP, the mobile node registers its RCoA with its home agent and eventually with its current correspondent nodes. The RCoA is now used as the mobile node's Care-of address. It is the address used by the home agent and

correspondent nodes to communicate with the mobile node away from home. When the mobile node moves from one network to another within a MAP domain, it registers its new *On-link Care-of Address* (LCoA) with the MAP. The MAP, acting like a local home agent, receives all packets on behalf of the mobile node and will encapsulate and forward them to the mobile node's current address. The boundaries of a MAP domain are defined by the Access Routers (AR1 and AR2 in Figure 8-7). This obviously greatly enhances performance of Mobile IPv6, because Binding Updates do not have to be sent to the home agent and the correspondent nodes every time the mobile node moves to another network within the MAP domain.

As mentioned in the section on the Binding Update message, a new bit has been added to the Binding Update message, the M-Bit, which indicates that this is a MAP registration and not a Binding Update with a home agent. There is also an extension to Neighbor Discovery specified to include the MAP's global IPv6 address.

A MAP can exist anywhere in the hierarchical network. Several MAPs can be located within the same domain independently. In addition, overlapping MAP domains are allowed and recommended. Both static and dynamic hierarchies are supported.

Proxy Mobile IPv6

Proxy Mobile IPv6 is a network-based mobility management protocol. It is defined in RFC 5213 and provides mobility on the Network layer, which means that network components take over the handling of mobility messages.

The elements are the *Local Mobility Anchor* (LMA) and the *Mobile Access Gateway* (MAG). The LMA is responsible for managing the reachability of the MN and its home prefixes. The LMA has the function of a topological anchor point of the MN. The MAG manages the mobility of the MN. The MAG is located on the access network of the MN and is responsible for the movements of the MN and for initiating the binding messages with the LMA. There can be multiple LMAs in a Mobile IPv6 domain, of which each can manage different groups of MNs.

Alternatively, localized routing can be used (RFC 6705) to allow nodes attached to the MAGs to directly exchange traffic by using localized forwarding or a direct tunnel between the gateways.

Multiple Care-of Addresses Registration

If an MN has multiple active interfaces, it can only bind one of its Care-of addresses with its home address. The specification in RFC 5648 defines extensions that support the binding of multiple Care-of addresses to the home address. For each binding the MN creates, there is a new *Binding Identification Number* (BID) created and included in the Binding Update. The HA creates a separate binding for each BID und stores it

accordingly in its Binding Cache. The same mechanism is used in binding messages with a CN.

Flow Binding

The Multiple Care-of addresses registration described above allows the binding of multiple CoAs with the Home address. With the Flow Binding Specification in RFC 6089, it is possible to associate different policies to each binding. This is also possible with bindings with CNs or MAPs.

Fast Handover

In RFC 5568, a protocol is specified to reduce the mobile node's handover latency when moving from one network to another. It is called "Fast Handover for Mobile IPv6." The handover operationally consists of movement detection, address configuration, and address update. The combined handover latency is often sufficient to affect real-time applications (e.g., VoIP). This specification describes a set of protocols and procedures to significantly reduce the handover latency. All throughput-sensitive applications can benefit from the Fast Handover.

RFC 4260, "Mobile IPv6 Fast Handovers for 802.11 Networks," discusses and gives some deployment examples for Mobile IPv6 Fast Handovers on 802.11 networks.

Support for Dual-Stack Hosts and Routers

The original Mobile IPv6 and NEMO specification describe the use of Mobility with IPv6 only. RFC 5555 extends this and allows the registration of IPv4 addresses and prefixes and also the transport of IPv4 and IPv6 packets across the tunnel to the HA. The MN can move from an IPv4 network to an IPv6 network and vice versa, even if there is a NAT in the path between the MN and the HA.

Now you have reached the end of the technical chapters in this book. Chapter 9 puts all the pieces together and explains what planning and designing your future network is all about. It integrates more than 10 years of experience, education, and consulting, with many recommendations based on best practices.

References

Here's a list of the most important RFCs and drafts mentioned in this chapter. Sometimes I include additional subject-related RFCs for your personal further study.

RFCs

- RFC 2710, "Multicast Listener Discovery (MLD) for IPv6," 1999
- RFC 3776, "Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents," 2004
- RFC 3963, "Network Mobility (NEMO) Basic Support Protocol," 2005
- RFC 4065, "Instructions for Seamoby and Experimental Mobility Protocol IANA Allocations," 2005
- RFC 4140, "Hierarchical Mobile IPv6," 2005
- RFC 4225, "Mobile IP Version 6 Route Optimization Security Design Background," 2005
- RFC 4260, "Mobile IPv6 Fast Handovers for 802.11 Networks," 2005
- RFC 4282, "The Network Access Identifier," 2005
- RFC 4283, "Mobile Node Identifier Option for Mobile IPv6 (MIPv6)," 2005
- RFC 4285, "Authentication Protocol for Mobile IPv6," 2006
- RFC 4303, "IP Encapsulating Security Payload (ESP)," 2005
- RFC 4306, "The Internet Key Exchange (IKEv2)," 2005
- RFC 4449, "Securing Mobile IPv6 Route Optimization Using a Static Shared Key," 2006
- RFC 4487, "Mobile IPv6 and Firewalls: Problem Statement," 2006
- RFC 4555, "IKEv2 Mobility and Multihoming Protocol (MOBIKE)," 2006
- RFC 4584, "Extensions to Sockets API for Mobile IPv6," 2006
- RFC 4621, "Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol," 2006
- RFC 4831, "Goals for Network-Based Localized Mobility Management (NETLMM)," 2007
- RFC 4866, "Enhanced Route Optimization for Mobile IPv6," 2007
- RFC 4877, "Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture," 2007
- RFC 4885, "Network Mobility Support Terminology," 2007
- RFC 4887, "Network Mobility Home Network Models," 2007
- RFC 4908, "Multi-homing for small scale fixed network Using Mobile IP and NEMO," 2007
- RFC 5094, "Mobile IPv6 Vendor Specific Option," 2007

- RFC 5096, "Mobile IPv6 Experimental Messages," 2007
- RFC 5142, "Mobility Header Home Agent Switch Message," 2008
- RFC 5149, "Service Selection for Mobile IPv6," 2008
- RFC 5164, "Mobility Services Transport: Problem Statement," 2008
- RFC 5213, "Proxy Mobile IPv6," 2008
- RFC 5270, "Mobile IPv6 Fast Handovers over IEEE 802.16e Networks," 2008
- RFC 5271, "Mobile IPv6 Fast Handovers for 3G CDMA Networks," 2008
- RFC 5380, "Hierarchical Mobile IPv6 (HMIPv6) Mobility Management," 2008
- RFC 5555, "Mobile IPv6 Support for Dual Stack Hosts and Routers," 2009
- RFC 5568, "Mobile IPv6 Fast Handovers," 2009
- RFC 5637, "Authentication, Authorization, and Accounting (AAA) Goals for Mobile IPv6," 2009
- RFC 5648, "Multiple Care-of Addresses Registration," 2009
- RFC 5844, "IPv4 Support for Proxy Mobile IPv6," 2010
- RFC 5846, "Binding Revocation for IPv6 Mobility," 2010
- RFC 5847, "Heartbeat Mechanism for Proxy Mobile IPv6," 2010
- RFC 5944, "IP Mobility Support for IPv4," 2010
- RFC 6089, "Flow Bindings in Mobile IPv6 and Network Mobility (NEMO) Basic Support," 2011
- RFC 6275, "Mobility Support in IPv6," 2011
- RFC 6279, "Proxy Mobile IPv6 (PIMPv6) Localized Routing Problem Statement," 2011
- RFC 6463, "Runtime Local Mobility Anchor (LMA) Assignment Support for Proxy Mobile IPv6," 2012
- RFC 6543, "Reserved IPv6 Interface Identifier for Proxy Mobile IPv6," 2012
- RFC 6572, "RADIUS Support for Proxy Mobile IPv6," 2012
- RFC 6610, "DHCP Options for Home Information Discovery in Mobile IPv6 (MIPv6)," 2012
- RFC 6611, "Mobile IPv6 (MIPv6) Bootstrapping for the Integrated Scenario," 2012
- RFC 6618, "Mobile IPv6 Security Framework Using Transport Layer Security for Communication between the Mobile Node and Home Agent," 2012
- RFC 6705, "Localized Routing for Proxy Mobile IPv6," 2012
- RFC 6909, "IPv4 Traffic Offload Selector Option for Proxy Mobile IPv6," 2013
- RFC 7077, "Update Notifications for Proxy Mobile IPv6," 2013

- RFC 7109, "Flow Bindings Initiated by Home Agents for Mobile IPv6," 2014
- RFC 7148, "Prefix Delegation Support for Proxy Mobile IPv6," 2014
- RFC 7156, "Diameter Support for Proxy Mobile IPv6 Localized Routing," 2014
- RFC 7161, "Proxy Mobile IPv6 (PMIPv6) Multicast Handover Optimization by the Subscription Information Acquisition through the LMA (SIAL)," 2014
- RFC 7222, "Quality-of-Service Option for Proxy Mobile IPv6," 2014

CHAPTER 9

Planning for IPv6

After all these chapters describing the features of IPv6 from a technical perspective, this final chapter puts it all together.

It summarizes all I have learned in more than 10 years of studying, working, playing, teaching, and consulting for IPv6. It puts at your disposal all that is relevant to understand what it takes to plan the integration of IPv6. It is a summary of all the answers to the most frequent questions I get when talking with customers. And I hope to make you feel a little enthusiastic about the opportunities that IPv6 offers. So I wish you a good read and lots of fun learning about and planning for IPv6.

While it is true that the integration in a larger network presents some challenges and takes a lot of time and careful planning, there is no need to panic or think that it could not be mastered. Every decent engineer that mastered the introduction of DHCP, VPNs, and NATs will also master the introduction of IPv6. The main difference is in the fact that IPv6 as the transport protocol touches almost every component in the network, so the complexity comes from considering all interactions between the different network components, services, and departments. The challenge is in the organizational aspects, the design of the future architecture, and the interdependencies of all groups and parts of the network and applications.

But the process can be broken down in single doable steps and they do not have to be accomplished in three months if you start early enough. This chapter provides an outline to help you determine how you want to go about it.

When to Choose IPv6?

A golden rule says to "never touch a running system." This rule also applies to your IPv4 networks. As long as they do what they need to do, let them run. But when an IPv4 network hits the limits for some reason, or as soon as an upgrade is due, choose IPv6. IPv6 is mature enough to be used in corporate and commercial networks, as many case

studies and deployments worldwide show. High investments in new IPv4 setups, fixes, or complex configurations for IPv4 (especially NATs) should be avoided if possible because they are investments in a technology that will slowly be phased out and is actually end-of-life. Getting familiar with the new protocol early, taking some time to play with it before you really need it, and planning for it early saves a lot of cost, time, and headaches.

Whatever you invest in IPv6 is an investment in future technology. An investment in IPv4 is an investment in an end-of-life technology.

As already mentioned in Chapter 1, here's the list of indicators that it may be time for you to consider integrating IPv6:

- Your IPv4 network or NAT implementation needs to be fixed or extended.
- A major hardware upgrade or redesign is on the schedule. Redesign for IPv6 and support IPv4 as needed.
- You are running out of address space.
- You want to prepare your network for acquisitions that might suddenly require IPv6 support.
- You want to prepare your network for applications that are based on advanced features of IPv6.
- You need end-to-end security for a large number of users and you do not have the address space, or you struggle with a NAT implementation.
- Your hardware (backbone, DMZ, data center, etc.) or applications reach the end of their life cycle and must be replaced. Make sure you buy products that support IPv6, even if you don't enable it right away. In order to determine the level of support your products need, you need to have an IPv6 strategy.

Integration Scenarios

As the discussion of the different techniques in this chapter shows, there are numerous mechanisms that support a step-by-step introduction of IPv6. There is no one mechanism that can cover all requirements or be optimal for all scenarios. In most cases, a combination of different mechanisms will be chosen. What the best combination and sequence is depends on the infrastructure of the current environment and the goals and requirements for the transition/integration. In the IETF, the work on the basic protocol is completed. They now focus on developing practical scenarios for different types of

environments, and the results are published. We offer a summary here, not with the intent to deliver a cookbook for your environment, but rather to provide food for thought that you can apply to your requirements.

If you want to follow what is going on in the IPv6 operational and maintenance working groups, refer to *http://www.ietf.org/html.charters/v6ops-charter.html* and *http://www.ietf.org/html.charters/6man-charter.html*.

Organizations

To connect a single host or a small network with the IPv6 Internet is not a big challenge and can be done either natively, if your Internet provider offers this, or with one of the tunnel mechanisms described earlier. It is easy to implement with most operating systems.

If you need a tunnel, have a public IPv4 address, and want access to the IPv6 Internet, 6rd or a Tunnel Broker can be used. If you have NAT in place and make use of private IPv4 addresses, you may choose to use Proto 41 Forwarding if the NAT box supports it. Organizations that have the privilege of their providers offering native IPv6 connections can have a dual-stack Internet connection. Dual-stack is in many cases the easiest way to go if your devices and operating systems support IPv6 (and they do if they are on an up-to-date level).

Many organizations have a number of IPv4 Virtual LANs (VLANs). In such situations, an IPv6 router can advertise one single IPv6 prefix into all VLANs that support dual-stack communication. This is only advisable for a transition period, though. All the IPv6 nodes in the VLANs can autoconfigure for an IPv6 address using the prefix advertised by the IPv6 router.

The tunnel mechanisms do not only support the transport of IPv6 over the IPv4 Internet, but also internally over an IPv4 backbone. A backbone upgrade is not something you choose to do every year; you probably want to wait for the end of the backbone router life cycle before touching it. This does not prevent rolling out IPv6 at the edge of the network. As long as the backbone is based on IPv4, IPv6 packets are tunneled to IPv6 islands on the other side.

If you want to integrate IPv6 in a more complex DMZ for your public services and/or in your internal network including multiple locations, possibly multiple data centers, and different types of applications and services used across the whole network, you need to define an enterprise transition strategy. Refer to the section "Planning for IPv6" on page 321 for more details.

RFC 4057, "IPv6 Enterprise Network Scenarios," can assist you in your first steps. It describes different scenarios for IPv6 deployment within enterprise networks and

provides guidance and checklists of how to approach this task. The following scenarios are covered in this RFC:

- Deploy IPv6 in conjunction with IPv4.
- Deploy IPv6 because of a specific set of applications to be used over an IPv6 network.
- Build a new network or restructure an existing network and deploy IPv6 as the predominant protocol within the enterprise in coexistence with IPv4.

The document then reviews a set of network infrastructure components common to most enterprises that must be analyzed.

RFC 4852, "IPv6 Enterprise Network Analysis—IP Layer 3 Focus," is based on RFC 4057 and analyzes the transition to IPv6 in enterprise networks focusing on Layer 3 for the scenarios given in RFC 4057. There is a table with a good overview of different scenarios and then each scenario is discussed in terms of how it could be addressed and what the transition mechanisms could be. Note that this RFC was written in 2007. In some sections mechanisms such as ISATAP, 6to4, and Teredo are mentioned. We strongly recommend that you do not use these in production environments, but replace them with newer technologies.

ISPs

IPv6 is designed to enable Internet Service Providers (ISPs) to meet the challenges associated with the exponential growth of the Internet, allowing them to provide new services to their customers. The number of devices will explode in the coming years, a challenge that can be met only with the address space of IPv6. Cable, DSL, wireless, and other always-on technologies can also benefit from the address space. Other benefits of IPv6 include the capability to enhance end-to-end security and mobile communications, and to ease system management burdens. Some examples include peer-to-peer communication without NAT traversal problems, being able to securely access devices and applications at work from home or vice versa, enhanced IP mobility, and many more.

Therefore, ISPs have to evaluate the capabilities of IPv6 to meet these needs. Some countries have taken a lead role in this area and moved from testing and evaluation to real deployments of IPv6 in the broadband arena. Japan is a prime example, along with other countries that are looking at moving toward large-scale production deployments of IPv6. Since the exhaustion of the IPv4 pool in 2011, many large providers in Europe and the U.S. have also started to deploy IPv6.

Globally, the number of IPv6 transit AS (Autonomous System) has increased substantially. As of 2014, more than 80% of the top 300 transit AS in the world are IPv6 enabled.

ISPs will have to offer both IPv4 and IPv6 services in the coming years. To provide access to IPv6 networks to customers in a first phase (as long as their backbones are still IPv4), tunnel mechanisms can be used. Many large ISPs have chosen to use 6rd, the mechanism that is based on 6to4 but provides more or less native performance (refer to the 6rd section in Chapter 7). This is a simpler and more economical method to start offering IPv6 services. Depending on customer needs and requirements, a native IPv6 deployment option might be more scalable and provide better service performance. You may be able to use the next backbone upgrade cycle and introduce dual-stack. All other services such as web hosting, email, and FTP are best if offered for both protocols (IPv4 and IPv6). The migration steps should be well planned, and a useful combination of mechanisms chosen and implemented. The main goal for an ISP is to offer all of the services over both protocols: this is the only way to cover the whole market. Especially for ISPs, the introduction of IPv6 offers the possibility to create business opportunities and new service offerings. You as a customer want dual-stack if you buy Internet services, because only dual-stack makes you a full member of both Internets (IPv4 and IPv6).

RFC 4029, "Scenarios and Analysis for Introducing IPv6 into ISP Networks," analyzes the challenges and opportunities for ISPs and discusses different integration and transition scenarios, divided into exploring backbone transition actions, customer connection transition actions, and network and service operation actions. RFC 4779, "ISP IPv6 Deployment Scenarios in Broadband Access Networks," presents the options available in deploying IPv6 services in the access part of a broadband Service Provider network, namely cable/HFC, broadband Ethernet, xDSL, WLAN, and PLC/BPL. It briefly discusses the other elements of a provider network as well. It provides different viable IPv6 deployment and integration techniques and models for each of the previously mentioned broadband technology. RFC 5181, "IPv6 Deployment Scenarios in 802.16 Networks," extends the discussion and goes into deployment scenarios for wireless broadband access networks.

Some ISPs shy away from deploying IPv6 and hope they can get away with extending their NATs and implementing CGNs. An IDC study commissioned by Cisco analyzed the capital cost (capex) over 5 years for an ISP with 5 million residential subscribers, comparing a CGN-only scenario with a scenario where 6rd was deployed in parallel to CGN. It shows that the business case for introducing IPv6 in parallel with CGN is much better. The reason is that the native IPv6 offloads traffic from the CGN. With the increasing number of IPv6-capable websites, the 6rd deployment offloads substantial traffic from the CGN, which reduces the maintenance on the CGN side.

The comparison shows that over the period of 5 years an ISP can save up to 69% in capex investment by choosing the right IPv6 strategy. If you are interested in the study, "The Business Case for Delivering IPv6 Service Now," you can find it at *http://bit.ly/1nac2SH*.

Mobile networks

For a long time IPv6 deployment in mobile networks was not happening. Early smartphones such as the once-sexy Sony Ericsson P900 had a Symbian stack with IPv6 support. But none of the mobile providers would deploy IPv6.

This has changed in the last two years. Two of the well-known and often discussed deployments are by T-Mobile U.S. and Verizon Wireless. T-Mobile has several million IPv6 users already in 2014. Other providers, such as China Mobile and also some European providers, have deployed IPv6. With the implementation of 464XLAT in Android 4.3 and higher, it becomes attractive for mobile operators to provide IPv6-only services. Demos have showed that with 464XLAT, IPv4 applications run well on IPv6-only devices. Other smartphone vendors will hopefully jump onto this bandwagon soon.

For more details on 464XLAT, refer to Chapter 7.

RFC 3574 discusses transition scenarios for 3GPP networks. RFC 4215 goes into more details for 3GPP networks and is a supplemental document to RFC 3574. RFC 6459, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)," describes the IPv6 support in different 3GPP architectures, and RFC 7066, "IPv6 for Third Generation Partnership Project (3GPP) Cellular Hosts," describes requirements for cellular devices that go beyond the standard node requirements for IPv6 (RFC 6434), because the characteristics of cellular links in terms of bandwidth, cost, and delay put special requirements on how IPv6 is used. And RFC *draft-ietf-v6ops-mobile-device-profile-07* defines an IPv6 profile that a number of operators recommend in order to connect 3GPP mobile devices to an IPv6-only or dual-stack wireless network (including 3GPP cellular network and IEEE 802.11 network). It defines a different profile than the one for general connection to IPv6 cellular networks defined in RFC 7066. In particular, it also describes features to deliver IPv4 connectivity service over an IPv6-only transport.

Home networks

In the early days of the Internet, a home "network" consisted of one PC with a dial-up modem. The connection had one single IP address. Today this scenario has drastically

changed and with all the different increasing number and types of devices, as well as increased internal routing, challenges increase also. Now we add IPv6 to the mix and this again adds complexity and requires new designs, new addressing principles, and new operational measures.

One big difference between the IPv4 and the IPv6 addressing architecture is that in IPv6 interfaces can and often do have multiple addresses. They have at least a link-local address, but in addition to that they have one or multiple global unicast addresses (GUAs) and possibly even a unique local address (ULA). If a homenet has multiple ISPs, the homenet may get an IPv6 prefix through various channels from each of the ISPs (let's say a 6rd /60 prefix from one ISP and a /48 prefix via DHCP prefix delegation, etc.). So all the devices in the network may end up with two GUAs, one from each prefix. And the home network may typically have multiple subnets.

Several new aspects have to be taken into consideration. There is a homenet working group at IETF, and this group is developing a new architecture for the more complex home networks of the future. The homenet group is defining a general architecture for IPv6-based home networking and describing the associated principles, considerations, and requirements. They discuss specific implications of the introduction of IPv6 for home networking, describe the elements of the architecture, and suggest how standard IPv6 mechanisms and addressing can be employed in home networking. The architecture developed in one of their first documents, "IPv6 Home Networking Architecture Principles," describes the need for specific protocol extensions for certain additional functionality. It is assumed that the IPv6 home network is not actively managed, and runs as an IPv6-only or dual-stack network. If you are an ISP and have to design home networks, the documents of this group are surely a good source.

You can find the homenet group at *http://bit.ly/1nac5Om*.

Planning for IPv6

In February 2011, the global IPv4 address pool was exhausted. Since April 2011, APNIC has been running on the last /8. Since September 2012, RipeNCC has been running on the last /8. This was a wake-up call for the whole industry. Since then many large enterprises started IPv6 projects with different levels of intensity. Many of them are still in the planning phase. As you will see in the coming sections, for a large enterprise the planning and deployment can take as long as three to five years.

Most enterprises have overcome the business case discussion. There are many reasons to do IPv6 and to do it now. By "now" I mean you have to do the planning now. Once

you know what it will take to deploy IPv6, how long it will take in different areas of your network and services, and how you will design your address plan and security concept, you may put these plans aside until the day comes where you need to turn it on. But if you don't plan and wait until that day comes, you will not be able to create decent and operationally useful designs under time pressure.

Planning and designing the integration of IPv6 in your network provides a unique opportunity to establish a new foundation for your next-generation IT environment.

IPv6 offers the opportunity to clean the existing inconsistencies, drive standardization, and implement new architectures and operational models.

The most common reasons for enterprises that started their IPv6 projects are:

- Business continuity
- Reachability (from outside)
- Life cycle management
- Investment protection
- Time for education and to build experience

What are the advantages of planning early? There are many areas where you can save a lot of money with early planning and make good use of the opportunities that an IPv6 integration offers. So here's a list of advantages:

- Make use of product life cycles, refresh cycles, and other IT projects.
- Investment protection by having clear IPv6 requirements for purchasing of new equipment, tools, and applications, and for negotiating outsourcing contracts and SLAs. All these must be adjusted to cover IPv6 requirements appropriately.
- Integrating IPv6 will take up to three or even more years (mostly in order to make use of refresh cycles). If you don't plan early, you won't be ready when you need it. Especially the design and concept for addressing and security take a lot of time and should be done carefully. They need to be updated to take advantage of the new addressing architecture and usually require a thorough discussion and multiple iterations until all IT departments can identify with it. Usually the security people are challenged most and need time to learn about IPv6 and adjust their concepts. But they also have a big opportunity to clean out and redesign the IPv4-based security concept.
- You need time to educate all IPv6 team members and IT staff.

- You want to use all the opportunities IPv6 offers!
- You need sufficient time for labs, testing, and pilots.
- You need time for bug fixing with vendors (early stacks).

When you run a larger or even global network, most IT decision makers wonder where to start and how to go about it.

The fact that the introduction of IPv6 touches each and every aspect and device in your network makes this an IT project of a larger scale than any other IT project.

So let me try to give you a quick start into planning for IPv6.

A more detailed discussion of the planning process can be found in my companion book, *Planning for IPv6* (O'Reilly).

Where to Start

One of the main success factors is that the IPv6 project needs management attention and an enterprise IPv6 project manager. All devices, services, and aspects of your network are affected and it is important to have an overall picture and be aware of the interfaces and dependencies (both organizational and technical).

The first steps:

- Get management attention.

Appoint an IPv6 project manager and establish an IPv6 transition office.

- Educate all groups participating in the process.
- Get an overall perspective.
- Involve all teams.

- Include business vision, business strategy, and IT strategy, and base the IPv6 strategy on this foundation to make it sustainable.
- Your IPv6 strategy is supposed to support the business.
- Develop a communication and internal marketing strategy for the IPv6 project.

The following steps are recommended:

Create a layout of your current situation
: In the first step analyze your business vision, your business strategy, and your IT strategy as well as your IT infrastructure, IT services and projects, life cycles of hard- and software, tools, and also contracts and service agreements. Create an overview of processes, policies, and standards, and the impact an IPv6 integration will have on all of these aspects.

Define your IPv6 strategy
: In this second step define your target architecture, your high-level IPv6 strategy, including a high-level address plan and a high-level security concept (which must be aligned with each other). Next define a roadmap, aligning it with other IT projects and refresh cycles. Get a clear picture of the adoption plan, milestones, and dependencies. Based on your IPv6 design, create *RFC requirements* for all types of devices and applications. Assess your *vendor strategy* and find out if it needs adjustments for IPv6. Don't assume that the vendors who were great for IPv4 are automatically the best for IPv6 too.

Assess current environment
: Based on the RFC requirements, assess your current infrastructure, hard- and software, operating systems, applications, services, and contracts for their IPv6 readiness. Only now will you be able to determine the cost of deploying IPv6, based on the information from the assessment. It may also be possible that you have to adjust your design or roadmap, or both, based on the findings in the assessment. Use this opportunity to build IPv6 experience in your team.

Create detailed concepts and deployment plans
: For all the different areas in your network, detailed design and deployment plans have to be created and tested in labs and pilots. All these processes may need several iterations. Enough time has to be allocated for testing, bug fixing, and vendor evaluation. Also, don't forget to adjust all processes, such as testing of upgrades, all levels of support, change management, security processes, monitoring, address management, etc.

Deploy and document
: As mentioned, the deployment of IPv6 in an enterprise network can take up to three or even five years. You will not deploy in all areas at once; you will have a roadmap and deploy in specific areas aligned with other projects or life cycles. So the last

steps, such as detailed design, deployment plans, and finally deployment, are usually executed for specific areas. Finally, make sure that all is well documented.

A Word on Applications

Enabling IPv6 in the network is doable. But as long as applications don't run over IPv6, it is not of much use having an IPv6-enabled network. So you need to start working on applications early in the process while enabling the network.

Third-party applications should usually be IPv6-enabled in the current release, and if not, IPv6 support should be on a clear roadmap. Talk to your vendor, and if necessary evaluate alternatives. It is a different story for self-developed applications, though. Here you need to differentiate between two cases: a) the application is used in-house only, and b) the application is a market application. In case a) you have control over how users are utilizing the applications. Applications developed for the market should be running in IPv4 networks, in IPv6 networks, and also in dual-stack networks, as the vendor has no control over how users are accessing the application.

If you develop applications for the market, it is important to start planning early, as often these applications have very long life cycles. If the developers don't understand the impacts of IPv6, the applications may either not run at all, or not be optimized for an IPv6 network. So whether you develop applications in-house or with a partner, ensure adequate IPv6 education of the developer team with focus on how to develop for a dual-stack network and update the application design specification to support the future IPv6 network. If you buy custom-developed applications on the market, you have to update your RFC requirements specifically.

This section is not a guide to porting applications. The goal here is to make you aware of situations that you will face and what to look out for.

You will encounter the following situations:

- IPv4 applications on dual-stack nodes
- IPv6 applications on dual-stack nodes
- Applications supporting IPv4 and IPv6 on dual-stack nodes
- Applications supporting IPv4 and IPv6 on IPv4-only nodes
- Applications supporting IPv4 and IPv6 on IPv6-only nodes

The challenge for developers that develop for the market is to create applications that work well in all situations. DNS names should be used whenever a service has to be called. But the DNS reply is not a reliable indicator of which protocol to use. For example, a host may be dual-stack and have an A record with an IPv4 address and a AAAA record with an IPv6 address in DNS. But on this host, there may be an IPv4-only application. So even though resolving the host name returns an IPv6 address, the application is not

reachable over IPv6. So you need to enter services names in DNS, with corresponding record types (A records for IPv4 services and AAAA records for IPv6 services) in DNS. But how this is handled by DNS administrators depends on operational practice and is therefore not reliable. A DNS request can also return a AAAA record, but the host may have no IPv6 path to get there. A node resolving a DNS name and getting multiple addresses in the reply should try them all and have a mechanism to choose the connection with the best performance.

Happy Eyeballs is one mechanism designed to help in this situation and is implemented in different operating systems and browsers. Refer to the DNS section in Chapter 5 for more information.

The following list shows the most important IP dependencies in applications:

- Format of the IP address (32-bit dotted decimal or 128-bit hexadecimal with colons)
- API functions for the establishment of connections and data exchange
- DNS, resolving host names to IP addresses and vice versa
- IP address selection, caching/storage of addresses
- Multicast applications, depending on situation; correspondence of IPv4 and IPv6 multicast addresses and selection of correct socket configuration options
- Some applications base licensing control on IPv4 addresses

The best way to go is to make applications independent of the IP version. This means that the source code should not have any IP dependencies. The communication library should provide APIs that have no dependencies on IP.

Find a more detailed discussion of these issues in RFC 4038, "Application Aspects of IPv6 Transition."

The University of Tokyo and the Yokogawa Electric Corporation started the TAHI project in 1998. They developed tests that can be used by IPv6 developers to test their implementations for conformance to the standard and for interoperability. The tests can be used at no cost. This is their contribution to an efficient development and deployment of IPv6. The results of the tests are also provided to the developer community at no cost. On the TAHI website, all tests are listed and documented. The TAHI project has been concluded by December 2012, because they believe it has accomplished

the mission of "supporting IPv6 developers by quality side for a highly interoperable IPv6 world." The information and testing tools are still there (at the time of writing).

The TAHI project worked closely with other well-known projects: the WIDE project, the KAME project, which has been concluded also (but the dancing Kame is still there), and the concluded USAGI project. Find some interesting links and some history on the early evolution of IPv6.

Do's and Don'ts

Based on many years of consulting with large enterprises, here are some viewpoints and perspectives I keep seeing that I would like to summarize for you. Some of them are successful and others belong to the category of what you should not be doing. In any case, they try to give you guidelines for how to go about your project.

Is IPv6 just like IPv4?

Typically, people that have not started on IPv6 yet have one of two viewpoints. The first group thinks IPv6 is a no-brainer; after all, it is not that different from IPv4 and it will be easy to adapt their IPv4 concepts and make it fly. Based on that, they often decide that they need no external help and their internal people can do it, or they choose consultants that have a good IPv4 background but no IPv6 experience and assume they will do the job. The other group thinks that IPv6 is a killer, way too complex, and hard to master, and that they will never manage to integrate it properly. So they shy away from the project and try to avoid it as long as possible, hoping the problem will solve itself.

Both viewpoints are extreme and the truth is somewhere in the middle. The first group would do well to at least have their IPv6 strategy, address plan, and security concept reviewed by someone with good IPv6 experience. It is like getting a second opinion from a doctor before undergoing a critical treatment. It is much easier to fix a few things before deployment than to try to change it after or have to live with it for many years. And if the review confirms all is fine, that is a good background, too. The second group is only partially right. Although IPv6 is an evolution of IPv4 and many things are the same, architecturally there are many advantages and new features, such as the address architecture. And it is a good advice to not shy away but take the time for thorough education of all teams before they have to dive into designs of the network, the address plan, and the security concept. If you wait too long and have to do all this under time pressure, you will not have the time to take advantage of the opportunities of an IPv6 integration.

Inescapable bugs and why generic assessments are not very useful

IPv6 stacks are much younger than most IPv4 stacks. While we are still sometimes fixing bugs in IPv4 stacks or finding security holes, we can expect that to be way more common

in IPv6 stacks simply because we don't have many years of operational experience and years of testing and bug fixing behind us. This leads to two things to be aware of.

One is that you need to allocate enough time in your planning for bug fixing with vendors. As soon as you start your labs and pilots, you will see those bugs and the vendor needs some time to work on a solution. But the other thing I have heard in many IPv6 strategy discussions is that people say, "oh, we don't use this technology; it's not working, we tested it in our lab."

It is not a good idea to base your design and target architecture on bugs in current stacks. Design your network and applications as they will best fit your business and your needs, and expect your vendors to fix the bugs.

Closely related to this is the procedure where organizations do a generic assessment of the IPv6 capability of their current environment (such as "can I enable IPv6 or not?") even before they have an IPv6 strategy. This means that they will base their design on availability of features, instead of having a clear strategy that supports the business and then making sure they have the equipment and applications that support that strategy. If you want a target architecture that supports your business, you will not get around defining a strategy first, creating detailed RFC requirements, and then using these requirements in your purchasing requirements.

Vendor strategy and RFC requirements

As already mentioned, if you have vendors that deliver great IPv4 services (be it products, Internet access, data center hosting, outsourcing of other services, or consulting), don't assume that they are best in IPv6 also. You need to verify this in more detail. Some customers have chosen to take the IPv6 integration as a good point to re-evaluate their vendor strategy for the coming years.

The following points should be taken into consideration:

- Your vendors face the same challenges as you do. Only they should be ahead of the game, but the reality is that many of them are not. So don't assume they are; check it out!
- Some customers think that their vendors will know what to do and that they can trust the vendor's recommendations. Not true in most cases. You will face the fact that your purchasing team (which is probably not too technical and not up to speed with IPv6) will sit at the table with sales guys from vendors that barely know how to spell IPv6. So unless you come up with very specific and informed ideas and RFC requirements, you cannot expect to get what you need.

- In the early stages of your planning project, write letters of intent to your strategic vendors, making them aware of the fact that you will require professional IPv6 services in the foreseeable future. The more such letters vendors get, the faster they will speed up with implementation. After all, their major excuse for lack of implementation or richness of features is that customers don't ask for IPv6.
- Don't forget to assess outsourcing contracts and SLAs for IPv6 consideration and requirements.

When evaluating vendors, you should look for the following:

- Check technical features according to RFC list. Get a list of implemented RFCs from the vendor, check it against your requirements, and then test in your lab whether the features that are critical to your plans work as expected. And remember, RFC requirements have to be device specific. An IPAM solution has different requirements than a firewall.
- Check the level of IPv6 knowledge in their staff. Do they have one single person who knows how to do IPv6? If yes, their support will probably not be sufficient for you as a customer. Ensure that they have enough knowledgeable staff in all channels (sales, engineering, support).
- Ensure that they also have IPv6 implemented in all their processes (upgrade processes for consistency of future updates, incident management, etc.).
- Don't trust brochures and fact sheets. Test in your lab.

Without these steps you may end up buying products that do not support your IPv6 strategy or you may become stuck in your deployment, because you suddenly find out that key features you need are not supported. And as you probably know from other IT projects, this can cost a lot of time and money, especially if critical business services or other IT projects are depending on this.

To build your RFC requirements lists, use RFC 6434, "IPv6 Node Requirements." In addition to that, there are two useful templates available: RIPE554 and USGv6 by NIST. Take it as a foundation and adapt it to your IPv6 strategy.

There are other documents that may be useful, such as RFC 7084, "Basic Requirements for IPv6 Customer Edge Routers." In April 2014, a discussion was started to update RIPE554. So when you start working with these documents, make sure you use the latest version.

General Design Guidelines

Every environment and business is different and the IPv6 strategy will therefore also look different. It is not possible to provide a cookbook that is applicable in every organization. But still, there are some very general strategies that make sense in most environments, so here's a list:

- Do native IPv6 wherever possible, dual-stack as long as necessary (and this may be many years).
- Deploy new internal services IPv6-only whenever possible.
- Use tunneling only if necessary and only as a temporary solution. Sometimes it is recommendable to wait a little longer and go native IPv6 right from the beginning.
- If you do tunnel, prefer stateless over stateful.
- Don't do NAT, no translation (only with a gun to your head).
- If you have to translate, again, prefer stateless over stateful.
- Future networks are end-to-end.
- The expanded address architecture allows for new security concepts (you may consider embedding service information in the address to simplify access rules and operation).
- Adapt and align your security concept to the new address architecture.
- Consider new services (monitoring, sensors, health care, Car2Car…depending on industry). Many new services have a much higher demand for addresses and mobility requirements.

More and more customers are considering migrating their infrastructure to IPv6-only as soon as possible and treat IPv4 as a service. The main advantage of this IPv6-centric strategy is to minimize cost. Operating an IPv6-only network is less costly than maintaining two protocols in parallel in a dual-stack network.

Address Plan

A good address plan is the cornerstone of a successful IPv6 deployment and can substantially help ease network operation and troubleshooting. But the address plan is a challenge for most. We have all been well trained to conserve addresses and the training has been so efficient that this sits in our body cells. We have to overcome this reflex action before we can create a useful IPv6 address concept for 128-bit addresses. So what are the rules; how can address concepts be designed for IPv6? These are the topics in this section to provide some guidance.

Before we start, I would like to recall something from Chapter 2 on addressing that may help you slowly overcome your conservation reflex and free your mind from limited IPv4 thinking. Use this as your daily mantra while you work on an address concept:

In Chapter 2, I showed that usually an ISP gets a /32 allocation for its customers and that one single /32 is actually more than the whole IPv4 address space (because it has 32 bits to define subnets and each /64 subnet has 64 more bits for interface IDs). Up to 2013, we have globally allocated approximately 130,000 /32s, which means 130,000 times more address space than the whole IPv4 Internet. Even though this sounds astronomically big, this amounts to only 0.025% of the currently available public IPv6 address pool (`2000::/3`). So let us be a bit generous in designing IPv6 address plans.

To quote Vint Cerf:

> The vast IPv6 address space opens up serious opportunities for the re-examination of the notion of "address." The IETF has only allocated 1/8th of the IPv6 address space for current use. The remaining 7/8 of the address space is still to be allocated. In consequence we may be able to interpret new segments of the IP address space in ways that are different from topological end points. This is precisely the reason that a focus on the future of IPv6 is so important at this point in the evolution of the Internet.

Okay, and now shake your cells and read on. By the way, most of this address plan section is taken from my companion book, *Planning for IPv6* (O'Reilly). I repeat it here because people always ask many questions about IPv6 address plans. So I decided to cover address plans in a section of this book as well. For more details on planning, please refer to the companion book.

This section assumes you are familiar with the technical aspects of the IPv6 address architecture described in Chapter 2.

Practice shows that in many cases customers choose to group address ranges logically, for the following purposes:

- Allow for prefix aggregation.
- Simplify the configuration, operation, and processing of access lists and firewall rules.
- Make addresses traceable, to contain information about the use type or location where the address is used.
- Make it scalable; create enough space for extensions (more services) and growth (more locations and users).
- Allow for efficient network management and ease of operation.

What is new?

There are three main differences to consider when designing an IPv6 address plan:

- One of the most obvious differences (besides the length of the address) is that we don't need variable length subnet masks (VLSMs) anymore. The prefix or network part of an IPv6 address should always be 64 bits (/64). Even for point-to-point links? Yes, even for point-to-point links (some people choose to change that rule—you will have to decide for yourself). So, in this respect, IPv6 addresses are simpler. No more guessing or misconfiguring of subnet masks. VLSMs have contributed a lot to the complexity of managing IPv4 address space.
- The preceding rule also relates to the following: we no longer have to size subnets to the number of devices in the subnet (in other words, no more address conservation practices are necessary). Each and every subnet can potentially have 2^{64} devices. The number of devices for each subnet will be determined by your switch's capacity, so consult your vendor to find out what is possible.
- In IPv6, addresses are assigned to interfaces. So an IPv6 address identifies an interface, not a host. And, in many cases, an interface will be multi-addressed and choose addresses to initiate connections based on policies or default address selection rules.

Please resist the temptation to use larger masks for your subnets, such as /96, in order to preserve address space. While the standard does not prohibit that, observing the /64 boundary makes your address plan future proof. Changing this mask may create many problems in the future, as all development is based on the /64 mask. For instance, SLAAC (Stateless Address Autoconfiguration) won't work with non /64 masks. There may be many other processes or services that fail with a nonstandard subnet mask. You have enough address space with IPv6! Optimize it for efficient management.

Once you are familiar with the IPv6 address architecture, take your list of things that you wanted to change in your IPv4 address plan if you could start all over, and then you begin to design your IPv6 address concept.

The ingredients are:

- Review your current IPv4 address plan and the operational lessons learned.
- Identify the constraints and inconsistencies in your current IPv4 plan.
- Identify your needs according to your target architecture.
- Create different address plan scenarios to address these items.
- Have internal reviews and discussions with all IT teams.
- Take the best options to the lab, including your provisioning tools, and test the common operational scenarios.

When you define your scenarios, you will have to find a good balance between the following two choices:

Focus on route aggregation
: Emphasizing the route aggregation by assigning the higher-order bits to the topology (regions, sites) reduces the number of routes in the core routing tables.

Focus on policy aggregation
: Emphasizing the policy aggregation by assigning the higher-order bits for service- or security-related information (services, zones) reduces the number of policies, minimizes the risk of user errors, and simplifies management.

To generalize, prioritize the most common and most important information in the high-order bits and allocate enough space for current and envisioned options. They can be new sites, new services, new users, etc.

In order to build a useful address plan, consider the following:

- Prefix aggregation based on your topology.
- Policy aggregation based on your services and security concepts.
- Subnet consistency (don't include address conservation mechanisms; focus on operational ease).
- Use of address types (only global addresses or global plus unique local).
- Consistently assign a /64 to subnets and point-to-point links; no need for host planning.
- If you have many locations and they have local Internet access, it is recommended to assign at least a /48 to each of them.
- If you are a global organization, consider regional address allocations (RIPE, APNIC, ARIN, etc.) for risk mitigation purposes.
- Observe nibble boundaries whenever possible (a nibble is 4 binary digits).
- Use of provisioning mechanisms and tools (DHCPv6, SLAAC, IPAM/DDI).
- Don't create structure for the sake of structure; only truly relevant information should be embedded.
- Avoid direct mapping of your IPv4 address plan, as it perpetuates legacy issues and limits the ability to leverage the IPv6 address space resources.
- Security aspects.
- Growth (network, users, services).
- New services with increased address demand (monitoring, sensors, management, etc.).

Your upper limit for address planning is the range that you have received (a /48 or a /32 depending on the size of your company). Your lower limit is the /64 that you have to assign to your subnets. So within that space you can define your own structure. You start with identifying your current address inventory and your clearly envisioned needs for the foreseeable future, and then you take the bits that are left free and use them to support growth.

Don't forget to define clear processes and rules for allocation of IPv6 address space.

Figure 9-1 shows an example of how such a high-level scenario might look.

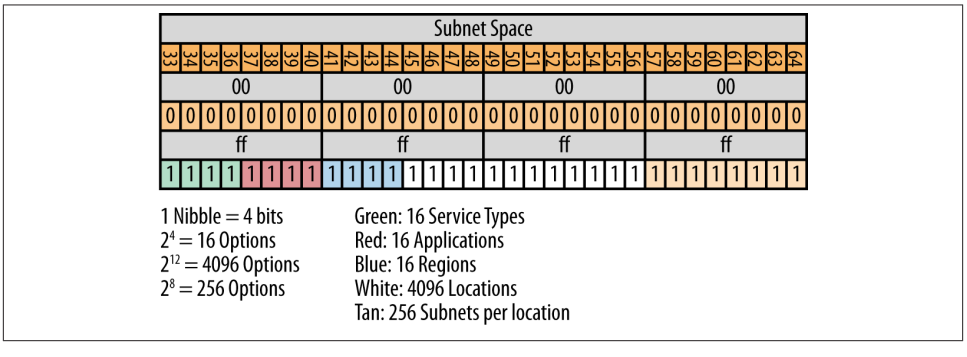

Figure 9-1. Example of a high-level address plan

In this example, a combination of service-based and topology-based design was chosen. The first nibble (the first 4 bits) was chosen to differentiate 16 service types that can then be used to assign special rules on a high level. The next nibble is used to differentiate 16 applications per service type. The third nibble is used for 16 super regions, and per super region, 4,096 locations are possible (12 bits). Each location then gets a /56 prefix and can still create 256 subnets. Whether this is a good address plan totally depends on the network, the service catalogue, and the security concept of the organization. It is simply an example to get you thinking. The main goal of the service-level approach is to make security policy design and operation simpler.

In all aspects you need to integrate the incredibly large IPv6 address space. This can't be repeated enough. So, for instance, when it comes to subnet consistency, there is no need to conserve address space.

For operational ease and simple administration, keep subnets for a certain type of location all the same size, regardless of how many users work there. While this is the best way to go, it may feel uncomfortable in the beginning. The same rule is true if you leave address space free for growth. A good rule to follow is: if your cells don't hurt, your plan isn't quite big enough yet.

Global addresses versus ULAs

As already mentioned, an IPv6 interface can typically have multiple addresses. Part of your address design is deciding what address types to use. Specifically, you have two options: to generally use global IPv6 addresses (GUAs), or to use unique local addresses (ULAs) internally. ULAs are defined in RFC 4193 and have a prefix of `fd00::/8`. They are similar to RFC 1918 private addresses in IPv4, which means they are routable but should only be used internally and never routed to the Internet. They can, for instance, be used for an internal deployment or to build a lab when you do not have global IPv6 address space allocated yet.

There is one big difference between the RFC 1918 concept and how these private ULA addresses are used today. Using ULAs does not mean that you need NAT (network address translation) to get outside to the Internet. Because IPv6 interfaces are designed to work with multiple address types, interfaces that need Internet connectivity will simply have a global IPv6 address in addition to the ULA address. When connecting to an internal server, the interface will use its ULA address; and when connecting to the Internet, it will use its global IPv6 address. RFC 6724 defines default address selection rules to deal with this. In larger networks you will have your own policies about source and destination address selection rules. So, with this design, internal server systems and databases that should not be reached from outside will be configured only with ULAs. If you want a ULA-only addressing internally, you could also use NPTv6 (Network Prefix Translation) described in the NAT section in Chapter 7.

There is no need to combine ULA addresses with NAT as we did with private addresses in IPv4. In IPv4 the NAT was needed to map many addresses into one or a few. In IPv6, hosts that need access to internal services and the Internet simply get two addresses, a ULA and a GUA (global address).

When it comes to choosing whether you want to use ULAs or go with global addresses, there are pros and cons to both approaches and many discussions surrounding them. Due to early deployment time, there is not too much operational experience available to draw from. From a technical point of view, you could make either choice—you will have to decide for yourself.

Here are some of the considerations:

- ULAs make you internally independent of your global prefix, which is advantageous if you need to renumber your network. In this case, all your internal systems and communications are not affected. If you have a PI (provider-independent) address space, this is not a concern.
- ULAs add a layer of security to your internal infrastructure, as servers and databases with critical data are not reachable from outside. Obviously, you still need firewalls to protect systems, but ULAs offer additional protection. Some organizations achieve that same goal by setting aside a part of their global prefix and use it as internal-only prefix, blocking it at the border in both directions.
- Attackers who know some of your global addresses cannot derive the addresses for internal systems.
- ULAs should not be used with NAT, so NAT is not a concern. Systems that need access to the Internet can get a global IPv6 address in addition to the ULA address. That is the advantage of the IPv6 multiple-addresses-per-interface architecture.

Now, whether you want to use ULAs is up to you. They may be useful in isolated domains such as manufacturing lines or inter-datacenter traffic for backup. Many people decide that with the unlimited IPv6 address space, where they can finally address all systems with global addresses, they want to go for this simple option—securing their internal systems with firewalls and having the advantage of administering only one prefix. For those who decide to use both global and ULA, I would hope that IPAM vendors are clever enough to develop systems that can manage multiple prefixes and import the subnet design to the IPAM solution for easy administration. You can probably choose the same addressing plan for both prefixes anyway. And your third option is that you can choose to use a specific prefix from your global prefix and assign it to internal systems only and block that prefix to and from the outside world. Even in the case of using ULAs, you will have to specifically block them at the border because they are not to be routed to the Internet.

General considerations

Whenever you translate something from your IPv4 address plan to your IPv6 address plan, make sure that you are not copying limitations over to the new concept. You are creating an address concept with a future native IPv6 network in mind, and the dual-stack network is only a transitional state—although it may exist for some years. But once you start running a native IPv6-only network, you want the almost unlimited freedom of the address space without having to renumber the network.

For a long time, the recommendation was that ISPs get a /32 from their RIR, and organizations and end sites get a /48 from their provider or a /48 as PI space. RFC 6177 changes the recommended assignment size to end sites. It states that "a one-size-fits-all recommendation of /48 is not nuanced enough for the broad range of end sites and is

no longer recommended as a single default." The RFC states that it is in the domain of the operational community to determine the best prefix size for end sites. This introduces some new considerations. When the default prefix was a /48, a change of provider or assignments from different providers always had the same prefix boundary. With the new policy, it is possible that an end site may have to renumber from a larger prefix into a smaller prefix, which means having to collapse subnets. You can prepare for this if you have a /48 and use only the low-order bits first (if the network size allows you to do that). It seems that many providers often assign /56 or /52 prefixes to smaller sites. This rule does not apply to large enterprises; they will always get a /48 or even more. According to RFC 6177, even home users should get more than just a /64 subnet. The development of services in the market suggests that future home networks have multiple subnets (smart buildings).

With regard to subnet size, there is nothing in the specification that prevents you from changing the /64 boundary. But I don't recommend that, because doing so will break many other features of IPv6 where applications and processes assume the /64 subnet size. This includes mechanisms such as Neighbor Discovery (ND), SEcure Neighbor Discovery (SEND), privacy extensions, parts of Mobile IPv6 specifications, protocol-independent multicast sparse mode (PIM-SM) with embedded RP, and site multihoming by IPv6 intermediation (SHIM6), among others. And there may be more developments in the future that assume a /64 boundary. If you change that, all these features will break.

The only benefit of creating prefixes longer than /64 would be address conservation. Use your mantra and remember that we don't need to conserve space anymore. The benefit of conserving space would be very small compared to the pain of managing a nonstandard subnet prefix.

Nevertheless, you can still choose to use high-order bits from the interface ID part to denote specific systems, services, or networks to make it easier to recognize these IDs.

A good and helpful practice when grouping bits is to always do it on a 4-bit boundary (a *nibble*). This way, it is much easier to decipher the address because 4 bits represent one hexadecimal digit.

Let's play with some examples to get our brains going. You could group subnets by location and then by service or use type, or the other way round (by service type first and then by location).

So, for instance, with the prefix `2001:db8:1::/48`, you have 16 bits for subnetting. You can choose to subnet as follows (L=location, S=service, F=free):

```
Location first:       2001:db8:1:LLLLSSSS FFFFFFFF::/64
Service type first:   2001:db8:1:SSSSLLLL FFFFFFFF::/64
```

Which option you choose depends on whether your main purpose is to optimize routing, in which case you choose the location identifier first, or whether you want to optimize security rules and ACLs (which are usually based on filters for specific services), in which case you would choose the service identifier first. In some cases, organizations only assign a subnet to a location, and the location further manages the prefix and creates its own address plan. In such a case, you would also choose the location identifier first.

Or you may want to mix these two options. You may have some specific services that you want to filter on at a high level, so you would put identifiers for these services in the highest-order bits of your subnet range, perhaps followed by location bits to optimize the routing within your network, and ending with bits to identify more services.

With the 4 bits used for location and service type in this example, you can create 16 locations and services (2^4). You have to adapt this scheme to the number of locations and services you want represented (and add enough room for growth). Sometimes organizations include a location or service description in their VLAN numbering plans. This could also be reflected in your IPv6 addressing plan to include the 12-bit VLAN ID in the prefix. You can include it in decimal notation (leaving out A–F) or convert it to hexadecimal notation.

These are all general ideas for input on your creative planning. Take the time to carefully craft your future addressing design, discuss all possible options again and again, have your address concept proposal reviewed by many people—including external consultants to challenge your plans—and go over it repeatedly until it feels right. The address design is the foundation for your next-generation network. If you don't get it right and have to change it during the deployment project, this will add costs and delay the project. In the long term, if you live with a suboptimal address design, returning operational costs may be unnecessarily high.

Configuration of interface IDs

How interface IDs are defined depends on the IP address management process that you choose. You can either choose *Stateless Address Autoconfiguration* (SLAAC), where interfaces create interface IDs based on MAC address (EUI-64 format defined in RFC 4291), or with the *Privacy Option* (as defined in RFC 4941), where the interface ID is randomly generated and changed regularly. In newer operating systems, Microsoft uses a stable interface ID created with a random identifier. You can use DHCPv6 and assign addresses just as you do in IPv4. With DHCPv6, the service may have options to define interface IDs in different ways. To best secure your hosts, you may not want to start at a low interface ID and sequentially number your interfaces. Instead, you may want to have random interface IDs, because the more random and distributed your interface IDs are, the harder it is to scan them.

If you have a global IPv6 prefix and use a ULA prefix internally, you may want to use the privacy option for the global prefix. This way, users accessing the Internet cannot

easily be tracked because their interface ID changes regularly. For the ULA prefix (or for the internal part of your global prefix filtered at the border), it's preferable to have fixed addresses assigned to make management, troubleshooting, security, and logging easier. Again, this also depends on whether vendors will provide IP address management solutions that help to deal with this.

There is a recommendation to not use hardware-based interface IDs. Microsoft already uses a stable interface ID that is created with a random identifier. RFC 7217 defines a stable interface ID not based on a hardware identifier. Refer to the section "Stateless Address Autoconfiguration (SLAAC)" on page 102 for more information.

Where Do You Get Your Address Space From?

The IANA (Internet Assigned Numbers Authority) is responsible for the global coordination of the IP addressing systems, as well as the ASN (Autonomous System Numbers) used for routing Internet traffic.

RIRs (Regional Internet Registries) get their IP address space from IANA. There are several RIRs in the world to allocate address space within their region. Figure 9-2 shows the global hierarchy, the RIRs and the regions they serve.

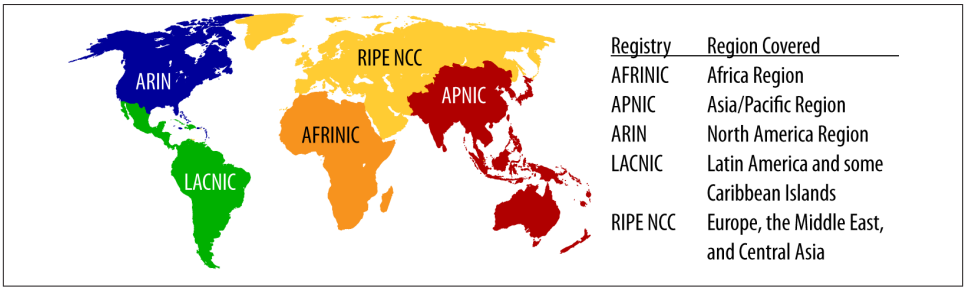

Figure 9-2. The Regional Internet Registries

The RIRs further serve LIRs (Local Internet Registries), NIRs (National Internet Registries), ISPs (Internet Service Providers), and in some cases also end customers with address space.

The address policies in the different regions differ slightly from each other. ISPs and customers must get the information about the policies in their region from their RIR. Each RIR has a website with all the information on address space and policies.

Links to all the RIRs can be found on IANA's website.

Customers have several options to apply for IPv6 address space.

Currently there is *PA space* (*provider aggregatable*) and *PI space* (*provider independent*). PA space is targeted at ISPs, providing Internet connectivity to many end customers. In order to receive PA space you must usually be a member of the RIR and pay an annual membership fee. Standard allocation sizes are in the range of /29 to /32. Providers then assign networks to their customers, usually in the range of /48 or /56. PA space handed out to customers by an ISP cannot be taken to another ISP, so in case of switching ISPs, renumbering is needed (or the use of NPTv6 prefix translation). For assignment recommendations, refer to RFC 6177, which is described in the section below.

PI space is intended for end users who are not able to or do not want to use addresses from their provider's PA space. This does not require membership but an annual fee per prefix. PI space belongs to the end user and can be taken along to the next ISP. Every single PI prefix used on the Internet shows up in the global routing table, impacting all routers worldwide. Originally, in the early days of IPv6, the plan was to not assign PI space in order to keep routing tables small. But it has shown that this policy cannot be sustained in practice.

So as an end customer you have the following choices:

- PA space from a local provider
- PA space from the RIR (requires membership)
- PI space from RIR (annual fee)

You can find the official information from RIPE about their IPv6 Address Allocation and Assignment Policy.

Again, this document number is valid in March 2014. At a later time you may want to check if there is a newer version.

How Much Space Will You Get?

Allocation rules are a work in progress and can still change. The latest update of how to use the address space is in RFC 6177, "IPv6 Address Assignment to End Sites." It

contains recommendations on how the address space should be further divided. It softens the rules defined in the previous version, which was RFC 3177, where home networks as well as small and large enterprises should receive a /48. As mentioned before, with the new version these rules have been adjusted, because the exact assignment of prefix sizes should be in the authority of the operational community.

The RFC further states: "An important goal in IPv6 is to significantly change the default and minimal end site assignment, from 'a single address' to 'multiple networks' and to ensure that end sites can easily obtain address space."

If you are responsible for allocating addresses or are requesting addresses for your network, then the following rules for address allocation in RFC 6177 can be used:

- A key principle for address management is that end sites must always be able to obtain a reasonable amount of address space for their actual and planned usage, and over ranges of time specified in years rather than just months. In practice, that means at least one /64, and in most cases significantly more. One particular situation that must be avoided is having an end site feel compelled to use IPv6-to-IPv6 Network Address Translation or other burdensome address conservation techniques because it could not get sufficient address space.
- It should be easy for an end site to obtain address space to number multiple subnets (i.e., a block larger than a single /64) and to support reasonable growth projections over long time periods (e.g., a decade or more).
- The default assignment size should take into consideration the likelihood that an end site will have need for multiple subnets in the future and avoid the IPv4 practice of having frequent and continual justification for obtaining small amounts of additional space.
- Although a /64 can (in theory) address a practically unlimited number of devices, sites should be given sufficient address space to be able to lay out subnets as appropriate, and not be forced to use address conservation techniques such as using bridging.
- Assigning a longer prefix to an end site, compared with the existing prefixes the end site already has assigned to it, is likely to increase operational costs and complexity for the end site, with insufficient benefit to anyone.
- The operational considerations of managing and delegating the reverse DNS tree under ip6.arpa on nibble versus non-nibble boundaries should be given adequate consideration.
- Home network subscribers should receive multiple subnets.

A site that receives a /48 prefix has 16 bits for subnetting, which allows for 65,536 subnets (default IPv6 subnet size of /64). In special cases, a very large enterprise can request a shorter prefix.

Your IPv6 address plan is the cornerstone of your future network and should support growth for many years and be sustainable and not fragmented. So there is no way around sitting down and doing your homework, finding out how much space you really need. And then go and request it.

I have spoken with many customers from all different sizes, SMEs up to large enterprises. And I keep hearing statements such as "Oh, this is what we got [some size of prefix]. We will see how we can live with it." Please don't do that. At the moment you may think this is more than you previously had with IPv4, so you can live with it somehow. So again, please don't do that; you (or the ones that will have to operate your network in the future) may be very unhappy about it and pay a lot for this in terms of headaches and operational disadvantages.

By defining an address plan as described in this section you will find out how much you really need. There are standard assignments that you can get easily without too much documentation. But you don't know if that is sufficient to support your target architecture. If you do your homework and come up with a sustainable and good plan and find out that you need more, you will get it; it just requires a little more work. But it is a little more work in the beginning. If you avoid that and base your address plan on too small an allocation, you will pay for it in the future. Because redesigning and redoing it will take a lot more effort than getting it right from the beginning.

Multihoming with IPv6

Multihoming is when a host or a site is reachable over different IP addresses. A multihomed host has multiple global IP addresses. These addresses can come from one or more different providers, and they can be assigned to one or more different interfaces on the host. A multihomed site is connected to the Internet with multiple global IP addresses from one or different providers.

The main reasons to configure multihoming are the following:

Redundancy
: When a link fails, the connection can be maintained over the alternative link(s).

Load balancing
: Provides more throughput because traffic is balanced over two or more links.

Cost
: It may be desirable to have multiple providers—for instance, because one provider may have a better offering for certain types of services.

The autoconfiguration features of IPv6 support an easier maintenance of multihoming scenarios because devices are more flexible in recognizing network prefixes and can configure multiple IPv6 addresses based on Router Advertisements.

In the common IPv4 multihoming approach, a site's local prefixes are announced as distinct routing prefixes into the interdomain routing system and propagated to the top-level hierarchy of the routing system. This approach works well if the address space of the site is provider independent. But even though this approach covers most of the requirements for multihoming, it is not scalable.

For these reasons, multihoming is an actively discussed topic in the working groups. The goal is to find solutions that provide multihoming without the scalability and transport issues. One idea would be to separate the identification of a node from the location. An IP address currently contains both information, the prefix identifying the location and the IID identifying the node. One approach to this would be LISP, discussed in Chapter 7. While we wait and hope that an elegant multihoming architecture will be found, currently multihoming in IPv6 is done the same way as in IPv4.

Cost of Introduction

Some people believe that the introduction of IPv6 is way too expensive, and they don't even start to think about it. Other people just want to know what it will cost. The next section contains aspects to consider and some thoughts on the business case for IPv6.

Gartner says that the integration of IPv6 costs approximately 6% of an annual IT budget divided by the number of years for the migration. In several cases where we had a closer look at this from a high-level perspective, the estimation was pretty accurate. The ongoing cost after completed migration is estimated at approximately 1% of the IT budget.

Hardware and Operating Systems

When planning ahead for IPv6 and making use of refresh cycles, the cost for hardware upgrades is not extensive. In many cases, IPv6 is not part of the hardware and can be installed as part of the operating system or as a software upgrade. Most operating systems and applications ship with IPv6 included at no extra cost. If IP functions are implemented purely in hardware or if a system's software cannot be upgraded, the hardware must be replaced to support IPv6. If you plan ahead, though, this can usually be accomplished with the next upgrade in the regular life cycle of the product and therefore generates no additional cost. As our case studies have shown, this is confirmed by organizations that have deployed IPv6.

Software

We covered IPv6 in application development in the section "A Word on Applications" on page 325. Nowadays, many common applications already support IPv6 or will do so in their next major upgrade version. In this case, you can limit the cost for software by planning early and using the regular life cycles of applications. For proprietary and self-developed applications, the situation has to be analyzed individually.

A simple porting of an application makes sure it runs equally well with IPv6 transport. A creative porting of an application may include using the advanced features of IPv6 and thereby extending the flexibility and functionality of the application. This can even be seen as a cost associated with introducing advanced next generation services based on new technologies more than as cost of introducing IPv6.

Education

Every technology upgrade requires education: for developers, vendors, service providers, and the infrastructure and systems operators in organizations. For home users, it should be their ISP's job to make the transition simple.

A well-planned education program according to job responsibilities is essential and supports a smooth introduction of IPv6 substantially. For system operators, the time and effort needed for the learning is not higher than for maintaining the IPv4 infrastructure. We are used to integrating new technologies all the time to keep our networks state-of-the-art. We had to introduce DHCP, NAT, and VPNs in the past, and we mastered them. Some of us have worked in dual-stack networks mastering the coexistence of IPX (Novell) and TCP/IP. Now we are going to introduce IPv6 and the challenge is no bigger. And it is worth effort, time, and money because in the long term, the maintenance of an IPv6 network will be less costly than the maintenance of an IPv4 network. People that have a good understanding of IP and have mastered the previously mentioned technologies will have no problems mastering IPv6.

It is well worth it to take enough time to learn about IPv6 before making detailed designs and before going into production. There are new concepts and possibilities in IPv6, and we need some time to become familiar with these, learn how to best make use of them, and then integrate the things we've learned into the planning. If you start doing your labs and hands-on tests early as recommended, you will build your understanding and experience on the job.

When I talk to organizations who have designed and deployed IPv6 and I ask, "what is the biggest risk?", the answer often is "lack of education." Early and thorough education lets you take advantage of the opportunities IPv6 offers.

Planning

The most important aspect of integration is the planning. The same rule applies here. Extending an infrastructure to keep it state-of-the-art always requires planning and should be seen as an investment. Instead of planning the next extension of IPv4, we will now plan the integration of IPv6.

The planning requires network architects and systems engineers with a good understanding of IPv4 and networking concepts. The first thing they need is a thorough IPv6 education and an IPv6 playground. The planning should not focus on how to make the same services available over IPv6. It should include understanding the new features of IPv6 and making use of them to create new concepts of architecture, security, mobility, and administration. Especially for IPv6 addressing concepts and IPv6 security concepts, an organization is well advised to take a lot of time, as with these two ingredients you are laying the foundation for your network for many years. And having a good address and security design will save a lot of cost and headaches. Here actually lies a lot of the opportunities of an IPv6 integration.

As practice shows, it takes some time to overcome IPv4 thinking and learn to leverage the advantages of IPv6.

The introduction of IPv6 is smoother and cheaper the earlier you plan. A step-by-step integration is the best way to go, as it gives you time to integrate what you learn as you go. Step-by-step integration is possible thanks to the many available transition mechanisms.

Other Costs

Probably the highest costs arise when you wait too long. The longer you wait, the more investments you make into maintaining and extending your IPv4 infrastructure. This is money and effort you invest in an end-of-life technology. You may need to build band aids for IPv4 (e.g., NAT) that later complicate the introduction of IPv6.

There are no reasons to tear down a performing network that fulfills all the requirements, but as soon as you have to invest in fixing or extending your IPv4 infrastructure, you should stop for a moment and consider IPv6 as an alternative. As mentioned earlier, putting IPv6 as an evaluation criterion on all your IT shopping lists for products that have more than two years' lifetime is a good idea. You may not want to turn IPv6 on tomorrow, but you may want to do so next year. If your equipment and software are ready, you can do it without additional cost when the right moment comes. When you deploy new services in the future, consider deploying them IPv6-only right from the

beginning. Why go through all the cost of testing and making them work over IPv4 if you will have to migrate them later anyway? This obviously requires a dual-stacked infrastructure.

Another cost factor can be if a business-critical application comes out that is based on the advanced functionality of IPv6. If your infrastructure is IPv6-ready, you can introduce that application with moderate cost. If at the same time you have to master the transition to IPv6, this might create significant cost and put your current infrastructure at risk (plus create some headaches and sleepless nights).

You have reached the end of this book. Now you know about the essentials of the IPv6 specification and have some guidelines on the transition mechanisms and the planning process. What you need next is to gather experience, by building labs and tests, and by using it. Have fun!

References

Here's a list of the most important RFCs and drafts mentioned in this chapter. Sometimes I include additional subject-related RFCs for your personal further study.

RFCs

- RFC 2185, "Routing Aspects of IPv6 Transition," 1997
- RFC 2529, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels," 1999
- RFC 2663, "IP Network Address Translator (NAT) Terminology and Considerations," 1999
- RFC 3022, "Traditional IP Network Address Translator (Traditional NAT)," 2001
- RFC 3053, "IPv6 Tunnel Broker," 2001
- RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds," 2001
- RFC 3493, "Basic Socket Interface Extensions for IPv6," 2003
- RFC 3542, "Advanced Sockets Application Program Interface (API) for IPv6," 2003
- RFC 3582, "Goals for IPv6 Site-Multihoming Architectures," 2003
- RFC 3715, "IPsec-Network Address Translation (NAT) Compatibility Requirements," 2004
- RFC 3789, "Introduction to the Survey of IPv4 Addresses in Currently Deployed IETF Standards Track and Experimental Documents," 2004
- RFC 3790, "Survey of IPv4 Addresses in Currently Deployed IETF Internet Area Standards Track and Experimental Documents," 2004

- RFC 3791, "Survey of IPv4 Addresses in Currently Deployed IETF Routing Area Standards Track and Experimental Documents," 2004
- RFC 3792, "Survey of IPv4 Addresses in Currently Deployed IETF Security Area Standards Track and Experimental Documents," 2004
- RFC 3793, "Survey of IPv4 Addresses in Currently Deployed IETF Sub-IP Area Standards Track and Experimental Documents," 2004
- RFC 3794, "Survey of IPv4 Addresses in Currently Deployed IETF Transport Area Standards Track and Experimental Documents," 2004
- RFC 3795, "Survey of IPv4 Addresses in Currently Deployed IETF Application Area Standards Track and Experimental Documents," 2004
- RFC 3796, "Survey of IPv4 Addresses in Currently Deployed IETF Operations & Management Area Standards Track and Experimental Documents," 2004
- RFC 3901, "DNS IPv6 Transport Operational Guidelines," 2004
- RFC 4029, "Scenarios and Analysis for Introducing IPv6 into ISP Networks," 2005
- RFC 4038, "Application Aspects of IPv6 Transition," 2005
- RFC 4057, "IPv6 Enterprise Network Scenarios," 2005
- RFC 4177, "Architectural Approaches to Multi-homing for IPv6," 2005
- RFC 4192, "Procedures for Renumbering an IPv6 Network without a Flag Day," 2005
- RFC 4213, "Basic Transition Mechanisms for IPv6 Hosts and Routers," 2005
- RFC 4215, "Analysis on IPv6 Transition in Third Generation Partnership Project (3GPP) Networks," 2005
- RFC 4218, "Threats Relating to IPv6 Multihoming Solutions," 2005
- RFC 4219, "Things Multihoming in IPv6 (MULTI6) Developers Should Think About," 2005
- RFC 4241, "A Model of IPv6/IPv4 Dual Stack Internet Access Service," 2005
- RFC 4472, "Operational Considerations and issues with IPv6 DNS," 2006
- RFC 4554, "Use of VLANs for IPv4-IPv6 Coexistence in Enterprise Networks," 2006
- RFC 4659, "BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN," 2006
- RFC 4779, "ISP IPv6 Deployment Scenarios in Broadband Access Networks," 2007
- RFC 4787, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP," 2007
- RFC 4798, "Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)," 2007

- RFC 4852, "IPv6 Enterprise Network Analysis—IP Layer 3 Focus," 2007
- RFC 5181, "IPv6 Deployment Scenarios in 802.16 Networks," 2008
- RFC 5375, "IPv6 Unicast Address Assignment Considerations," 2008
- RFC 5569, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)," 2010
- RFC 5902, "IAB Thoughts on IPv6 Network Address Translation," 2010
- RFC 6036, "Emerging Service Provider Scenarios for IPv6 Deployment," 2010
- RFC 6052, "IPv6 Addressing of IPv4/IPv6 Translators," 2010
- RFC 6144, "Framework for IPv4/IPv6 Translation," 2011
- RFC 6146, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers," 2011
- RFC 6147, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers," 2011
- RFC 6164, "Using 127-Bit IPv6 Prefixes on Inter-Router Links," 2011
- RFC 6177, "IPv6 Address Assignment to End Sites," 2011
- RFC 6180, "Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment," 2011
- RFC 6250, "Evolution of the IP Model," 2011
- RFC 6269, "Issues with IP address sharing," 2011
- RFC 6296, "IPv6-to-IPv6 Network Prefix Translation," 2011
- RFC 6302, "Logging Recommendations for Internet-Facing Servers," 2011
- RFC 6434, "IPv6 Node Requirements," 2011
- RFC 6459, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)," 2012
- RFC 6540, "IPv6 Support Required for All IP-Capable Nodes," 2012
- RFC 6586, "Experiences from an IPv6-only Network," 2012
- RFC 6724, "Default Address Selection for Internet Protocol Version 6 (IPv6)," 2012
- RFC 6866, "Problem Statement for Renumbering IPv6 Hosts with Static Addresses in Enterprise Networks," 2013
- RFC 6879, "IPv6 Enterprise Network Renumbering Scenarios, Considerations, and Methods," 2013
- RFC 6883, "IPv6 Guidance for Internet Content Providers and Application Service Providers," 2013
- RFC 6921, "Design Considerations for Faster-Than-Light (FTL) Communication," April 1, 2013

- RFC 7021, "Assessing the Impact of Carrier-Grade NAT on Network Applications," 2013
- RFC 7050, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis," 2013
- RFC 7059, "A Comparison of IPv6-over-IPv4 Tunnel Mechanisms," 2013
- RFC 7066, "IPv6 for Third Generation Partnership Project (3GPP) Cellular Hosts," 2013
- RFC 7084, "Basic Requirements for IPv6 Customer Edge Routers," 2013
- RFC 7094, "Architectural Considerations of IP Anycast," 2014
- RFC 7098, "Using the IPv6 Flow Label for Load Balancing in Server Farms," 2013
- RFC 7123, "Security Implications of IPv6 on IPv4 Networks," 2014
- RFC 7136, "Significance of IPv6 Interface Identifiers," 2014
- RFC 7157, "IPv6 Multihoming without Network Address Translation," 2014
- RFC 7168, "The Hyper Text Coffee Pot Control Protocol for Tea Efflux Appliances (HTCPCP-TEA)," April 1, 2014
- RFC 7217, "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)," 2014

Drafts

Drafts can be found at *http://www.ietf.org/ID.html*. To locate the latest version of a draft, refer to *https://datatracker.ietf.org/public/pidtracker.cgi*. You can enter the draft name without a version number and the most current version will come up. If a draft does not show up, it was possibly deleted. If it was published as an RFC, the RFC number will be displayed. *http://tools.ietf.org/wg* is also a very useful site. More information on the process of standardization, RFCs, and drafts can be found in the Appendix A.

Here's a list of drafts I refer to in this chapter, as well as interesting drafts that relate to the topics in this chapter:

"Recommendations of Using Unique Local Addresses"
draft-ietf-v6ops-ula-usage-recommendations-02

"Mapping of Address and Port with Encapsulation (MAP)"
draft-ietf-softwire-map-10

"Mapping of Address and Port using Translation (MAP-T)"
draft-ietf-softwire-map-t-05

"DHCPv6 Options for Configuration of Softwire Address and Port Mapped Clients"
draft-ietf-softwire-map-dhcp-07

"Lightweight 4over6: An Extension to the DS-Lite Architecture"
draft-ietf-softwire-lw4over6-08

"IPv4 Residual Deployment via IPv6—a Stateless Solution (4rd)"
draft-ietf-softwire-4rd-08

"An Internet Protocol Version 6 (IPv6) Profile for 3GPP Mobile Devices"
draft-ietf-v6ops-mobile-device-profile-07

"NAT64 Deployment Options and Experience"
draft-ietf-v6ops-nat64-experience-10

"IPv6 Operational Guidelines for Datacenters"
draft-ietf-v6ops-dc-ipv6-01

"DHCPv6/SLAAC Interaction Operational Guidance"
draft-liu-v6ops-dhcpv6-slaac-guidance-01

"IPv6 Home Networking Architecture Principles"
draft-ietf-homenet-arch-13

APPENDIX A
RFCs

If you want to learn more about IPv6 or any other standardized protocol, you need to read RFCs. They are the most accurate source of information—but yes, I do agree, not always fun reading (except if you read the ones published on April 1). This appendix provides a short overview of the standards and the RFC process. It also includes a list of IPv6-relevant RFCs mentioned throughout this book.

General RFC Information

The Internet Engineering Task Force (IETF) and the Internet Engineering Steering Group (IESG) are the organizations that define the official specification documents of the Internet Protocol suite. These documents are recorded and published as standards track Request for Comment (RFC). If you want to understand the role of the IETF and the standardization process, if you need a list of all the organizations involved in the process and a description of what they do, or if you wish to attend an IETF meeting, there is an interesting and humorous RFC that describes the background, processes, and rules: RFC 4677, titled "The Tao of IETF—A Novice's Guide to the Internet Engineering Task Force."

RFCs are written reports describing most of the information regarding TCP/IP and the architecture, protocols, and history of the Internet. There are many sites on the Internet where RFCs are electronically accessible. The sites are very different, but most of them support some form of search mechanism. Find the site that best suits your preferences.

A good starting point is *http://www.rfc-editor.org*. There used to be a tribute to Jon Postel, one of the fathers of the Internet, who died in October 1998. He was *the* RFC editor. Besides this information, there is also an overview of the RFC series and process.

On the search and retrieve page of this site, there are many ways to access the wealth of information. RFCs can be viewed by number or in an index; they can be in forward or

reverse chronological order; and they can be searched by author, title, number, or keyword. Of course, there is also a link to alternative RFC repositories.

RFC 2555 is an interesting overview of 30 years of RFC history and a good description of the contribution of Jon Postel's services to the Internet community. There is even more information about Jon Postel at *http://www.postel.org/postel.html*.

The first RFC, RFC 0001, was published by Steve Crocker on April 7, 1969. Today, the number of RFCs continues to rise quickly and has exceeded 7,000. RFCs can have different statuses, such as standard, informational, experimental, and historic. A good overview of the different statuses and current level of standardization can be found at *http://www.rfc-editor.org*.

At one time, the RFC Editor published snapshots of the "Internet Official Protocol Standards." These documents were known as xx00 documents, the last of which was published in May 2008. These snapshots have been replaced by a web page, so the RFC Editor will no longer be publishing these snapshots as RFCs. As a result, the RFC Editor will classify unpublished RFC xx00 numbers through 7000 as "never issued." Starting with the RFC number 7100, xx00 numbers will be available for assignment. This is published in RFC 7101, "List of Internet Official Protocol Standards: Replaced by a Web Page."

Drafts

I refer to drafts often throughout this book. Drafts always have version numbers in their names, and this number often increases frequently during the process. So let us have a closer look at the draft process.

During the standardization process, the drafts are published on the Internet. Drafts may eventually become RFCs. At the IETF website, click on Internet Drafts to find all documents. To use the search engine and find the most updated status of any document, refer to *https://datatracker.ietf.org/public/pidtracker.cgi*. You can also click on "IETF Working Group" to find all groups sorted by areas (application, operation, routing, security, etc.). Within the groups, you find all relevant RFCs and drafts. This is the best place to find out what is in the queue and what groups are working on. The goal of this process is to make a specification under development accessible to a large audience in order to get reviews and comments. Another good link to get an overview of RFCs and active drafts with regard to specific working groups is *http://tools.ietf.org*.

So let us understand the draft version numbers. As you have noted, it can change frequently. When you enter a draft name in the search engine, it will always show you the latest version. The rules for draft numbers are as follows.

Every time a draft is updated, it receives a new version number. At some point, a draft may be published as an RFC and then removed from the draft directory. A draft with a specific version number has a lifetime of six months at a maximum. After this, if it has not been updated or become an RFC, it is removed from the draft directory. As this book is going to be on the market for some time, some drafts mentioned in here might not be active when you try to find them. They may have been removed or published as RFCs. RFC 7221, "Handling of Internet-Drafts by IETF Working Groups," is the latest update on drafts, the documents, and the processes.

Drafts are not standards and should not be implemented in commercial products, because they are going to change for sure in the case they ever become an RFC. This can also lead to incompatibility issues, such as when vendors implement drafts at different maturity levels of development or if one implementation is based on draft and another on RFC. In practice, you will find draft implementations in commercial products, but now you know to be careful when using them.

RFC Index for IPv6

This is a list of all RFCs relevant to IPv6 and RFCs regarding related technologies published as of April 2014. It is sorted by RFC number.

- RFC 1, "Host Software," 1969
- RFC 318, "Telnet Protocol," 1972
- RFC 791, "Internet Protocol," 1981
- RFC 854, "TELNET PROTOCOL SPECIFICATION," 1983
- RFC 855, "TELNET OPTION SPECIFICATIONS," 1983
- RFC 959, "FILE TRANSFER PROTOCOL (FTP)," 1985
- RFC 1058, "Routing Information Protocol," 1988
- RFC 1191, "Path MTU Discovery," 1991
- RFC 1195, "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments," 1990
- RFC 1321, "The MD5 Message Digest Algorithm," 1992
- RFC 1546, "Host Anycasting Service," 1993
- RFC 1700, "Assigned Numbers," 1994
- RFC 1707, "CATNIP: Common Architecture for the Internet," 1994
- RFC 1710, "Simple Internet Protocol Plus White Paper," 1994
- RFC 1745, "BGP4/IDRP for IP—OSPF Interaction," 1994
- RFC 1752, "The Recommendation for the IP Next Generation Protocol," 1995
- RFC 1771, "A Border Gateway Protocol 4 (BGP-4)," 1995

- RFC 1793, "Extending OSPF to Support Demand Circuits," 1995
- RFC 1812, "Requirements for IP Version 4 Routers," 1995
- RFC 1819, "Internet Stream Protocol Version 2," 1995
- RFC 1828, "IP Authentication using Keyed MD5," 1995
- RFC 1829, "The ESP DES-CBC Transform," 1995
- RFC 1883, "Internet Protocol, Version 6 (IPv6) Specification," 1995
- RFC 1918, "Address Allocation for Private Internets," 1996
- RFC 1981, "Path MTU Discovery for IP version 6," 1996
- RFC 1997, "BGP Communities Attribute," 1996
- RFC 2003, "IP Encapsulation within IP" (October, 1996)
- RFC 2080, "RIPng" for IPv6," 1997
- RFC 2085, "HMAC-MD5 IP Authentication with Replay Prevention," 1997
- RFC 2101, "IPv4 Address Behaviour Today," 1997
- RFC 2104, "HMAC: Keyed-Hashing for Message Authentication," 1997
- RFC 2136, "Dynamic Updates in the Domain Name System," 1997
- RFC 2149, "Multicast Server Architectures for MARS-based ATM multicasting," 1997
- RFC 2185, "Routing Aspects Of IPv6 Transition," 1997
- RFC 2205, "Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification," 1997
- RFC 2207, "RSVP Extensions for IPSEC Data Flows," 1997
- RFC 2210, "The Use of RSVP with IETF Integrated Services," 1997
- RFC 2233, "The Interfaces Group MIB using SMIv2," 1997
- RFC 2235, "Hobbes' Internet Timeline," 1997
- RFC 2324, "Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)," 1998
- RFC 2328, "OSPF Version 2," 1998
- RFC 2362, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," 1998
- RFC 2365, "Administratively Scoped IP Multicast," 1998
- RFC 2375, "IPv6 Multicast Address Assignments," 1998
- RFC 2403, "The Use of HMAC-MD5-96 within ESP and AH," 1998
- RFC 2404, "The Use of HMAC-SHA-1-96 within ESP and AH," 1998
- RFC 2405, "The ESP DES-CBC Cipher Algorithm With Explicit IV," 1998

- RFC 2410, "The NULL Encryption Algorithm and Its Use With IPsec," 1998
- RFC 2412, "The OAKLEY Key Determination Protocol," 1998
- RFC 2428, "FTP Extensions for IPv6 and NATs," 1998
- RFC 2430, "A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)," 1998
- RFC 2450, "Proposed TLA and NLA Assignment Rules," 1998
- RFC 2451, "The ESP CBC-Mode Cipher Algorithms," 1998
- RFC 2453, "RIP Version 2," 1998
- RFC 2460, "Internet Protocol, Version 6 (IPv6) Specification," 1998
- RFC 2464, "Transmission of IPv6 Packets over Ethernet Networks," 1998
- RFC 2467, "Transmission of IPv6 Packets over FDDI Networks," 1998
- RFC 2465, "Management Information Base for IP Version 6: Textual Conventions and General Group," 1998
- RFC 2466, "Management Information Base for IP Version 6: ICMPv6 Group," 1998
- RFC 2471, "IPv6 Testing Address Allocation" (6Bone), 1998
- RFC 2473, "Generic Packet Tunneling in IPv6 Specification," 1998
- RFC 2474, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," 1998
- RFC 2475, "An Architecture for Differentiated Services," 1998
- RFC 2491, "IPv6 over Non-Broadcast Multiple Access (NBMA) networks," 1999
- RFC 2492, "IPv6 over ATM Networks," January, 1999
- RFC 2507, "IP Header Compression," 1999
- RFC 2526, "Reserved IPv6 Subnet Anycast Addresses," 1999
- RFC 2529, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels," 1999
- RFC 2545, "Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing," 1999
- RFC 2555, "30 Years of RFCs," 1999
- RFC 2590, "Transmission of IPv6 Packets over Frame Relay Networks Specification," 1999
- RFC 2597, "Assured Forwarding PHB Group," 1999
- RFC 2608, "Service Location Protocol, Version 2," 1999
- RFC 2663, "IP Network Address Translator (NAT) Terminology and Considerations," 1999

- RFC 2675, "IPv6 Jumbograms," 1999
- RFC 2710, "Multicast Listener Discovery (MLD) for IPv6," 1999
- RFC 2711, "IPv6 Router Alert Option," 1999
- RFC 2715, "Interoperability Rules for Multicast Routing Protocols," 1999
- RFC 2784, "Generic Routing Encapsulation (GRE)," 2000
- RFC 2845, "Secret Key Transaction Authentication for DNS (TSIG)," 2000
- RFC 2884, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks," 2000
- RFC 2894, "Router Renumbering for IPv6," 2000
- RFC 2914, "Congestion Control Principles," 2000
- RFC 2921, "6BONE pTLA and pNLA Formats (pTLA)," 2000
- RFC 2925, "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations," 2000
- RFC 2928, "Initial IPv6 Sub-TLA ID Assignments," 2000
- RFC 2963, "A Rate Adaptive Shaper for Differentiated Services," 2000
- RFC 2983, "Differentiated Services and Tunnels," 2000
- RFC 2993, "Architectural Implications of NAT," 2000
- RFC 2998, "A Framework for Integrated Services Operation over Diffserv Networks," 2000
- RFC 3006, "Integrated Services in the Presence of Compressible Flows," 2000
- RFC 3007, "Secure Domain Name System (DNS) Dynamic Update," 2000
- RFC 3019, "IP Version 6 Management Information Base for The Multicast Listener Discovery Protocol," 2001
- RFC 3022, "Traditional IP Network Address Translator (Traditional NAT)," 2001
- RFC 3027, "Protocol Complications with the IP Network Address Translator," 2001
- RFC 3041, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," 2001
- RFC 3053, "IPv6 Tunnel Broker," 2001
- RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds," 2001
- RFC 3068, "An Anycast Prefix for 6to4 Relay Routers," 2001
- RFC 3086, "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification," 2001
- RFC 3101, "The OSPF NSSA Option," 2003
- RFC 3107, "Carrying Label Information in BGP-4," 2001

- RFC 3111, "Service Location Protocol Modifications for IPv6," 2001
- RFC 3118, "Authentication for DHCP Messages," 2001
- RFC 3122, "Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification," 2001
- RFC 3124, "The Congestion Manager," 2001
- RFC 3140, "Per Hop Behavior Identification Codes," 2001
- RFC 3142, "An IPv6-to-IPv4 Transport Relay Translator," 2001
- RFC 3146, "Transmission of IPv6 Packets over IEEE 1394 Networks," 2001
- RFC 3162, "RADIUS and IPv6," 2001
- RFC 3168, "The Addition of Explicit Congestion Notification (ECN) to IP," 2001
- RFC 3175, "Aggregation of RSVP for IPv4 and IPv6 Reservations," 2001
- RFC 3178, "IPv6 Multihoming Support at Site Exit Routers," 2001
- RFC 3226, "DNSSEC and IPv6 A6 aware server/resolver message size requirements," 2001
- RFC 3232, "Assigned Numbers: RFC 1700 is Replaced by an On-line Database," 2002
- RFC 3246, "An Expedited Forwarding PHB," 2002
- RFC 3247, "Supplemental Information for the New Definition of the EF PHB (Expedited Forwarding Per-Hop Behavior)," 2002
- RFC 3260, "New Terminology and Clarifications for Diffserv," 2002
- RFC 3289, "Management Information Base for the Differentiated Services Architecture," 2002
- RFC 3290, "An Informal Management Model for Diffserv Routers," 2002
- RFC 3306, "Unicast-Prefix-based IPv6 Multicast," 2002
- RFC 3307, "Allocation Guidelines for IPv6 Multicast Addresses," 2002
- RFC 3315, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," 2003
- RFC 3317, "Differentiated Services Quality of Service Policy Information Base," 2003
- RFC 3319, "Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers," 2003
- RFC 3353, "Overview of IP Multicast in a Multi-Protocol Label Switching (MPLS) Environment," 2002
- RFC 3376, "Internet Group Management Protocol, Version 3," 2002
- RFC 3493, "Basic Socket Interface Extensions for IPv6" 2003

- RFC 3514, "The Security Flag in the IPv4 Header," April 1, 2003
- RFC 3526, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)," 2003
- RFC 3542, "Advanced Sockets Application Program Interface (API) for IPv6," 2003
- RFC 3569, "An Overview of Source-Specific Multicast (SSM)," 2003
- RFC 3582, "Goals for IPv6 Site-Multihoming Architectures," 2003
- RFC 3587, "IPv6 Global Unicast Address Format," 2003
- RFC 3590, "Source Address Selection for the Multicast Listener Discovery (MLD) Protocol," 2003
- RFC 3596, "DNS Extensions to Support IP Version 6," 2003
- RFC 3602, "The AES-CBC Cipher Algorithm and Its Use with IPsec," 2003
- RFC 3631, "Security Mechanisms for the Internet," 2003
- RFC 3633, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6," 2003
- RFC 3646, "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," 2003
- RFC 3701, "6bone (IPv6 Testing Address Allocation) -Phaseout," 2004
- RFC 3715, "IPsec-Network Address Translation (NAT) Compatibility Requirements," 2004
- RFC 3717, "IP over Optical Networks: A Framework," 2004
- RFC 3736, "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6," 2004
- RFC 3739, "Internet X.509 Public Key Infrastructure: Qualified Certificates Profile," 2004
- RFC 3740, "The Multicast Group Security Architecture," 2004
- RFC 3748, "Extensible Authentication Protocol (EAP)," 2004
- RFC 3754, "IP Multicast in Differentiated Services (DS) Networks," 2004
- RFC 3756, "IPv6 Neighbor Discovery (ND) Trust Models and Threats," 2004
- RFC 3765, "NOPEER Community for Border Gateway Protocol (BGP) Route Scope Control," 2004
- RFC 3769, "Requirements for IPv6 Prefix Delegation," 2004
- RFC 3776, "Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents," 2004
- RFC 3789, "Introduction to the Survey of IPv4 Addresses in Currently Deployed IETF Standards Track and Experimental Documents," 2004

- RFC 3790, "Survey of IPv4 Addresses in Currently Deployed IETF Internet Area Standards Track and Experimental Documents," 2004
- RFC 3791, "Survey of IPv4 Addresses in Currently Deployed IETF Routing Area Standards Track and Experimental Documents," 2004
- RFC 3792, "Survey of IPv4 Addresses in Currently Deployed IETF Security Area Standards Track and Experimental Documents," 2004
- RFC 3793, "Survey of IPv4 Addresses in Currently Deployed IETF Sub-IP Area Standards Track and Experimental Documents," 2004
- RFC 3794, "Survey of IPv4 Addresses in Currently Deployed IETF Transport Area Standards Track and Experimental Documents," 2004
- RFC 3795, "Survey of IPv4 Addresses in Currently Deployed IETF Application Area Standards Track and Experimental Documents," 2004
- RFC 3796, "Survey of IPv4 Addresses in Currently Deployed IETF Operations & Management Area Standards Track and Experimental Documents," 2004
- RFC 3810, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6," 2004
- RFC 3849, "IPv6 Documentation Address," 2004
- RFC 3879, "Deprecating Site Local Addresses," 2004
- RFC 3898, "Network Information Service (NIS) Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," 2004
- RFC 3901, "DNS IPv6 Transport Operational Guidelines," 2004
- RFC 3936, "Procedures for Modifying the Resource reSerVation Protocol (RSVP)," 2004
- RFC 3947, "Negotiation of NAT-Traversal in the IKE," 2005
- RFC 3948, "UDP Encapsulation of IPsec ESP Packets," 2005
- RFC 3956, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address," 2004
- RFC 3963, " Network Mobility (NEMO) Basic Support Protocol," 2005
- RFC 3964, "Security Considerations for 6to4," 2004
- RFC 3971, "Secure Neighbor Discovery," 2005
- RFC 3972, "Cryptographically Generated Addresses (CGA)," 2005
- RFC 3973, "Protocol Independent Multicast, Dense Mode (PIM-DM): Protocol Specification (Revised)," 2005
- RFC 3986, "Uniform Resource Identifier (URI): Generic Syntax," 2005
- RFC 4007, "IPv6 Scoped Address Architecture," 2005
- RFC 4029, "Scenarios and Analysis for Introducing IPv6 into ISP Networks," 2005

- RFC 4033, "DNS Security Introduction and Requirements," 2005
- RFC 4034, "Resource Records for the DNS Security Extensions," 2005
- RFC 4035, "Protocol Modifications for the DNS Security Extensions," 2005
- RFC 4038, "Application Aspects of IPv6 Transition," 2005
- RFC 4057, "IPv6 Enterprise Network Scenarios," 2005
- RFC 4065, "Instructions for Seamoby and Experimental Mobility Protocol IANA Allocations," 2005
- RFC 4074, "Common Misbehavior Against DNS Queries for IPv6 Addresses," 2005
- RFC 4075, "Simple Network Time Protocol (SNTP) Configuration Option for DHCPv6," 2005
- RFC 4076, "Renumbering Requirements for Stateless Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," 2005
- RFC 4094, "Analysis of Existing Quality-of-Service Signaling Protocols," 2005
- RFC 4106, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)," 2005
- RFC 4107, "Guidelines for Cryptographic Key Management," 2005
- RFC 4109, "Algorithms for Internet Key Exchange version 1 (IKEv1)," 2005
- RFC 4113, "Management Information Base for the User Datagram Protocol," 2005
- RFC 4135, "Goals of Detecting Network Attachment in IPv6," 2005
- RFC 4140, "Hierarchical Mobile IPv6," 2005
- RFC 4159, "Deprecation of 'ip6.int,'" 2005
- RFC 4177, "Architectural Approaches to Multi-homing for IPv6," 2005
- RFC 4191, "Default Router Preferences and More-Specific Routes," 2005
- RFC 4192, "Procedures for Renumbering an IPv6 Network without a Flag Day," 2005
- RFC 4193, "Unique Local IPv6 Unicast Addresses," 2005
- RFC 4213, "Basic Transition Mechanisms for IPv6 Hosts and Routers," 2005
- RFC 4215, "Analysis on IPv6 Transition in Third Generation Partnership Project (3GPP) Networks," 2005
- RFC 4218, "Threats Relating to IPv6 Multihoming Solutions," 2005
- RFC 4219, "Things Multihoming in IPv6 (MULTI6) Developers Should Think About," 2005
- RFC 4225, "Mobile IP Version 6 Route Optimization Security Design Background," 2005

- RFC 4241, "A Model of IPv6/IPv4 Dual Stack Internet Access Service," 2005
- RFC 4242, "Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," 2005
- RFC 4243, "Vendor-Specific Information Suboption for the Dynamic Host Configuration Protocol (DHCP) Relay Agent Option," 2005
- RFC 4260, "Mobile IPv6 Fast Handovers for 802.11 Networks," 2005
- RFC 4280, "Dynamic Host Configuration Protocol (DHCP) Options for Broadcast and Multicast Control Servers," 2005
- RFC 4282, "The Network Access Identifier," 2005
- RFC 4283, "Mobile Node Identifier Option for Mobile IPv6 (MIPv6)," 2005
- RFC 4285, "Authentication Protocol for Mobile IPv6," 2006
- RFC 4286, "Multicast Router Discovery (MRD)," 2005
- RFC 4291, "IP Version 6 Addressing Architecture," 2006
- RFC 4301, "Security Architecture for the Internet Protocol," 2005
- RFC 4302, "IP Authentication Header," 2005
- RFC 4303, "IP Encapsulating Security Payload (ESP)," 2005
- RFC 4305, "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)," 2005
- RFC 4306, "Internet Key Exchange (IKEv2) Protocol," 2005
- RFC 4307, "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)," 2005
- RFC 4308, "Cryptographic Suites for IPsec," 2005
- RFC 4309, "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)," 2005
- RFC 4338, "Transmission of IPv6, IPv4, and Address Resolution Protocol (ARP) Packets over Fibre Channel," 2006
- RFC 4339, "IPv6 Host Configuration of DNS Server Information Approaches," 2006
- RFC 4359, "The Use of RSA/SHA-1 Signatures within Encapsulating Security Payload (ESP) and Authentication Header (AH)," 2006
- RFC 4380, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," 2006
- RFC 4429, "Optimistic Duplicate Address Detection (DAD) for IPv6," 2006
- RFC 4443, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," 2006

- RFC 4449, "Securing Mobile IPv6 Route Optimization Using a Static Shared Key," 2006
- RFC 4456, "BGP Route Reflection," 2006
- RFC 4472, "Operational Considerations and Issues with IPv6 DNS," 2006
- RFC 4477, "Dynamic Host Configuration Protocol (DHCP): IPv4 and IPv6 Dual-Stack Issues," 2006
- RFC 4487, "Mobile IPv6 and Firewalls: Problem Statement," 2006
- RFC 4489, "A Method for Generating Link-Scoped IPv6 Multicast Addresses," 2006
- RFC 4554, "Use of VLANs for IPv4-IPv6 Coexistence in Enterprise Networks," 2006
- RFC 4555, "IKEv2 Mobility and Multihoming Protocol (MOBIKE)," 2006
- RFC 4581, "Cryptographically Generated Addresses (CGA) Extension Field Format," 2006
- RFC 4584, "Extensions to Sockets API for Mobile IPv6," 2006
- RFC 4601, "Internet Group Management Protocol, Version 2," 2006
- RFC 4604, "Using MLDv2 for Source Specific Multicast," 2006
- RFC 4620, "IPv6 Node Information Queries," 2006
- RFC 4621, "Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol," 2006
- RFC 4659, "BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN," 2006
- RFC 4677, "The Tao of IETF: A Novice's Guide to the Internet Engineering Task Force," 2006
- RFC 4703, "Resolution of Fully Qualified Domain Name (FQDN) Conflicts among Dynamic Host Configuration Protocol (DHCP) Clients," 2006
- RFC 4727, "Experimental Values in IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers," 2006
- RFC 4760, "Multiprotocol Extensions for BGP-4," 2007
- RFC 4779, "ISP IPv6 Deployment Scenarios in Broadband Access Networks," 2007
- RFC 4787, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP," 2007
- RFC 4795, "Link-Local Multicast Name Resolution (LLMNR)," 2007
- RFC 4798, "Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)," 2007
- RFC 4822, "RIPv2 Cryptographic Authentication," 2007

- RFC 4831, "Goals for Network-Based Localized Mobility Management (NETLMM)," 2007
- RFC 4835, "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)," 2007
- RFC 4852, "IPv6 Enterprise Network Analysis—IP Layer 3 Focus," 2007
- RFC 4861, "Neighbor Discovery for IP Version 6," 2007
- RFC 4862, "IPv6 Stateless Address Autoconfiguration," 2007
- RFC 4864, "Local Network Protection for IPv6," 2007
- RFC 4866, "Enhanced Route Optimization for Mobile IPv6," 2007
- RFC 4877, "Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture," 2007
- RFC 4884, "Extended ICMP to Support Multi-Part Messages," 2007
- RFC 4885, "Network Mobility Support Terminology," 2007
- RFC 4887, "Network Mobility Home Network Models," 2007
- RFC 4890, "Recommendations for Filtering ICMPv6 Messages in Firewalls," 2007
- RFC 4891, "Using IPsec to Secure IPv6-in-IPv4 Tunnels," 2007
- RFC 4908, "Multi-homing for small scale fixed network Using Mobile IP and NEMO," 2007
- RFC 4941, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," 2007
- RFC 4942, "IPv6 Transition/Coexistence Security Considerations," 2007
- RFC 4944, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," 2007
- RFC 4957, "Link-Layer Event Notifications for Detecting Network Attachments," 2007
- RFC 4966, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status," 2007
- RFC 4982, "Support for Multiple Hash Algorithms in Cryptographically Generated Addresses (CGAs)," 2007
- RFC 5015, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)," 2007
- RFC 5059, "Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM)," 2008
- RFC 5065, "Autonomous System Confederations for BGP," 2007
- RFC 5072, "IP Version 6 over PPP," 2007
- RFC 5084, "Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS)," 2007

- RFC 5094, "Mobile IPv6 Vendor Specific Option," 2007
- RFC 5095, "Deprecation of Type 0 Routing Headers in IPv6," 2007
- RFC 5096, "Mobile IPv6 Experimental Messages," 2007
- RFC 5120, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)," 2008
- RFC 5142, "Mobility Header Home Agent Switch Message," 2008
- RFC 5149, "Service Selection for Mobile IPv6," 2008
- RFC 5156, "Special Use IPv6 Addresses," 2008
- RFC 5157, "IPv6 Implications for Network Scanning," 2008
- RFC 5164, "Mobility Services Transport: Problem Statement," 2008
- RFC 5172, "Negotiation for IPv6 Datagram Compression using IPv6 Control Protocol," 2008
- RFC 5181, "IPv6 Deployment Scenarios in 802.16 Networks," 2008
- RFC 5213, "Proxy Mobile IPv6," 2008
- RFC 5214, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)," 2008
- RFC 5220, "Problem Statement for Default Address Selection in Multi-Prefix Environments," 2008
- RFC 5247, "Extensible Authentication Protocol (EAP) Key Management Framework," 2008
- RFC 5268, "Mobile IPv6 Fast Handovers," 2008
- RFC 5270, "Mobile IPv6 Fast Handovers over IEEE 802.16e Networks," 2008
- RFC 5271, "Mobile IPv6 Fast Handovers for 3G CDMA Networks," 2008
- RFC 5308, "Routing IPv6 with IS-IS," 2008
- RFC 5340, "OSPF for IPv6 (OSPFv3)," 2008
- RFC 5350, "IANA Considerations for the IPv4 and IPv6 Router Alert Options," 2008
- RFC 5375, "IPv6 Unicast Address Assignment Considerations," 2008
- RFC 5380, "Hierarchical Mobile IPv6 (HMIPv6) Mobility Management," 2008
- RFC 5453, "Reserved IPv6 Interface Identifiers," 2009
- RFC 5492, "Capabilities Advertisement with BGP-4," 2009
- RFC 5555, "Mobile IPv6 Support for Dual Stack Hosts and Routers," 2009
- RFC 5568, "Mobile IPv6 Fast Handovers," 2009
- RFC 5569, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)," 2010

- RFC 5571, "Softwire Hub and Spoke Deployment Framework with Layer Two Tunneling Protocol Version 2 (L2TPv2)," 2009
- RFC 5572, "IPv6 Tunnel Broker with the Tunnel Setup Protocol," 2010
- RFC 5637, "Authentication, Authorization, and Accounting (AAA) Goals for Mobile IPv6," 2009
- RFC 5648, "Multiple Care-of Addresses Registration," 2009
- RFC 5722, "Handling of Overlapping IPv6 Fragments," 2009
- RFC 5739, "IPv6 Configuration in Internet Key Exchange Protocol Version 2 (IKEv2)," 2010
- RFC 5796, "Authentication and Confidentiality in Protocol Independent Multicast Sparse Mode (PIM-SM) Link-Local Messages," 2010
- RFC 5838, "Support of Address Families in OSPFv3," 2010
- RFC 5844, "IPv4 Support for Proxy Mobile IPv6," 2010
- RFC 5846, "Binding Revocation for IPv6 Mobility," 2010
- RFC 5847, "Heartbeat Mechanism for Proxy Mobile IPv6," 2010
- RFC 5871, "IANA Allocation Guidelines for the IPv6 Routing Header," 2010
- RFC 5887, "Renumbering Still Needs Work," 2010
- RFC 5902, "IAB Thoughts on IPv6 Network Address Translation," 2010
- RFC 5909, "Securing Neighbor Discovery Proxy: Problem Statement," 2010
- RFC 5942, "IPv6 Subnet Model: the Relationship between Links and Subnet Prefixes," 2010
- RFC 5944, "IP Mobility Support for IPv4," 2010
- RFC 5952, "A Recommendation for IPv6 Address Text Representation," 2010
- RFC 5969, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)—Protocol Specification," 2010
- RFC 5991, "Teredo Security Updates," 2010
- RFC 5996, "Internet Key Exchange Protocol Version 2 (IKEv2)," 2010
- RFC 6014, "Cryptographic Algorithm Identifier Allocation for DNSSEC," 2010
- RFC 6036, "Emerging Service Provider Scenarios for IPv6 Deployment," 2010
- RFC 6040, "Tunneling of Explicit Congestion Notification," 2010
- RFC 6052, "IPv6 Addressing of IPv4/IPv6 Translators," 2010
- RFC 6071, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," 2011
- RFC 6081, "Teredo Extensions," 2011

- RFC 6085, "Address Mapping of IPv6 Multicast Packets on Ethernet," 2011
- RFC 6089, "Flow Bindings in Mobile IPv6 and Network Mobility (NEMO) Basic Support," 2011
- RFC 6092, "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service," 2011
- RFC 6104, "Rogue IPv6 Router Advertisement Problem Statement," 2011
- RFC 6105, "IPv6 Router Advertisement Guard," 2011
- RFC 6106, "IPv6 Router Advertisement Options for DNS Configuration," 2010
- RFC 6119, "IPv6 Traffic Engineering in IS-IS," 2011
- RFC 6144, "Framework for IPv4/IPv6 Translation," 2011
- RFC 6145, "Stateless IP/ICMP Translation," 2011
- RFC 6146, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers," 2011
- RFC 6147, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers," 2011
- RFC 6151, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms," 2011
- RFC 6164, "Using 127-Bit IPv6 Prefixes on Inter-Router Links," 2011
- RFC 6169, "Security Concerns with IP Tunneling," 2011
- RFC 6177, "IPv6 Address Assignment to End Sites," 2011
- RFC 6180, "Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment," 2011
- RFC 6221, "Lightweight DHCPv6 Relay Agent," 2011
- RFC 6226, "PIM Group-to-Rendezvous-Point Mapping," 2011
- RFC 6250, "Evolution of the IP Model," 2011
- RFC 6269, "Issues with IP address sharing," 2011
- RFC 6273, "The Secure Neighbor Discovery (SEND) Hash Threat Analysis," 2011
- RFC 6275, "Mobility Support in IPv6," 2011
- RFC 6279, "Proxy Mobile IPv6 (PIMPv6) Localized Routing Problem Statement," 2011
- RFC 6282, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," 2011
- RFC 6294, "Survey of Proposed Use Cases for the IPv6 Flow Label," 2011
- RFC 6296, "IPv6-to-IPv6 Network Prefix Translation," 2011

- RFC 6302, "Logging Recommendations for Internet-Facing Servers," 2011
- RFC 6308, "The Internet Multicast Address Allocation Architecture," 2011
- RFC 6324, "Routing Loop Attack Using IPv6 Automatic Tunnels," 2011
- RFC 6326, "Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS," 2011
- RFC 6333, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion," 2011
- RFC 6334, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Option for Dual-Stack Lite," 2011
- RFC 6343, "Advisory Guidelines for 6to4 Deployment," 2011
- RFC 6398, "IP Router Alert Considerations and Usage," 2011
- RFC 6422, "Relay Supplied DHCP Options," 2011
- RFC 6434, "IPv6 Node Requirements," 2011
- RFC 6436, "Rationale for Update to the IPv6 Flow Label Specification," 2011
- RFC 6437, "IPv6 Flow Label Specification," 2011
- RFC 6438, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels," 2011
- RFC 6459, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)," 2012
- RFC 6494, "Certificate Profile and Certificate Management for SEcure Neighbor Discovery (SEND)," 2012
- RFC 6495, "Subject Key Identifier (SKI) SEcure Neighbor Discovery (SEND) Name Type Fields," 2012
- RFC 6535, "Dual Stack Hosts using the 'Bump-in-the-Host' Technique (BIHS)," 2012
- RFC 6540, "IPv6 Support Required for All IP-Capable Nodes," 2012
- RFC 6543, "Reserved IPv6 Interface Identifier for Proxy Mobile IPv6," 2012
- RFC 6553, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams," 2012
- RFC 6554, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)," 2012
- RFC 6555, "Happy Eyeballs: Success with Dual-Stack Hosts," 2012
- RFC 6556, "Testing Eyeball Happiness," 2012
- RFC 6564, "A Uniform Format for IPv6 Extension Headers," 2012
- RFC 6572, "RADIUS Support for Proxy Mobile IPv6," 2012

- RFC 6583, "Operational Neighbor Discovery Problems," 2012
- RFC 6586, "Experiences from an IPv6-only Network," 2012
- RFC 6603, "Prefix Exclude Option for DHCPv6-based Prefix Delegation," 2012
- RFC 6610, "DHCP Options for Home Information Discovery in Mobile IPv6 (MIPv6)," 2012
- RFC 6611, "Mobile IPv6 (MIPv6) Bootstrapping for the Integrated Scenario," 2012
- RFC 6618, "Mobile IPv6 Security Framework Using Transport Layer Security for Communication between the Mobile Node and Home Agent," 2012
- RFC 6619, "Scalable Operation of Address Translators with Per-Interface Bindings," 2012
- RFC 6620, "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses," 2012
- RFC 6621, "Simplified Multicast Forwarding," 2012
- RFC 6655, "AES-CCM Cipher Suites for Transport Layer Security (TLS)," 2012
- RFC 6705, "Localized Routing for Proxy Mobile IPv6," 2012
- RFC 6724, "Default Address Selection for Internet Protocol Version 6 (IPv6), 2012
- RFC 6775, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," 2012
- RFC 6791, "Stateless Source Address Mapping for ICMPv6 Packets," 2012
- RFC 6822, "IS-IS Multi-Instance," 2012
- RFC 6830, "The Locator/ID Separation Protocol (LISP)," 2013
- RFC 6831, "The Locator/ID Separation Protocol (LISP) for Multicast Environments," 2013
- RFC 6832, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites," 2013
- RFC 6833, "Locator/ID Separation Protocol (LISP) Map-Server Interface," 2013
- RFC 6834, "Locator/ID Separation Protocol (LISP) Map-Versioning," 2013
- RFC 6835, "The Locator/ID Separation Protocol Internet Groper (LIG)," 2013
- RFC 6836, "Locator/ID Separation Protocol Alternative Logical Topology (LISP +ALT)," 2013
- RFC 6845, "OSPF Hybrid Broadcast and Point-to-Multipoint Interface Type," 2013
- RFC 6853, "DHCPv6 Redundancy Deployment Considerations," 2013
- RFC 6860, "Hiding Transit-Only Networks in OSPF," 2013

- RFC 6866, "Problem Statement for Renumbering IPv6 Hosts with Static Addresses in Enterprise Networks," 2013
- RFC 6874, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers," 2013
- RFC 6877, "464XLAT: Combination of Stateful and Stateless Translation," 2013
- RFC 6879, "IPv6 Enterprise Network Renumbering Scenarios, Considerations, and Methods," 2013
- RFC 6883, "IPv6 Guidance for Internet Content Providers and Application Service Providers," 2013
- RFC 6886, "NAT Port Mapping Protocol (NAT-PMP), 2013
- RFC 6888, "Common Requirements for Carrier-Grade NATs (CGNs)," 2013
- RFC 6889, "Analysis of Stateful 64 Translation," 2013
- RFC 6895, "Domain Name System (DNS) IANA Considerations," 2013
- RFC 6908, "Deployment Considerations for Dual-Stack Lite," 2013
- RFC 6909, "IPv4 Traffic Offload Selector Option for Proxy Mobile IPv6," 2013
- RFC 6911, "RADIUS Attributes for IPv6 Access Networks," 2013
- RFC 6921, "Design Considerations for Faster-Than-Light (FTL) Communication," April 1, 2013
- RFC 6939, "Client Link-Layer Address Option in DHCPv6," 2013
- RFC 6946, "Processing of IPv6 'Atomic' Fragments," 2013
- RFC 6959, "Source Address Validation Improvement (SAVI) Threat Scope," 2013
- RFC 6977, "Triggering DHCPv6 Reconfiguration from Relay Agents," 2013
- RFC 6980, "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery," 2013
- RFC 6992, "Routing for IPv4-Embedded IPv6 Packets," 2013
- RFC 7010, "IPv6 Site Renumbering Gap Analysis," 2013
- RFC 7021, "Assessing the Impact of Carrier-Grade NAT on Network Applications," 2013
- RFC 7031, "DHCPv6 Failover Requirements," 2013
- RFC 7039, "Source Address Validation Improvement (SAVI) Framework," 2013
- RFC 7040, "Public IPv4-over-IPv6 Access Network," 2013
- RFC 7045, "Transmission and Processing of IPv6 Extension Headers," 2014
- RFC 7048, "Neighbor Unreachability Detection Is Too Impatient," 2014
- RFC 7050, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis," 2013

- RFC 7051, "Analysis of Solution Proposals for Hosts to Learn NAT64 Prefix," 2013
- RFC 7059, "A Comparison of IPv6-over-IPv4 Tunnel Mechanisms," 2013
- RFC 7066, "IPv6 for Third Generation Partnership Project (3GPP) Cellular Hosts," 2013
- RFC 7077, "Update Notifications for Proxy Mobile IPv6," 2013
- RFC 7078, "Distributing Address Selection Policy Using DHCPv6," 2014
- RFC 7084, "Basic Requirements for IPv6 Customer Edge Routers," 2013
- RFC 7094, "Architectural Considerations of IP Anycast," 2014
- RFC 7098, "Using the IPv6 Flow Label for Load Balancing in Server Farms," 2013
- RFC 7101, "List of Internet Official Protocol Standards: Replaced by a Web Page," 2013
- RFC 7108, "A Summary of Various Mechanisms Deployed at L-Root for the Identification of Anycast Nodes," 2014
- RFC 7109, "Flow Bindings Initiated by Home Agents for Mobile IPv6," 2014
- RFC 7112, "Implications of Oversized IPv6 Header Chains," 2014
- RFC 7113, "Implementation Advice for IPv6 Router Advertisement Guard (RA Guard)," 2014
- RFC 7123, "Security Implications of IPv6 on IPv4 Networks," 2014
- RFC 7136, "Significance of IPv6 Interface Identifiers," 2014
- RFC 7148, "Prefix Delegation Support for Proxy Mobile IPv6," 2014
- RFC 7156, "Diameter Support for Proxy Mobile IPv6 Localized Routing," 2014
- RFC 7157, "IPv6 Multihoming without Network Address Translation," 2014
- RFC 7161, "Proxy Mobile IPv6 (PMIPv6) Multicast Handover Optimization by the Subscription Information Acquisition through the LMA (SIAL)," 2014
- RFC 7168, "The Hyper Text Coffee Pot Control Protocol for Tea Efflux Appliances (HTCPCP-TEA)," April 1, 2014
- RFC 7217, "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)," 2014
- RFC 7219, "SEcure Neighbor Discovery (SEND) Source Address Validation Improvement (SAVI)," 2014
- RFC 7221, "Handling of Internet-Drafts by IETF Working Groups," 2014

- RFC 7222, “Quality-of-Service Option for Proxy Mobile IPv6,” 2014
- RFC 7225, “Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol (PCP),” 2014
- RFC 7279, “An Acceptable Use Policy for New ICMP Types and Codes,” 2014

APPENDIX B
Recommended Reading

This appendix provides a list of books that I recommend:

- *Planning for IPv6*, by Silvia Hagen (O'Reilly, 2011)
- *Practical IPv6 for Windows Administrators*, by Ed Horley (Apress, 2013)
- *IPv6 Security*, by Scott Hogg and Eric Vyncke (Cisco Press, 2009)
- *DNS and BIND on IPv6*, by Cricket Liu (O'Reilly, 2011)
- *Global IPv6 Strategies*, by Patrick Grossetete, Ciprian Popoviciu, and Fred Wettling (Cisco Press, 2008)
- *OSPF and IS-IS: Choosing an IGP for Large-Scale Networks*, by Jeff Doyle (Addison-Wesley, 2005)
- *Routing TCP/IP, Volume 1 (2nd edition)*, by Jeff Doyle (Cisco Press, 2005)
- *Deploying IPv6 Networks*, by Ciprian Popoviciu, Eric Levy-Abegnoli, and Patrick Grossetete (Cisco Press, 2006)
- *IPv6 for Enterprise Networks*, by Shannon McFarland, Muninder Sambi, Nikhil Sharma, and Sanjay Hooda (Cisco Press, 2011)
- *Mobile IPv6*, by Hesham Soliman (Addison-Wesley, 2004)
- *IPv6 Network Administration*, by David Malone and Niall Richard Murphy (O'Reilly, 2005)
- *The Biology of Transcendence: A Blueprint of the Human Spirit*, by Joseph Chilton Pearce (Park Street Press, 2004)
- *The View from the Center of the Universe*, by Joel. R. Primack and Nancy Ellen Abrams (Riverhead Books, 2007)

Index

Symbols

A

We'd like to hear your suggestions for improving our indexes. Send email to index@oreilly.com.

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

About the Author

Silvia Hagen is the author of the successful book *IPv6 Essentials* published by O'Reilly. She is owner and CEO of the Swiss Consulting and Education company Sunny Connection, which specializes in IPv6 and network and application performance troubleshooting. She has worked with IPv6 for more than 10 years by writing, teaching, and consulting enterprises in Europe and the United States for the integration of IPv6. She is the president of the Swiss IPv6 Council, which is a non-profit platform to support the integration of IPv6 in Switzerland. As a result of these activities, Switzerland was the first country to reach a double-digit user adoption rate (10% in April 2013) and has therefore received the Jim Bound Award of the International IPv6 Forum for IPv6 World Leadership.

In her private time, Silvia likes to read, listen to music and go to concerts, meet with friends, be out in nature walking with her dog, and tend to her garden.

Colophon

The animal on the cover of *IPv6 Essentials, Third Edition*, is a rigatella snail. The rigatella snail (*Eobania vermiculata*) is native to the Mediterranean region, especially to Turkey and Crete. The snail lives in gardens, hedges, and dunes, where it feeds on vegetation. The snail got its scientific name because the rings on its shell resemble vermicelli (a type of pasta). It is also sometimes called the "noodle snail."

Rigatella snails commonly have about five brown rings on their cream-colored shells. Their eyes sit on stalks, or tentacles, which protrude from their heads. The snails are 17 to 21 millimeters high and 20 to 25 millimeters wide. They move by rhythmically contracting their muscular base, or foot. As they move, the snails secrete a colorless discharge that creates a type of carpet, which protects them from the surfaces on which they travel. This discharge is so effective that a snail could crawl along the blade of a razor and not be cut.

Rigatella snails are edible. They are one of the most popular types of snail used to make the European delicacy, escargot.

The cover image is a 19th-century engraving from *Cuvier's Animals*. The cover fonts are URW Typewriter and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.