The DHCP Handbook

Ralph Droms, Ph.D.
Ted Lemon

Second Edition

SAMS

Ralph Droms and Ted Lemon

The DHCP Handbook

Second Edition

SAMS

201 West 103rd Street, Indianapolis, Indiana 46290 USA

The DHCP Handbook, Second Edition

International Standard Book Number: 0-672-32327-3

Library of Congress Catalog Card Number: 2001095796

Printed in the United States of America

First Printing: October 2002

05 04 03 02 4 3 2 1

Trademarks

Warning and Disclaimer

Acquisitions Editors
Jenny L. Watson
Dayna Isley

Development Editor
Clint McCarty

Managing Editor
Charlotte Clapp

Project Editor
George E. Nedeff

Copy Editor
Kitty Wilson Jarrett

Indexer
Chris Barrick

Proofreader
Linda Seifert

Technical Editors
Richard Barr Hibbs
Bernie Volz

Team Coordinator
Amy Patton

Interior Designer
Gary Adair

Cover Designer
Sandra Schroeder

Page Layout
Ayanna Lacey

Contents at a Glance

Table of Contents

Preface to the Second Edition

Since the publication of the first edition of this book, DHCP has continued to grow in importance as a network management tool. Service providers have expanded the use of DHCP as a part of automated subscriber management systems that reduce their cost of operation. Inexpensive routers used for home and small office Internet connections now often include DHCP services that provide plug-and-play operation of computers that use private IP addressing.

In this second edition of *The DHCP Handbook*, the authors have updated their description of the details of DHCP to include new DHCP options and messages. Several new DHCP options have been defined since the publication of the first edition, including a mechanism for the authentication of DHCP messages, the Relay Agent Information option, through which relay agents can provide additional information to a DHCP server and the User Class option, through which a host gives more information about itself to the DHCP server. There is also a new message, DHCPFORCERENEW, through which a DHCP server can cause a DHCP client to contact the server for a new IP address and other configuration information. Redundant operation of DHCP servers can now be implemented through the DHCP failover protocol.

Another change in this edition is the use of the Microsoft Windows 2000 DHCP server in several examples. The Windows 2000 server has some new features, which are explained in detail. There is also a new user interface, which is illustrated in the screen shots of the operation of the Windows 2000 server throughout the book.

Although the fundamental definition of DHCP, as specified in RFC 2131 and RFC 2132 has not changed, there have been many developments in the function and use of DHCP since the first edition. The authors have updated many chapters and added several new chapters in this second edition. Chapters 10 and 18 describe the DHCP failover protocol and how to use it. Chapters 11 and 23 describe the details and configuration of DHCP–Dynamic DNS (DDNS) interaction. Our intention is to provide the most current and useful guide to the operation and use of DHCP in this second edition of *The DHCP Handbook*.

An Introduction to DHCP

Early in the development of the TCP/IP protocols, little motivation existed for automating the configuration of devices that use TCP/IP. Few computers used TCP/IP, and the computers that were networked weren't very portable. Perhaps most significantly, the majority of computers were shared among many users and had designated administrators who managed many operational details, including TCP/IP configuration.

Today, everything is different. An organization may have thousands—in some cases tens of thousands—of computers on its internal network. Devices ranging from mainframes to desktop computers to personal digital assistants (PDAs) to embedded processors are networked. Computers are highly mobile so that laptops, PDAs, and similar devices can move between network segments many times during the course of a day. And today's computers are not typically managed by trained system administrators. Most computers are set up and installed by users who aren't familiar with (and who probably don't want to know about) the arcane details of the TCP/IP protocol suite.

To meet the demands of plug-and-play operation through automating the configuration of networked computers, the IETF developed DHCP. DHCP provides automated, managed configuration of computers and other devices that use TCP/IP. Through DHCP, a network administrator can assign a network address and supply a subnet mask and a default router. DHCP is built around a client/server model in which networked computers (the clients) contact a centralized configuration server for configuration parameters. The administrator supplies the server with a description of the network infrastructure, along with rules about how to assign addresses and other configuration parameters. The server interacts directly with clients, according to the rules the administrator provides. Thus, the DHCP server acts as the network administrator's agent for managing the configurations of DHCP clients.

Through the DHCP server, you can control the assignment of addresses and the configuration of other TCP/IP protocol parameters in whatever way is appropriate for the network and the organization. You can use fully dynamic address assignment, preassign a specific address to each computer, or use a mixed strategy in which your server computers are assigned fixed addresses and other computers are assigned addresses on demand.

The bottom line is that DHCP allows you to build a networking system that enables your users to freely add new computers, replace existing computers, and move computers between networked locations, all without explicit intervention on the part of the users or of a network administrator. In fact, this introduction was written on a laptop that was connected to a campus network from Ralph Droms' home through an ADSL link, a campus office, and two campus classrooms (while he gave two final exams). Although each of these locations is serviced by a different part of the campus network, Ralph was able to simply turn on his laptop in each location and use the network immediately.

DHCP is currently a Draft Standard of the IETF. It is an open, vendor-independent standard. The specifications for DHCP are written in RFC 2131 and RFC 2132, which are available from `www.rfc-editor.org`. DHCP clients and servers are widely available from major software vendors, as well as from the ISC.

DHCP is a product of the Dynamic Host Configuration Working Group (DHCWG) of the IETF. The DHCWG first met at the IETF meeting in Cocoa Beach, Florida, in April 1989. At that meeting, the DHCWG defined the problem that it would address to be the automated configuration of TCP/IP hosts, including allocation of a network address and transmission of other parameters, such as the subnet mask and a default router.

DHCP is loosely based on BOOTP (RFC 951). DHCP retains the basic message format of BOOTP and the operation of BOOTP relay agents, and it shares the UDP ports initially assigned to BOOTP (67 and 68). This backward compatibility with BOOTP allows DHCP to use the installed base of BOOTP relay agents and avoid the requirement of a DHCP server on every network segment.

DHCP is still a work in progress. The DHCWG has several additional functions under development for DHCP, which are described in the last few chapters of this book. For current information on the status of DHCP and the activities of the DHCWG, visit `www.dhcp.org` or `www.ietf.org/html.charters/dhc-charter.html`.

Objectives of This Book

As we wrote this book, we set as our goal the development of a complete resource for understanding DHCP, designing DHCP services, and debugging problems with DHCP clients and servers. We start with the background and theory of DHCP, including message exchanges between clients and server, message formats, and an introduction to the ISC DHCP server. Next, we describe the implementation and operation of DHCP servers and clients. We spend more time describing the DHCP server than the client; the DHCP server is more interesting because it is the component that the network administrator usually interacts with, whereas the DHCP client simply runs automatically and in the background. We also discuss practical aspects of DHCP—why you should use it, when to use it, and how to design and run an efficient DHCP service.

We include examples and case studies of DHCP in operation throughout the book. We drew the case studies from our experience with DHCP in real IP networks, and we constructed the examples to illustrate specific concepts and ideas. Along with these examples and case studies, we included notes, tips, and warnings based on experience with the design of DHCP, the implementation of DHCP clients and servers, and the application of DHCP to production networks. We included this material to flesh out the framework of the theory and principles of DHCP with as much information about DHCP in practice as we could.

This Book's Audience

This book is intended for network planners, implementers, and administrators; in short, it is for anyone who must design, implement, manage, or debug a network that uses DHCP. Planners considering the use of DHCP or designing a DHCP service will find the protocol description and design guidelines of particular value. If you are not already familiar with the use and architecture of DHCP, be sure to read the first three chapters, which introduce DHCP through an example and explain some of the details of DHCP.

The discussion of DHCP and its applications assumes some familiarity with the details of the TCP/IP protocols. In particular, we assume that you understand hardware and IP addressing, subnetting, routing, and some of the application-layer services, such as DNS. We review some aspects of TCP/IP that are specific to understanding and using DHCP in Chapter 4, "Configuring TCP/IP Stacks." For a more comprehensive introduction to TCP/IP, we recommend either *Internetworking with TCP/IP*, by Doug Comer, or *TCP/IP Illustrated*, by W. Richard Stevens.

Readers who are already using DHCP will find the material in this book on configuring and tuning a DHCP server of particular interest. Anyone running a large installation will want to read about reliable DHCP service.

The later chapters are intended for anyone who is tracking and planning for future developments in DHCP. We, the authors, are both participants in the IETF working group that is responsible for DHCP, and we have included material on current work within the IETF in areas such as authentication, interserver communication, DHCP/LDAP integration, and DHCP for IPv6.

Organization of This Book

The book is written in three main parts. Part I, "Introduction to DHCP," introduces DHCP through examples and provides some background on configuring TCP/IP protocol stacks.

Part II, "DHCP Theory of Operation," focuses on the specification and operation of DHCP. This section of the book begins with an explanation of the objectives of DHCP and the motivation behind the design decisions in DHCP. Part II also includes detailed descriptions of the DHCP message formats and message exchanges between clients and servers, as well as the role of relay agents in those message exchanges.

Part III, "DHCP Servers and Clients," begins with a description of the operation of the ISC and Microsoft DHCP servers and clients. Next, we explain how to configure the ISC server, with several specific examples. In the following chapters, we discuss more advanced topics in DHCP service design, such as customized client configurations, reliable DHCP service, tuning a DHCP service, and setting up DHCP in a small office. Part III concludes with material on current work in DHCP, including authentication, interaction between DHCP and DNS, communication between DHCP servers, and the development of DHCP for IPv6.

Throughout the book, we give examples that use the ISC DHCP server. This server is freely available and therefore accessible even to readers whose employers may already have purchased a commercial DHCP server and, thus, are not in a position to purchase whatever commercial DHCP server we might have used in our examples. Appendix A, "Microsoft DHCP Server Examples," includes a list of examples and expository text for the Microsoft DHCP server, keyed to the examples throughout the book that use the ISC DHCP server. This server is the commercial server that a reader is most likely to have ready access to.

We would have liked to provide examples for a wider variety of DHCP servers, but unfortunately, every DHCP server has a different configuration syntax, and we simply couldn't provide examples for all of them. We believe that by showing examples presented for two DHCP servers with very different configuration mechanisms, a reader using a third DHCP server will most likely understand each example as it relates to whatever configuration mechanism that server uses.

About the Authors

The authors of this text, Ralph Droms and Ted Lemon, bring extensive expertise and experience with DHCP and IP networking to this book. In this text, the authors combine their insights to create a unique perspective on the theory and design of the DHCP specification, as well as the practical aspects of implementing a DHCP server and running a DHCP service.

Ralph Droms, Ph.D., organized the DHCWG with Phil Gross in 1989. He has chaired the working group since its inception and is a key contributor to the design and development of DHCP. Ralph is also editor of the DHCP RFCs and continues to participate in the evolution of DHCP.

Since joining Cisco in 2000, Ralph has continued his work on DHCP and network management. Previously, he was a member of the Computer Science Department faculty at Bucknell University, where he guided students through the study of TCP/IP internetworking, operating systems, and computer architecture. Ralph has also been a member of the computer science faculty at Pennsylvania State University, and he was on the research staff at IBM and Burroughs (now Unisys).

As a consultant in network architecture and infrastructure design, Ralph has worked with large and small companies on a variety of TCP/IP issues, including network architecture, server strategies and configurations, and the use of DHCP, DNS, and other technologies in network management. Ralph served as co-director of the computer center at Bucknell, where he supervised the design and implementation of the campuswide multiprotocol network.

Ralph lives with his wife and two daughters in Westford, Massachusetts. You can reach him at `rdroms@cisco.com`.

Ted Lemon first encountered DHCP while working as a network administrator at Digital Equipment Corporation in the early 1990s. In 1996 Paul Vixie of the Internet Software Consortium became concerned that there was no high-quality open-source implementation of DHCP, and he asked Ted if he would be willing to produce one. The ISC DHCP distribution was the result.

As part of the work of producing the ISC DHCP distribution, Ted has been active in the IETF DHCWG since 1996. Along with Ralph, Bernie Volz, and Jim Bound, Ted is working on a new version of DHCP for IPv6, as well as extensions to the DHCP protocol for IPv4.

One of the important ways that open-source projects are improved is through examination of user feedback for ways to do things better and for common problems that users have. Ted has had a great deal of experience helping people with common problems with the various aspects of DHCP. His motivation in working on this book has been to help people who need to use DHCP to learn what they need to know to install and manage a DHCP installation *without sending him e-mail*.

Ted currently works for Nominum, Inc., a leading vendor of DHCP and DNS solutions.

Dedication

To my father, who inspired me to ask questions and encouraged me to find answers.

—Ralph Droms

To all my teachers, without whose kindness I would never have been able to do this, and to all the living beings, may this book help you in some small way to reach happiness.

—Ted Lemon

Acknowledgments

Thanks to everyone who contributed to this book. We had the pleasure and good fortune to work with a great team at Pearson. Linda Engelman, Jen Garrett, Dayna Isley, Kitty Jarrett, Shannon Leuma, Clint McCarty, Lisa Thibault, Karen Wachs, and Jenny Watson all provided guidance and support every step along the way in the writing and production of this book.

Without the work of the DHCWG IETF, we wouldn't have DHCP to write about.

We would like to thank Richard Barr Hibbs and Bernie Volz, our technical editors, who read the text with diligence that was above and beyond the call of duty. As a result of their efforts, we clarified some of the more difficult sections in the book. We also thank Mike Carney and Mark Sirota for their careful reading of the text and their helpful comments. We are extremely grateful to Thomas Hickman, Kim Kinnear, and Mark Stapp, all with Cisco Systems, who wrote or contributed to several chapters, as noted in the text.

From Ralph Droms:

Thanks to Cisco, Bucknell University, Vint Cerf and the Corporation for National Research Initiatives, Joe D'Andrea and my other friends at Quadritek Systems (now a part of Lucent Technologies), and Mike Carney of Sun Microsystems for supporting my activities as chair of the DHCWG and making it possible to write this book. Doug Comer first gave me an opportunity to learn which end of a packet is up.

I thank Ted for his experience with the practice of networking that he brought to this project, for his insightful input and careful review of my writing, and for his vision about how we could make this book truly educational and useful.

My wife, Jan, improved the book with her careful editing and suggestions for organizing its content. Jan, along with our daughters, Stacey and Becca, inspired me to tackle this project and see it through to completion.

From Ted Lemon:

Thanks to Paul Vixie for his compassion and help both in my professional and personal life. Thanks to David Conrad for his patience, kindness, and patience. Thanks to my mother and father for putting up with me and setting an example for me that I haven't always been able to appreciate. Thanks to Mel, for her great kindness, and to Signe, Queen of Repartee. Thanks to Scanner, Bob, Wendy, April, Betty, Stephen, Ben, James, George, Matt, Lucy, Ashton, Cristi, and all the other nice people at Nominum with whom it is a joy to work. Thanks to Richard Stallman, for teaching me the key to writing difficult programs: start typing.

Thanks to my wonderful wife, Andrea, for not being sad when I've stayed up through the night working on chapters, and for cheering me up when I'm tired and depressed, and to Phil and Sylvia, Mike, Debby, Martin, Ellen, and Keith—the best in-laws I could ever have hoped for. Thanks to Ralph for his thoughtful comments and to Jenny and Karen for not reaching through the phone lines to throttle me when I didn't come up with a chapter on time.

Thanks to Walt Congdon for taking me under his wing and showing me how to be a good person while pretending to teach me about ham radio. Thanks to my grandmother Carrie for showing me to look beyond my own concerns. Thanks to Hughes Pack for being stern when I needed it. Thanks to Hartley Pfeil for teaching me how to write. Thanks to Jennifer Bobbe for her patience and support when I was working on the first edition. Thanks to Kim Kinnear for all his hard work on the failover protocol and for putting up with all my suggestions, some of which may have been useful. Thanks to Edward Lemon, Sr., and Robert Dickerson, my grandfathers, for never believing something couldn't be done and letting a little of that attitude rub off on me (would that it had been more). Thanks to my grandmother, Leone Lemon, for trying to teach me to be considerate.

Last but definitely not least, thanks to Khen Rinpoche Geshe Lobsang Tharchin, Geshe Michael Roach, the one who calls himself Tenzin Gyatso, Geshe Thupten Rinchen, Winston McCullough, Elly van der Pas, Lobsang Chukyi, Lobsang Chunzom, Thupten Chudrun, John Brady, Sal Lupo, Seamus Rutherford, Rebecca Vinacour, Deborah Bye, Mercedes Bahleda, and Rafael Cervantes for gifts of unsurpassable value.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book's title and authors as well as your name and phone or email address. I will carefully review your comments and share them with the authors and editors who worked on the book.

Email: `networking@samspublishing.com`

Mail: Mark Taber
Associate Publisher
Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

For more information about this book or others from Sams Publishing, visit our Web site at `www.samspublishing.com`. Type the ISBN (excluding hyphens) or the title of the book in the Search box to find the book you're looking for.

PART I

Introduction to DHCP

IN THIS PART

1

An Introduction to DHCP

IN THIS CHAPTER

- Configuring Devices on a Network
- A First Attempt at Automating Device Configuration
- The Benefits of DHCP
- Assigning IP Addresses Using DHCP
- Perceived Problems of DHCP
- Address Allocation Policies

The *Dynamic Host Configuration Protocol (DHCP)* automates the process of configuring new and existing devices on TCP/IP networks. DHCP performs many of the same functions a network administrator carries out when connecting a computer to a network. DHCP enables a program to automatically manage policy decisions and bookkeeping tasks. Replacing manual configuration with a program adds flexibility, mobility, and control to networked computer configurations.

This chapter provides an overview of how network administrators allocate, manage, and configure IP addresses, and it shows how administrators can use DHCP to accomplish these same tasks. It also introduces some of the basic terminology required to understand the capabilities that the protocol provides and examines some reasons for, and caveats about, using DHCP.

Configuring Devices on a Network

Any network administrator using TCP/IP can testify that manually configuring computers attached to a network is a time-consuming and error-prone process. Indeed, at almost any site—regardless of whether DHCP is in use—the address assignment and configuration process is automated in some way.

One of the authors of this book, Ted Lemon, worked as a network administrator at the Digital Equipment Corporation (DEC) campus in Palo Alto, California, before DHCP was available to simplify the tasks of address management and configuration. The DEC campus used a central IP address administration system, which was based on a single list, or *host table*, of computers, IP addresses, and *Domain Name System* (*DNS*) names for the entire network.

To help introduce you to the tasks that a DHCP server performs, this section describes, from Ted's perspective, what network administrators did before DHCP became widely available.

As part of the network administration task, we network administrators updated the host table with new computers as they were added to the network and changed the entries for computers as their names and addresses changed. Periodically, we ran a shell script on the host table to update the DNS server database. We configured individual computers manually, from the entries in the host table, by physically walking up to each computer and entering the configuration information.

Users had a variety of questions about connecting their computers to the campus network. Usually, they wanted to know what IP address they could use for their computers. To respond to such questions, we asked the following:

- Who are you?
- Is this a new device, or was it connected to the network before?
- What is the device's old IP address?
- Where do you need to install this device?
- In what department do you work?

IP Address Allocation

After we obtained this information, we decided whether to give the user an IP address. It was usually easy to make this decision; if the user was an employee or a contractor working in a DEC Palo Alto building, we gave the user an address. Next, we decided what IP address to assign to the user. To do this, we had to know what network segments were present at the site, which segment or segments were available in the user's office, and how those network segments were configured.

If we supported a single network segment with a single IP subnet, answering these questions would have been simple, and everyone would have been allocated addresses from that subnet. However, the DEC Palo Alto campus network consisted of many network segments, routed together through a backbone network. Thus, it was a bit more difficult to assign IP addresses. In essence, each network administrator had to remember which network segments were available in which buildings, on which floors, and, in some cases, in which offices. If we remembered incorrectly, the address might have been allocated from the wrong subnet or the address might have already been assigned to another device, and we would have to perform the process again.

After we determined the network segment to which the user's computer would be attached, we determined whether any IP addresses were available on it and chose one for the user. If no IP addresses were available on the segment—and this was often the case—we examined the host table for addresses that appeared to be no longer in use. Occasionally, we configured a new network segment and moved some devices to it to expand the pool of available addresses.

Configuration Information

In addition to choosing an IP address on the correct subnet, we also provided the user with additional information about the network, which usually consisted of the following:

- The addresses of the default routers for the network segment to which the device was to be connected
- The addresses of primary and secondary domain name servers that the device would use
- The subnet mask and broadcast address

If the device needed specific network services that were not used by all devices on the network, we also informed the user how to access those services and programmed that information into the device. For example, we manually configured a diskless Network File System (NFS) client's NFS mount information, and we usually gave different information to each diskless NFS client.

Configuring Network Devices

In general, we got network configuration information into devices in two ways:

- When configuring a knowledgeable user's machine, we gave the information directly to that user; thus, it took a minute or two to configure a machine over the phone or via a single e-mail message exchange.
- For users who could not configure their own machines, we had to determine where the user was, walk to that user's station (possibly in a different building), log in as `root`, type the necessary information, restart the machine, and then verify that it worked correctly.

Moving Devices to Different Network Segments

From time to time, a user would move from one office to another, or a user's machine would move from a lab into an office. If the network segment (or segments) to which the user's devices were attached was not available in the new location, we would de-allocate the IP addresses previously assigned to those devices and allocate new addresses to the devices on the network segment (or segments) available in the new location.

Moving or Adding Network Services

As organizations within DEC Palo Alto grew, it was not uncommon for us to add new facilities such as printers, name servers, and Network Time Protocol (NTP) servers to the network and then manually configure the address for each client. Because we did not always have time to modify the configurations of existing functioning clients, we disseminated information about new network services when new machines were installed or when users complained that, for example, they couldn't access the printers closest to their cubicles.

Renumbering the Network

As the organization grew, we restructured the network. On one occasion, the entire DEC corporate network number changed from Class B (128.45.0.0) to Class A (16.0.0.0) addressing. This necessitated changing the IP address of every network device on the Palo Alto campus—more than 1,000 computers. Because we did not have an automatic configuration mechanism, network administrators had to renumber the machines, a process that consisted of walking to every machine and manually changing its IP address. Because people worked odd hours at DEC Palo Alto, we often forced people out of the buildings as we updated the machines. We then waited for users to report problems. This indicated which machines had not been renumbered correctly.

Reclaiming Disused IP Addresses

Machines for which we allocated IP addresses eventually failed, moved out of our jurisdiction, or were reassigned to different users and reinstalled, at which time the machines lost their old identities. When we were aware of these transitions, we easily updated our records and reclaimed the IP addresses belonging to such machines. However, if transfers occurred without our knowledge, we were not aware that these IP addresses were no longer in use.

In such cases, we had no reliable way to determine whether an IP address was no longer in use. Although we often used the `ping` command (that is, the ICMP echo request/reply protocol) to determine whether an address was still in use. Unfortunately, if there was no response to such a test, this indicated only that the address was not in use *at that moment*. The address could have been configured in a device that was powered off. We eventually found ways of handling this problem. First, we tried to locate the person who owned the device for which the IP address was allocated. If we couldn't find the owner of the device, we sent a `ping` to a suspect IP address periodically for about a month, and if we did not receive a response during that period, we reclaimed the address. Occasionally, someone would power up a device that had been disused for a few months, and the new device to which the old device's IP address was assigned would start behaving erratically.

A First Attempt at Automating Device Configuration

In 1985 Bill Croft and John Gilmore devised a protocol called *Bootstrap Protocol* (*BOOTP*). The idea behind BOOTP was to automate network device configuration. This could eliminate the need for the system administrator to manually configure each network device.

BOOTP requires that the administrator create a table that contains a list of BOOTP clients, their IP addresses, and other configuration parameters they might need. When a BOOTP client needs to configure itself, it broadcasts a request, which the BOOTP server receives. The BOOTP server looks up the client in the table, finds its parameters, and sends these parameters to the client.

BOOTP works fairly well, except that it only configures the device; the network administrator must perform the remaining tasks. Various sites experimented with dynamic address allocation using BOOTP, but they were not very successful because the protocol was limited in what it could do; it was a simple database lookup protocol, and it provided no means for reclaiming addresses.

DHCP is a direct descendent of BOOTP. DHCP packets and BOOTP packets look very much alike, and DHCP and BOOTP clients and servers can take advantage of the same network infrastructure. Both protocols accomplish the task of automatically configuring network devices; the difference is that DHCP can solve other problems.

The Benefits of DHCP

The tedious and time-consuming method of assigning IP addresses described in the section "Configuring Devices on a Network," earlier in this chapter, was once commonplace. However, thanks to DHCP, a network administrator no longer has to manually configure each new network device before it can be used on the network. DHCP has enabled the use of laptops that roam between networks, by eliminating the need for manual reconfiguration when a laptop changes its access point. The use of DHCP is especially important with wireless LANs, in which mobile devices can move between access points without even reconnecting a cable.

With the proliferation of DHCP, network administrators can also choose the level of control they want to exercise with regard to address allocation; they can still manually assign IP addresses to DHCP clients, or they can have the DHCP server automatically allocate IP addresses for clients. They can also decide whether clients must be registered before they are assigned IP addresses.

Availability of DHCP Clients

The widespread availability of DHCP clients is due in large part to Microsoft Corporation's early decision to include a DHCP client in the Windows 95 distribution. Since then, most major desktop operating system vendors have followed suit, and new Apple Macintosh, Windows 2000, and Windows XP systems come preconfigured to use DHCP when IP networking is enabled. Most free, Unix-like operating systems also come with DHCP clients that can be configured fairly easily, and the Internet Software Consortium (ISC) provides an open-source implementation of DHCP that includes a server, a client, and a relay agent. The ISC implementation of DHCP runs on many commercial Unix implementations, as well as on all free, Unix-like operating systems.

DHCP on Large Networks

Using DHCP on a large network offers clear benefits. When you must allocate and then configure a great number of devices, a protocol that completely automates these processes saves a tremendous amount of time. Even if you still manually allocate IP addresses for each client, DHCP's ability to automatically reclaim IP addresses from DHCP clients saves time and hassle in the long run.

Mobility

One big advantage of DHCP is that it allows for mobile devices (that is, devices that are plugged in at different network locations at varying times). For example, at the University of Oregon, where DHCP is used, network connectivity is provided in the dorms, the library, teachers' offices, and classrooms. When students do homework in their dorms, they plug their laptop computers into Ethernet jacks. When these students need to go to the library to work with reference material, they unplug their laptops from the network, put them to sleep, take them to the library, and plug them back in. The laptops automatically acquire new IP addresses and continue using the network.

Teachers can work on presentations on their laptop computers in their offices but store the presentations on a file server. A teacher can then use the same laptop that is plugged into the network hookup in a classroom to access the presentation from the file server, without reconfiguring the machine.

Visiting faculty members and salespeople with DHCP-ready laptop computers can plug those computers into the network and immediately use the network, without requiring any intervention from a system administrator. University of Oregon faculty members who go to conferences can bring their laptops, which are DHCP-ready, plug them into the terminal room network at the conference, and immediately use the network there.

DHCP on Small Networks

It can be convenient to set up a DHCP server on a very small network. Even though few clients are involved in such cases, the advantages of not having to manually configure each client's IP stack can be significant. Configuring a simple DHCP server for a single subnet shouldn't take much more time than configuring the IP stack on a new machine. You must configure the IP stack on the DHCP server manually, of course, but the time you save configuring the second machine attached to the local network makes up for the time you spent configuring the DHCP server. DHCP saves time on every machine configured from the third onward.

Of course, if you are an experienced network administrator, you will probably already know how to set up a DHCP server; if you are doing this for the first time, you might need to add five or six machines to the network before you realize any time savings. But in any case, you will have learned a valuable skill.

Another advantage of setting up DHCP on small networks is enhanced mobility. If a University of Oregon professor has a laptop computer configured to use DHCP and has a network at home, it is convenient to run DHCP on the home network. If that professor does not have DHCP service on his or her home network, then the professor must manually reconfigure the laptop computer every time he or she moves it from the home to the office or back again.

Assigning IP Addresses Using DHCP

A major difference between the ways in which the DHCP server and a network administrator allocate addresses is that DHCP enforces a limit on how long an IP address can be used. This seemingly subtle change makes many of the problems experienced with IP address allocation on the network at DEC Palo Alto much easier to solve. The rest of the protocol is analogous to what a network administrator does: A *DHCP client* (in the DHCP protocol, this refers to the device itself, not to the user) requests an IP address. The server, using its knowledge of the network and a list of IP addresses and client identities it maintains, provides one. Chapter 8, "DHCP Message Exchanges," and Chapter 15, "Configuring a DHCP Server," discuss this process in detail.

DHCP Server as Agent

The DHCP server acts as an agent in performing address allocation. Like a network administrator, the DHCP server must have a clear, unambiguous understanding of the network's layout in order to assign addresses, and it must know the network's address allocation policy.

NOTE

One of the most common errors new DHCP administrators make is thinking of the DHCP server as just another database lookup engine and, therefore, providing the DHCP server with only the information the administrator thinks it needs to know. Remember that it is just as important for the DHCP server to know what not to do as what to do.

Address Leasing

As explained earlier, DHCP servers must operate automatically, and are unable to exercise judgment or ask what happened to old devices. Further differentiating between DHCP and manual address allocation is the *lease*. Rather than simply assigning each client an IP address to keep until the client is done with it, the DHCP server assigns the client an IP address with a lease; the client is allowed to use this IP address only for the duration of that lease. When the lease expires, the client is forced to stop using that IP address. To prevent a lease from expiring, which essentially shuts down all network access for the client, the client must renew its lease on its IP address before it expires. Most DHCP clients renew their leases many times.

Address Reclamation with DHCP

By constraining clients from using IP addresses after their leases expire, and by providing a mechanism for clients to continue renewing their leases as long as they are powered on and connected to the network, DHCP enables the reliable reclamation of disused IP addresses. If a device is left powered off for an extended period, it must contact the DHCP server for its IP address when it is powered on again. If the device's former address is not available, it is given a new address. This prevents most address allocation conflicts.

Renumbering with DHCP

The lease mechanism also facilitates renumbering. If every device on a network uses DHCP, then renumbering is a simple matter of reconfiguring the server's idea of what the network looks like. It is possible to renumber so transparently that users who do not pay close attention to their TCP/IP configuration information are unaware that the network has been renumbered.

Describing Network Services with DHCP

In addition to providing a means for distributing IP addresses, DHCP enables configuration information to be distributed in the form of *DHCP options*, including the following:

- The default router addresses
- The domain name servers' addresses
- The name of a bootfile to load (for devices that boot over the network)
- The name of the root file system and swap server (for diskless clients)

Chapter 9, "DHCP Options," provides a complete list of these options. Chapter 11, "DHCP–DNS Interaction," describes how to use them, and Chapter 21, "DHCP Clients," discusses options some common clients use.

Moving or Adding Network Services with DHCP

When a network service is added or needs to be moved, it is possible to take advantage of the regular lease renewal process to propagate new information. The administrator simply updates the DHCP server configuration as appropriate. If a service's IP address changes, the configuration is updated to reflect this. If there is a new printer, that printer is added to the list of printers that are available on the subnet (or subnets) serving the area near the printer. As DHCP clients renew their leases, they automatically acquire this new information and begin using it.

One problem with updating information about services through the DHCP server is that the DHCP client, not the DHCP server, decides when to renew its lease. The new configuration information does not reach the DHCP client until the client renews its lease. If the DHCP client takes a long time to renew its lease, it does not get its new configuration information for a long time.

A new DHCP message, called `DHCPFORCERENEW`, allows DHCP servers to notify DHCP clients that new configuration information is available. When a DHCP client receives a `DHCPFORCERENEW` message from a DHCP server, the client contacts the server immediately. The server can then pass a new IP address and other configuration information to the client.

Perceived Problems of DHCP

Despite DHCP's benefits, many network administrators resist deploying DHCP because of various perceived problems, which are discussed in the following sections.

Excess Broadcast Traffic

Some network administrators believe that DHCP generates a large amount of broadcast traffic. Although DHCP does use broadcasts, the amount of broadcasting is relatively minimal. In the worst case, the first two DHCP messages that a DHCP client sends must be broadcast; depending on the networking capabilities of the operating systems on the client and server, the server might also need to broadcast its responses to the client. When combined, this creates a total of four broadcast packets.

In most cases, however, the client need only send one broadcast packet on startup; most clients do not require a broadcast response. Most servers can also respond without broadcasting. Thus, in the most typical DHCP startup case, only a single packet is broadcast—not four. After a client configures its network connection and either as long as that connection remains valid or until the client machine is restarted, all communication with the server is unicast. Chapter 7, "Transmitting DHCP Messages," discusses these issues in detail.

Another common misconception is that DHCP traffic is broadcast across the entire enterprise network. In fact, the broadcast traffic generated through the use of DHCP is typically limited to the network segment to which the DHCP client is connected. Traffic across the rest of the network between the DHCP client and server is usually sent directly to the server by using IP unicast.

In comparison, consider *Address Resolution Protocol* (*ARP*), which all IP broadcast networks use. When one device needs to communicate with another device that is on the same network segment, it must have the link-layer address of the second device. Because it initially knows only the other device's IP address, it must send an ARP broadcast to obtain the second device's link-layer address. After the device has that link-layer address, it periodically verifies that it still has the correct address by broadcasting another ARP request.

Some ARP implementations verify the link-layer address as frequently as every two minutes. The ARP response is also broadcast, so any pair of devices on a network segment that are in contact with one another using TCP/IP broadcast, on average, one ARP message every minute. Computers tend to remain powered on for longer than one minute at a time; usually, computers are powered on all day. In this case, ARP generates as many as 180 broadcast packets for every packet DHCP generates. Therefore, at four broadcast messages, DHCP is a comparatively insignificant producer of broadcast traffic.

Server Load

Another common assumption is that because a DHCP server is most likely serving all the DHCP clients at a site, it is difficult for a DHCP server to function at a large site. Fortunately, DHCP is comparatively undemanding. A name server for a large site might need a fairly fast machine with a good deal of memory. However, a DHCP server for the same site can usually run quite well on an old piece of junk found in the closet. It is very common to hear of people running the ISC DHCP server on an old Intel 486 machine running Linux, serving several thousand DHCP clients. Many sites serve on the order of 10,000 clients with Linux-based platforms. Although the

ISC DHCP server keeps its entire client database in memory, a 10,000-client network consumes, at most, 20MB of virtual memory.

DEALING WITH SPURIOUS DHCP TRAFFIC

One problem DHCP servers must deal with is broken clients that send too many requests. At Pacific Bell, a single DHCP server running on a Tandem mainframe serves more than 50,000 DHCP clients. Some of the old network hardware the company installed at some of its sites can get into a state in which it tries to obtain an IP address once every few seconds. On bad days, this means the DHCP server receives about 5 requests every second, with occasional sustained peaks of 50 packets per second that last for an hour or more. The DHCP server looks at all requests and decides what to do with them, even though these requests are not legitimate. Nonetheless, it handles this load without difficulty.

DHCP Reliability

One problem with DHCP is that if the DHCP client and the DHCP server are unable to communicate with each other for some reason, the DHCP client's lease eventually expires, and it must stop using the network. This commonly occurs if the DHCP server goes down for longer than the duration of a lease. This can also happen if a central DHCP server is configured to serve addresses on a *wide area network* (*WAN*) and one of the links in this WAN fails for a long time. In practice, temporary outages of DHCP service have little or no effect on clients because a client tries to extend the lease on its address well before the lease expires and continues to use its old address while attempting to contact the DHCP server.

SPEAKING FROM EXPERIENCE

At DEC Palo Alto, we actually considered deploying DHCP very early on. We ran a fairly homogenous environment—almost every machine was a DEC station of some sort, running Ultrix. Therefore, we could have fairly easily configured all machines to run a DHCP client on startup. However, we were afraid that if we did this, we would experience reliability problems because of the lease mechanism, and users would complain.

Based on subsequent experiences, it is clear that this fear was unfounded. Several strategies exist for avoiding this problem, and we found that these strategies work. Clients can renew their leases, and users do not call to complain about losing network connectivity.

If you are deploying DHCP on a WAN and don't have an extremely reliable, redundant network setup, it is best to locate DHCP servers at each site rather than run one central DHCP server for the entire WAN. Some sites do run a central DHCP server for their entire WAN, but they avoid trouble by having multiple redundant links to each site so that if one link goes down, an alternate path to the DHCP server is available.

DHCP does not currently allow a backup DHCP server to serve addresses from the same range as a primary server. Several strategies exist for working around this limitation:

- You can set leases to be long enough so that they do not expire before you fix a failed server.
- You can set up secondary servers that do not serve the same sets of addresses.
- If you conduct static address allocation, you can set up completely redundant DHCP servers.

Chapter 15, "Configuring a DHCP Server," discusses these strategies in detail.

The Internet Engineering Task Force (IETF) Dynamic Host Configuration Working Group (DHCWG) has developed a new protocol—DHCP failover—that allows DHCP servers to operate in primary/secondary pairs, allocating IP addresses out of a shared pool of addresses. The protocol allows the primary and secondary to share the DHCP service load equally. It also allows the secondary to be configured to act as a backup in case the primary server fails.

When Not to Use DHCP

Unlike manual network configuration, the automated DHCP process depends on the presence of a DHCP server. If you use DHCP to configure a device on a network, and if that device cannot talk to the DHCP server, it eventually stops using the network.

For this reason, using DHCP to configure network servers is not recommended. If the DHCP server fails and, as a result, the machine on which you are running your corporate name service suddenly loses its ability to communicate on the network, or your corporate SMTP (e-mail) gateway goes down, you will quickly forget how convenient DHCP is.

Some managed hubs and other network components obtain the IP address of their management port from a vendor-supplied custom BOOTP server. It might not be a good idea in some cases to substitute a general-purpose DHCP server for the manufacturer's controller, depending on how faithfully the manufacturer adhered to the protocol and whether you have sufficient documentation to configure the server. Also, as with other network infrastructure, it might be more reliable to configure the devices manually.

Address Allocation Policies

One of the primary goals for DHCP was to design a protocol that provides a mechanism through which a network administrator can implement any desired administrative policy. The network administrator can manually configure the DHCP server with an IP address for each machine that is connected to the network. The administrator can also simply provide a range of addresses for the DHCP server to use and allow the DHCP server to allocate these addresses to clients automatically. It is also feasible to implement a scheme that combines both of these methods.

Static Allocation

With *static (or fixed) allocation*, the DHCP server receives a list of identification information for DHCP clients. These identifiers specifically and uniquely identify each client. Chapter 16, "Client Identification and Fixed-Address Allocation," discusses the types of identifiers that can be used.

For each identifier, the administrator gives the DHCP server an IP address to assign to that client. If the client is mobile, the administrator can assign an address for each client on each network segment to which the client is connected. A client cannot configure itself on a network segment on which the administrator has not assigned it an IP address.

Dynamic Allocation

With *dynamic allocation*, the DHCP server receives a range of IP addresses for each network segment on which DHCP clients are expected to be configured. When a DHCP client asks for an IP address, the DHCP server finds a free address on that network segment and supplies it to the client.

Automatic Allocation

The DHCP specification talks about another method of address allocation, called *automatic allocation*, in which the DHCP server allocates IP addresses as it does in dynamic allocation, but the addresses are allocated permanently. The DHCP servers described in this book do not actually implement automatic allocation as a specific third alternative. However, you can approximate this scenario by simply using very long lease durations.

Hybrid Allocation Policies

A variety of hybrid address allocation policies are possible with DHCP. With one common policy, the administrator registers a list of known client identifiers for which DHCP service is allowed, but the administrator does not assign fixed IP addresses to those clients. Those clients can then acquire IP addresses dynamically wherever they are connected. This allows the administrator to limit the use of DHCP to registered clients, but it saves the administrator the trouble of updating the DHCP server every time a client moves. With the growing popularity of portable computers, this can be a major advantage.

Some DHCP servers can assign a fixed IP address for a DHCP client on the client's home network and enable the client to acquire a dynamically assigned IP address if it roams elsewhere.

Another fairly common strategy is to assign fixed addresses to registered DHCP clients but enable unregistered DHCP clients to acquire dynamically assigned addresses. This allows a user to get a specific IP address when he or she starts up the computer but allows the network administrator to avoid keeping track of every device that connects to the network.

Which of these strategies should be adopted at a particular site is a policy decision. Obviously, keeping track of which DHCP clients are connected to the network is a lot of work. However, in some environments it may be worth it.

NOTE

Not all servers support the same set of policies. Because DHCP does not specify what policies must be supported, DHCP server implementers must choose how these policies are implemented, and the choices they make might limit the range of possible policies. For example, the Microsoft DHCP server supports a much more limited set of policies than does the ISC DHCP server, even though both servers implement the same protocol.

You cannot use DHCP as a security mechanism. If a DHCP client elects not to follow the protocol, a network administrator can do little, other than track down the offending device and shut it off. A malicious user who wants to access the network can always simply make up an IP address, send an ARP request for it, and then, if it does not get an answer, use the fabricated address. Access control based on client identification can be very convenient, but it does not prevent unauthorized access to a network.

Summary

Every device that is attached to a network is configured with a unique address and with other information about the network. Historically, an administrator or a knowledgeable user manually configured each device. Addresses were allocated manually, and reclaiming addresses required a great deal of knowledge—and the process was still not very reliable.

BOOTP attempted to address this problem, but it fell short of a complete solution because it solved only the problem of configuring systems, and not the problem of allocating addresses.

DHCP automates the entire process of configuring devices to use the network. A properly configured network with a DHCP server can accommodate new devices with little or no administrator intervention, and it requires no special knowledge on the part of users.

You can configure DHCP in such a way that it is reliable and consumes few network resources. Furthermore, DHCP servers can run on fairly inexpensive computers, and it is so easy to set them up that it is cost-effective to do so on a very small network.

If you run any sort of IP network, then you will almost certainly use DHCP on that network. DHCP can greatly simplify your work when you understand how it functions.

2

An Example of DHCP in Operation

IN THIS CHAPTER

- Setting Up the GSI Network
- Using DHCP to Configure Computers
- Leases on IP Addresses in DHCP

DHCP provides a mechanism through which a computer can obtain an IP address and configuration parameters for its network protocol software. The DHCP server allocates an IP address and provides configuration information that is appropriate for the network segment to which a DHCP client computer is connected. As a way of introducing the basic functions of DHCP, this chapter presents a case study of an enterprise network that uses DHCP to automate the configuration process.

Setting Up the GSI Network

This chapter uses an internal network for Generic Startup, Inc. (GSI) as the basis for an introductory example of DHCP operation. GSI's roughly 100 employees are located in a single building. Each employee has a computer attached to the GSI network. The GSI data center includes about 20 computers that provide file storage, printing, DNS, DHCP, and related services. GSI also has a connection to the Internet that is managed by the data center staff. Figure 2.1 depicts the GSI network.

The network architect at GSI has organized the network around five network segments, all connected to each other through a single router. The router also connects the GSI network to the Internet. Four of the network segments are used by the staff desktop computers; the remaining network segment is used for all of the data center computers. The GSI network architect has obtained five Class C IP network addresses, 192.168.11.0 through 192.168.15.0, for use on the GSI network and has assigned these addresses to the network segments as shown in Figure 2.2.

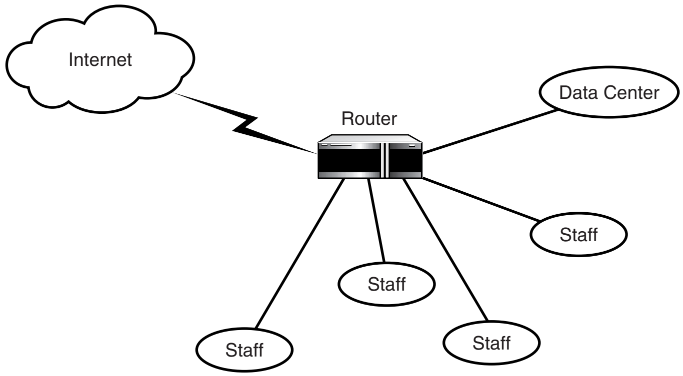

FIGURE 2.1 A diagram of the GSI network.

NOTE

The IP network addresses 192.168.11.0 through 192.168.15.0 are included in the block of Class C network addresses reserved for private use in RFC 1918 and are used here only as an example.

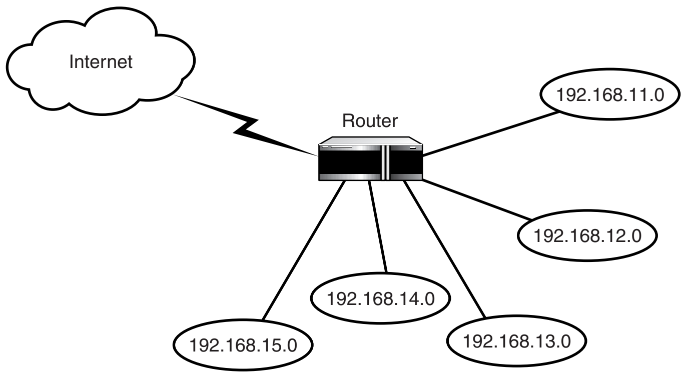

FIGURE 2.2 IP addresses in the GSI network.

So, how is DHCP used in the GSI network? There is a computer running a DHCP server on the 192.168.11.0 network. This DHCP server is configured to assign IP addresses and manage the configurations of all the GSI network segments. Computers attached to the GSI network contact the DHCP server to obtain an IP address and other configuration information. The DHCP server is configured with a description of the physical network architecture as well as the IP network address of each network segment. Using this information, the server automatically selects configuration information for each computer based on the network segment to which the computer is attached. The DHCP server assigns an IP address to each computer without requiring any manual intervention in the selection or tracking of assigned addresses.

Using DHCP to Configure Computers

The DHCP server manages the configuration of computers attached to the GSI network throughout their life cycle. This section describes the interactions between a computer and the GSI DHCP server in the following instances:

- When the new computer is first connected to the GSI network.
- When the computer is restarted.
- When the computer is moved to a new location within GSI.
- When the computer is removed from use in GSI.

As shown in Figure 2.3, the GSI DHCP server, `dhcpserve`, is connected to the data center network segment, and a new computer, `desktop1`, is attached to one of the staff network segments.

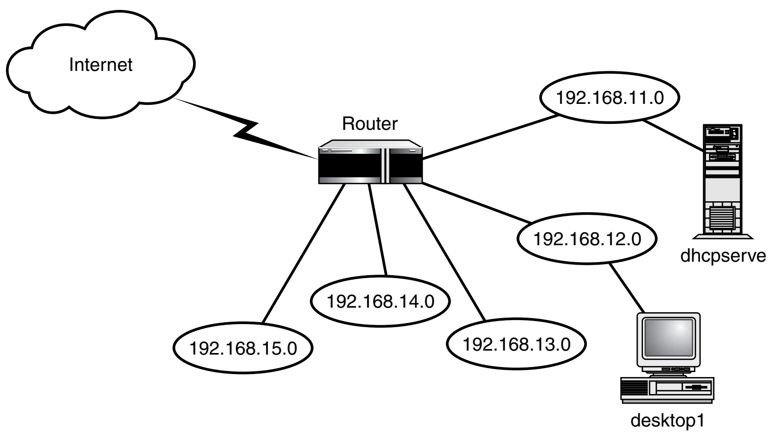

FIGURE 2.3 `desktop1` and `dhcpserve` connected to the GSI network.

Using the DHCP Server to Obtain a New IP Address

When `desktop1` is first connected to the GSI network, it needs to contact the DHCP server to obtain an IP address and other configuration parameters. To locate a server, `desktop1` broadcasts a message to locate potential DHCP servers on the GSI network.

Then, `dhcpserve` receives this broadcast and replies to `desktop1`, identifying itself to `desktop1` as a DHCP server. Because `desktop1` and `dhcpserve` are on different network segments, the router, acting as a *relay agent*, forwards messages between the two computers. Chapter 7, "Transmitting DHCP Messages," discusses relay agents in more detail. The interactions between `desktop1` and `dhcpserve` are discussed in the following sections.

After receiving the initial message from `desktop1`, `dhcpserve` selects an IP address, 192.168.12.25, which is appropriate for the 192.168.12.0 network to which `desktop1` is connected. `dhcpserve` also chooses other configuration parameters, such as the subnet mask (255.255.255.0), the address of the router interface on the 192.168.12.0 network, and the address of the GSI DNS server. `dhcpserve` uses the DHCP client configuration rules defined by the network architect and information sent by the relay agent to determine these parameters. It then returns an offer message that contains the selected address and parameters to `desktop1`.

After `desktop1` receives the offer message from `dhcpserve`, it broadcasts a message requesting the IP address and other configuration parameters from `dhcpserve`. The DHCP server confirms that the address is still available, and sends the parameters to `desktop1` in the final message of the sequence. When the message arrives, the DHCP client software on `desktop1` extracts the configuration parameters from the message, configures the client computer's IP stack to use the IP address and any other parameters it receives. As soon as the IP stack is configured, the client computer can use network. The DHCP client software also records the IP address and configuration parameters locally in a file on `desktop1` for later use.

A total of four messages are exchanged: two messages from the client and two replies from the server. This sequence of messages may seem confusing and redundant at first. Why would `desktop1` send back a request for the address and parameters that `dhcpserve` returned in the first offer? The extra message exchange makes it possible to have more than one DHCP server serving a network. Each server receives the DHCP client's initial message and sends a response. The DHCP client chooses one of these responses. Each server whose response is not chosen reclaims the IP address it offered. This issue is discussed in more detail in Chapter 8, "DHCP Message Exchanges."

Restarting `desktop1`

When `desktop1` is restarted (for example, when it is first turned on in the morning), it retrieves the IP address and configuration parameters it previously received from `dhcpserve` and attempts to reconfirm the configuration. As the next section explains, reconfirming its configuration gives `desktop1` the opportunity to determine whether that configuration is still valid. If `desktop1` is moved to a new network segment, it must get a new IP address.

For the moment, however, `desktop1` is still attached to the 192.168.12.0 network. When its confirmation message is received by `dhcpserve`, the server sends back a message indicating that `desktop1` can continue to use 192.168.12.25, and resends the other configuration parameters that the client needs. After receiving the reply message from `dhcpserve`, `desktop1` begins network activity.

Now, suppose that `desktop1` fails to contact `dhcpserve` when it restarts. This could occur if, for example, all the GSI computers are affected by a buildingwide power outage and `desktop1` has completed its restart process before `dhcpserve`. If `desktop1` receives no response to its reconfirmation message, it assumes that it is still connected to the same network segment (192.168.12.0). At this point, `desktop1` has no better information available, so it uses the IP address (192.167.12.25) and other parameters previously assigned by `dhcpserve`.

NOTE

As discussed later in this chapter, `dhcpserve` assigns an IP address to `desktop1` for a limited period of time called a *lease*. `desktop1` tries to reconfirm its address and reuse its old IP address only if that lease has not expired. If the lease has expired, `desktop1` restarts the DHCP process as though it had never been assigned an address.

Moving `desktop1` to a New Network Segment

Now, consider what happens when `desktop1` is moved to a new network segment. When `desktop1` is started up in the new location and sends out a confirmation request, `dhcpserve` determines that its old IP address is not valid for use on the new network segment. For example, if `desktop1` is moved to the segment with IP address 192.168.13.0, as Figure 2.4 illustrates, `dhcpserve` receives a confirmation request from `desktop1` to use its old address, 192.168.12.25, but it determines that the message originated on network 192.168.13.0. Because the address sent by `desktop1` is not a valid address for network 192.168.13.0, `dhcpserve` determines that `desktop1` has moved to a new segment.

When `dhcpserve` determines that `desktop1` is connected to a different network segment and that its IP address is no longer usable, `dhcpserve` sends a response to `desktop1`, denying use of its old address. After receiving the negative response, `desktop1` marks the IP address as invalid.

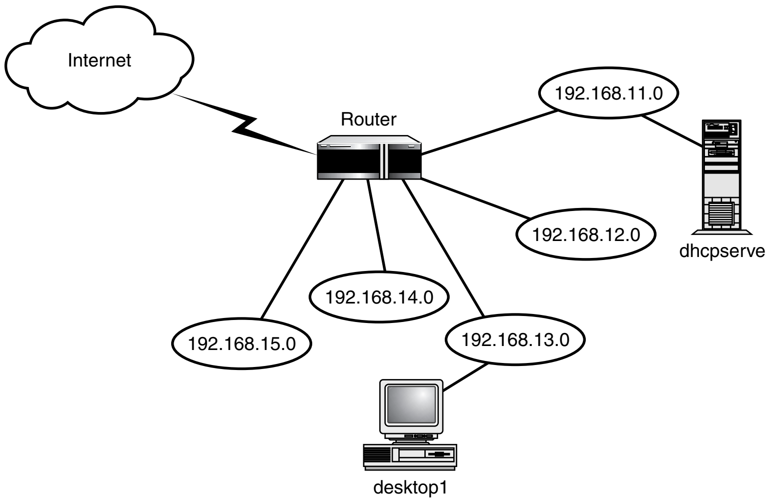

FIGURE 2.4 `desktop1` connected to the 192.168.13.0 network segment.

Of course, `desktop1` still needs an IP address from its new network segment. At this point, to obtain that address, `desktop1` uses the process described in the section "Using the DHCP Server to Obtain a New IP Address":

1. `desktop1` broadcasts a request to obtain an address from a DHCP server.
2. `dhcpserve` selects an address from the 192.168.13.0 network and sends an offer with the new address.
3. `desktop1` requests the new address.
4. `dhcpserve` confirms the requested address and returns that address to `desktop1`. In this case, `dhcpserve` selects 192.168.13.37 from the 192.168.13.0 network to allocate to `desktop1`.
5. `dhcpserve` and `desktop1` record the new address, and `desktop1` begins to use the new address immediately.

How does `dhcpserve` determine the network segment to which `desktop1` is attached? As described in Chapter 7, when `desktop1` has not yet been configured with an actual IP address, it sends its confirmation message with the source address field in the IP datagram header set to 0.0.0.0. This means that `dhcpserve` cannot deduce the source of the message from the contents of the IP datagram header.

dhcpserve deduces the source of the message from desktop1 in one of two ways:

- If desktop1 and dhcpserve are on two different network segments, a relay agent must forward the message from desktop1 to dhcpserve. The relay agent includes information about the source of the message in the gateway address field of the message from desktop1 when it forwards the message to dhcpserve. dhcpserve uses that information to determine to which segment desktop1 is currently attached.
- If dhcpserve receives a message with the gateway address field set to 0.0.0.0, it knows that the relay agent did not forward the message and therefore desktop1 and dhcpserve must be connected to the same subnet.

Retiring desktop1 from Service

As additional computers are connected to the 192.168.13.0 subnet, dhcpserve allocates addresses to these new computers. Recall that 192.168.13.0 is a Class C IP address. Thus, the 192.168.13.0 subnet has addresses for only 254 computers and, eventually, dhcpserve will run out of available addresses. DHCP allows for the reallocation to new computers of addresses that are no longer in use. As computers are removed from the 192.168.13.0 subnet, dhcpserve makes those addresses available for reallocation to new computers. As long as no more than 254 computers are connected to the 192.168.13.0 subnet at any one time, dhcpserve can allocate the available addresses to those connected computers.

NOTE

Because the addresses 0 and 255 are reserved for network broadcasts and network numbering, a Class C network address has 254 distinct IP addresses for hosts.

Eventually, of course, desktop1 must go the way of all computers and be retired from service at GSI. When desktop1 is disconnected from the GSI network for the last time, it does not send specific notification to dhcpserve that it is being decommissioned. Instead, dhcpserve simply receives no further requests for confirmation or for new addresses from desktop1. After the lease on desktop1's IP address has expired, the DHCP server simply reclaims that IP address for use by some other DHCP client. For dhcpserve, the decommissioning of desktop1 is indistinguishable from desktop1 simply not being powered on.

Leases on IP Addresses in DHCP

When dhcpserve allocates an address, it specifies the length of time over which the requesting computer may use that address. This time period is called a *lease*.

Much like the lease on an apartment, the lease on an address is an agreement between the DHCP server and the DHCP client that defines the time period during which the client will use that address. When the lease expires, the DHCP server can reallocate the address, with a new lease, to a new computer. Therefore, when the lease granted to `desktop1` for the use of its address expires, `dhcpserve` is free to reallocate the address to a new computer. In the same way the lease on an apartment can be renewed, the lease on an IP address can be renewed. This allows a DHCP client to continue to use the same address for much longer than the duration of an individual lease. This mechanism is described in more detail in Chapter 3, "Configuring the DHCP Server."

NOTE

At a many large conferences, terminal rooms with network connections for laptops are provided. Addresses are assigned to the laptops using DHCP. Because there are usually more laptops than IP addresses available on such a terminal room network, the addresses must be reassigned to newly arriving laptops as other laptops leave the network. The addresses are assigned with relatively short lease times (for example, 2 hours) so that soon after a laptop is disconnected from the terminal room network, the DHCP server can reassign that address to another laptop.

Many more issues are related to the use of address leases than are suggested by the simple example previously described. Chapter 19, "Tuning a DHCP Service," explains more about allocating leases, lease extension, and explicit lease termination.

It is important that DHCP servers manage IP addresses carefully because serious problems can be caused by allocating the same address to two different computers, and these can be difficult to debug. For example, network communication with computers assigned to the same IP address may be unreliable and IP datagrams sent to an address used by both computers may be delivered to only one or the other computer. A DHCP server must take care not to reallocate to one computer an address currently in use by another computer, as this would result in both computers using the same address.

Two Alternative Mechanisms to Leasing

A lease might seem like an overly complex mechanism for the coordination of the use of IP addresses. This section describes two alternative mechanisms:

- Explicit notification from a computer to the server that an address can be reassigned
- Dynamic probing of addresses

Both of these mechanisms were considered by the IETF DHC Working Group during the development of the DHCP specification. Although both mechanisms are useful and are, in fact, recommended for use in the DHCP specification to help detect some duplicate assignments, neither can guarantee that two or more clients will not use the same IP address.

Explicit Departure Messages

One way in which an IP address allocation server might learn that a computer has left the 192.168.13.0 subnet is for that computer to send an explicit message that informs the server that it is leaving. After receiving such a message, the server would be free to allocate the address of the departed computer to another computer.

Unfortunately, it is impossible to depend on a computer to send a message when it is disconnected from the network. Computers are not always shut down gracefully, and users may not remember to send the departure message prior to disconnecting a computer. Also, if a computer has been disconnected from the segment due to a change in the network architecture—perhaps the computer has been connected to a different hub in the wiring closet—there may be no reason to send a departure message.

Dynamic Address Probing

Another strategy that can be used to determine when an IP address is no longer in use is to probe to see if a computer is already using that address. For example, an IP address server might send an *ICMP echo request* (`ping`) to determine whether a candidate address might already be in use. If the server receives a matching *ICMP echo reply*, a computer must be using the candidate address, so the server would continue to probe for an unused address.

This strategy is impractical because it cannot reliably determine whether a specific address has been assigned to a computer. If the IP address server receives an echo reply after probing an address, that address is definitely in use. However, not receiving a reply does *not* necessarily mean that the address is not in use. A network problem could exist between the server and the computer using the candidate address, or that computer might simply be turned off. In either case, the server would not receive an echo reply, even though the address is in use.

Benefits of DHCP Leases

DHCP specifies the use of leases in address allocation so that a server can know reliably when it can reallocate an address. The lease constitutes an agreement between the client and the server; the server cannot reallocate the address until the lease has expired, and the client cannot use the address after the lease has expired. If the computer to which the address was allocated never contacts the server again, the server must wait until the lease expires before reallocating the address. Even if the computer to which the address was allocated has active network connections in place, it must terminate those connections and stop using the address as soon as the lease expires.

Summary

A computer communicates with a DHCP server in four basic ways. When the computer is first turned on, it contacts the server for a new IP address and other configuration parameters. Each time the computer is turned on after it has obtained its initial configuration information, it confirms its address and parameters with the server. As long as the computer remains connected to the same network segment, it periodically contacts the DHCP server to renew the lease on the address it is using. If the computer is moved to a new network segment, the server notifies the computer that its address is no longer valid; the computer then contacts the server to obtain a new address and configuration parameters that are appropriate to the new network segment.

A DHCP server allocates a currently unused address to a computer that is newly attached to a network segment. As more computers are attached to a network segment, all the available addresses might be assigned. When all the addresses have been allocated, a DHCP server begins to reuse previously allocated addresses that are no longer in use. DHCP uses leases on addresses to determine when an address is no longer in use. A DHCP server does not reassign an address before its lease expires, and a computer must stop using an address as soon as the lease on that address expires.

3

Configuring the DHCP Server

IN THIS CHAPTER

- Specifying the Basic Network Architecture
- Required Configuration Parameters
- Specifying Leases
- Other DHCP Options
- Extending a Lease and Moving Between Subnets
- Other Configuration Information

In managing an enterprise network, the network architect designs the network architecture and determines the configuration parameters to be assigned to hosts throughout the network. When the network architecture has been determined, the network architect must indicate the structure of the network to the DHCP server. Based on that structure, the DHCP server selects configuration parameters and appropriate addresses for DHCP clients.

The examples in this chapter are based on the GSI network architecture example and scenarios described in Chapter 2, "An Example of DHCP in Operation." The configuration files are designed for use with the ISC DHCP server and use the syntax of the ISC DHCP server configuration files.

Specifying the Basic Network Architecture

The network architect describes the network architecture to the DHCP server by identifying the IP subnets, the addresses, and the subnet masks for each of those subnets. Using this information, the DHCP server associates incoming DHCP messages with subnets in the network. Based on the subnet from which a DHCP message was received, the server selects an appropriate IP address to assign to the client or determines that a DHCP client has moved to a new subnet.

The ISC DHCP server configuration file is an ASCII text file that contains a series of declarations describing the network to be managed by the server. The server reads and parses the file when it first starts running.

Subnet Declarations

The basic subnet declaration in the ISC server configuration file follows the format in Example 3.1.

Example 3.1

```
subnet subnet-address netmask subnet-mask {
        subnet declarations
        }
```

In this subnet declaration, *subnet-address* is the IP address of the subnet, and *subnet-mask* is the subnet mask to be used with this subnet. Both *subnet-address* and *subnet-mask* are written in dotted-decimal notation.

NOTE

In the examples in this chapter, keywords are shown in **bold**, and arguments that must be supplied are shown in *italic*.

The sample network shown in Figure 3.1 is described with the partial configuration file shown in Example 3.2. The sample configuration file includes a subnet declaration for each of the five subnets, with the IP address for each subnet and the 255.255.255.0 subnet mask. Figure 3.1 shows the IP addresses in the network.

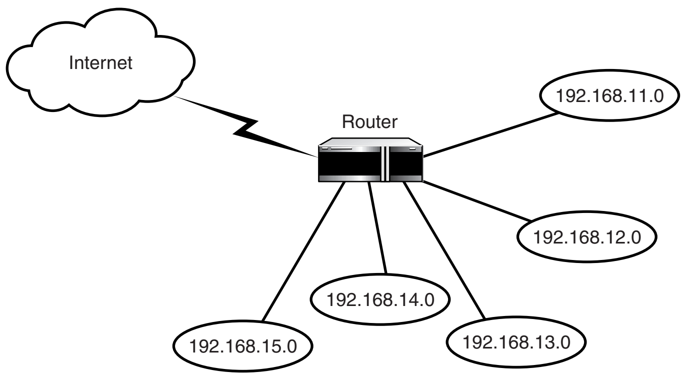

FIGURE 3.1 IP addresses in the GSI network.

You can include comments in the configuration file for the ISC server as lines that begin with the # character. Example 3.2 includes several comments that explain some of the details of the configuration file.

Example 3.2

```
# Server subnet
subnet 192.168.11.0 netmask 255.255.255.0 {
    }
# Staff subnet 1
subnet 192.168.12.0 netmask 255.255.255.0 {
    }
# Staff subnet 2
subnet 192.168.13.0 netmask 255.255.255.0 {
    }
# Staff subnet 3
subnet 192.168.14.0 netmask 255.255.255.0 {
    }
# Staff subnet 4
subnet 192.168.15.0 netmask 255.255.255.0 {
    }
```

Subnet Address Allocation

In addition to defining the subnets, the network architect must define the range of addresses within each subnet, or *scope*, that is available for allocation by the server. Any addresses assigned to hosts or devices through some other mechanism must be excluded from the range of available addresses for each subnet. For example, in the GSI network, the router interface on each subnet is assigned the host address 254. Thus, on the 192.168.11.0 subnet, the router uses address 192.168.11.254.

The network architect manually configures the router interfaces, rather than using DHCP to assign the addresses. The server is configured so that the range of available addresses on each subnet does not include the router's address.

In the ISC server configuration file, the syntax for specifying the range of available addresses in a subnet is shown in Example 3.3.

Example 3.3

```
range first-available-address last-available-address;
```

Example 3.4 gives the configuration file for the GSI network, specifying that IP addresses 1 through 251 are available on the server subnet and IP addresses 1 through 253 are available on the other subnets for assignment to DHCP clients in each subnet. This configuration file reserves host address 254 on each subnet for the router interface on that subnet. The server subnet declaration also reserves addresses for a DHCP server and a DNS server.

Example 3.4

```
# Server subnet
subnet 192.168.11.0 netmask 255.255.255.0 {
    range 192.168.11.1 192.168.11.251;
    # 192.168.11.252 reserved for DHCP server
    # 192.168.11.253 reserved for DNS server
    # 192.168.11.254 reserved for router interface
}
# Staff subnet 1
subnet 192.168.12.0 netmask 255.255.255.0 {
    range 192.168.12.1 192.168.12.253;
    # 192.168.12.254 reserved for router interface
}
# Staff subnet 2
subnet 192.168.13.0 netmask 255.255.255.0 {
    range 192.168.13.1 192.168.13.253;
    # 192.168.13.254 reserved for router interface
    }
# Staff subnet 3
subnet 192.168.14.0 netmask 255.255.255.0 {
    range 192.168.14.1 192.168.14.253;
    # 192.168.14.254 reserved for router interface
}
# Staff subnet 4
subnet 192.168.15.0 netmask 255.255.255.0 {
    range 192.168.15.1 192.168.15.253;
    # 192.168.15.254 reserved for router interface
    }
```

Required Configuration Parameters

DHCP provides other configuration parameters in addition to an IP address. In fact, several additional parameters must be provided to a TCP/IP host before that host can communicate with other hosts. A host must be configured with the following:

- Its local subnet mask
- The address of at least one router on its subnet
- The address of a DNS server

These parameters are provided to a DHCP client through *options* in the DHCP message. Chapter 9, “DHCP Options,” describes all the DHCP options in detail. A few of the most commonly used options are discussed in the following sections.

Configuration Options

The syntax for specifying an option is shown in Example 3.5.

Example 3.5

```
option option-name option-value;
```

If an `option` statement appears within a subnet declaration, it is applied to any DHCP client in that subnet. In Example 3.6, adding a routers `option` statement to the declaration for the 192.168.11.0 subnet causes the DHCP server to send 192.168.11.254 as the default router address to any DHCP client on the 192.168.11.0 subnet.

Example 3.6

```
# Server subnet
subnet 192.168.11.0 netmask 255.255.255.0 {
    range 192.168.11.1 192.168.11.251;
    # 192.168.11.252 reserved for DHCP server
    # 192.168.11.253 reserved for DNS server
    # 192.168.11.254 reserved for router interface
    option routers 192.168.11.254;
    }
```

In addition to the `routers` option, the DHCP client also needs a `subnet-mask` and a domain-name-servers option. GSI maintains a DNS server at 192.168.11.253, and the subnet mask for every GSI network is the same: 255.255.255.0. The configuration file for GSI, including default routers, subnet masks, and DNS servers, is shown in Example 3.7.

Example 3.7

```
# Server subnet
subnet 192.168.11.0 netmask 255.255.255.0 {
    range 192.168.11.1 192.168.11.251;
    # 192.168.11.252 reserved for DHCP server
    # 192.168.11.253 reserved for DNS server
    # 192.168.11.254 reserved for router interface
    option routers 192.168.11.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
    }
# Staff subnet 1
subnet 192.168.12.0 netmask 255.255.255.0 {
    range 192.168.12.1 192.168.12.253;
```

```
    # 192.168.12.254 reserved for router interface
    option routers 192.168.12.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
}
# Staff subnet 2
subnet 192.168.13.0 netmask 255.255.255.0 {
    range 192.168.13.1 192.168.13.253;
    # 192.168.13.254 reserved for router interface
    option routers 192.168.13.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
}
# Staff subnet 3
subnet 192.168.14.0 netmask 255.255.255.0 {
    range 192.168.14.1 192.168.14.253;
    # 192.168.14.254 reserved for router interface
    option routers 192.168.14.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
}
# Staff subnet 4
subnet 192.168.15.0 netmask 255.255.255.0 {
    range 192.168.15.1 192.168.15.253;
    # 192.168.15.254 reserved for router interface
    option routers 192.168.15.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
}
```

Specifying Leases

Chapter 2 describes the use of leases as a mechanism through which a DHCP server knows when a host will stop using an IP address. The DHCP specification allows a lease to be up to 2^{32}–2 seconds (49,710 days, or about 135 years).

NOTE

2^{32}–2 is expressed as $FFFFFFFE_{16}$. This is the largest number that can be stored in the 32-bit lease field in a DHCP message. DHCP uses $FFFFFFFF_{16}$ to represent an infinite lease.

Lease Durations

The DHCP specification does not include rules or requirements for lease allocation or duration; those policies are defined by the network architect and are implemented by the DHCP server. A DHCP client might request a particular lease duration, but the DHCP server will always choose a lease duration based on the policies for lease assignment defined by the network architect.

Default, Minimum, and Maximum Lease Lengths

The ISC server allows the network architect to specify a *default lease length*, a *minimum lease length*, and a *maximum lease length*. The default lease length is used if the client does not request a specific lease. The minimum lease length is used to force the client to take a longer lease than it has requested. The maximum lease length defines the longest lease that the server can allocate. If a client requests a lease longer than the maximum lease length, then the server simply issues a lease equal to the maximum lease length.

The syntax for defining the default, minimum and maximum lease lengths is shown in Example 3.8, and `time` is expressed in seconds.

Example 3.8

```
default-lease-time time;
max-lease-time time;
min-lease-time time;
```

Subnet Lease Lengths

Choosing appropriate lease times for a subnet depends on the types of hosts that will connect to that subnet. Table 3.1 lists some types of subnets that might be a part of the GSI network and examples of lease times.

TABLE 3.1 Examples of Lease Times

Type of Subnet	Primary Use	Default Lease
GSI training lab	Students with laptops	One class period plus 10 minutes
Conference room	Visitors with laptops	2 hours
"Hotel" office	Staff members who use it daily	12 hours
Telecommuters	DSL/cable service	7 days
Staff offices	Permanent staff members	30 days
Central servers	Organization servers	3 months

NOTE

Hoteling means the temporary use of offices by staff on a daily basis. Each "hotel" office is equipped with a wall jack through which a laptop is connected to the GSI network. Because a different person may use these offices each day, the network must accommodate dynamic allocation of an IP address to the computer or computers in those offices on a daily basis.

Chapter 19, "Tuning a DHCP Service," includes a more detailed discussion of lease times for specific scenarios.

GSI Subnet Leases

In the GSI network, the 192.168.11.0 subnet is used for servers, which have a default lease of 90 days. The 192.168.12.0, 192.168.13.0, and 192.168.14.0 subnets are used for staff offices, and the computers connected to those subnets have a default lease of 30 days. The remaining subnet, 192.168.15.0, is used for hoteling, and computers connected to that subnet have a default lease of 12 hours.

The configuration file for these lease times is shown in Example 3.9.

Example 3.9

```
# Server subnet
subnet 192.168.11.0 netmask 255.255.255.0 {
    range 192.168.11.1 192.168.11.251;
    # 192.168.11.252 reserved for DHCP server
    # 192.168.11.253 reserved for DNS server
    # 192.168.11.254 reserved for router interface
    option routers 192.168.11.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
    # default lease = 90 days, max lease = 120 days
    default-lease-time 7776000;
    max-lease-time 10368000;
    }
# Staff subnet 1
subnet 192.168.12.0 netmask 255.255.255.0 {
    range 192.168.12.1 192.168.12.253;
    # 192.168.12.254 reserved for router interface
    option routers 192.168.12.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
    # default lease = 30 days, max lease = 45 days
    default-lease-time 2592000;
    max-lease-time 3888000;
```

```
}
# Staff subnet 2
subnet 192.168.13.0 netmask 255.255.255.0 {
    range 192.168.13.1 192.168.13.253;
    # 192.168.13.254 reserved for router interface
    option routers 192.168.13.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
    # default lease = 30 days, max lease = 45 days
    default-lease-time 2592000;
    max-lease-time 3888000;
}
# Staff subnet 3
subnet 192.168.14.0 netmask 255.255.255.0 {
    range 192.168.14.1 192.168.14.253;
    # 192.168.14.254 reserved for router interface
    option routers 192.168.14.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
    # default lease = 30 days, max lease = 45 days
    default-lease-time 2592000;
    max-lease-time 3888000;
}
# Staff subnet
subnet 192.168.15.0 netmask 255.255.255.0 {
    range 192.168.15.1 192.168.15.253;
    # 192.168.15.254 reserved for router interface
    option routers 192.168.15.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
    # default lease = 12 hrs, max lease = 24 hrs
    default-lease-time 43200;
    max-lease-time 86400;
}
```

Other DHCP Options

The configuration file in Example 3.9 specifies the configuration parameters that the network architect must define to the DHCP server. Many other options (as specified in RFC 2132) can be provided to a DHCP client. Some of these options, such as `domain name`, are widely used, and others, such as `Impress server`, are rarely used.

NOTE

The `Impress server` option specifies a list of Imagen Impress servers that the DHCP client can use. (Imagen Impress is a type of networked printer that is no longer manufactured.) This option was originally defined as a BOOTP (in RFC 1048 and RFC 951) vendor extension and is included as an option in DHCP for backward compatibility.

However, all these DHCP options are specified in the configuration file for an ISC DHCP server, using the syntax illustrated in Example 3.10.

Example 3.10

```
option option-name option-value;
```

Appendix B, "ISC DHCP Server Configuration File Reference," includes a complete list of options that can be specified to the ISC server.

Subnet Options

Some options should apply to all subnets, and others are specific to certain subnets. In the GSI network, all hosts use the same DNS server, but each subnet uses a different default router.

In an ISC DHCP server configuration file, global options are defined at the beginning of the file. These global options apply to each defined subnet so that the definitions need not be repeated in each subnet definition. The configuration file in Example 3.11 specifies that `dns.genericstartup.com` should be used by all the hosts in the GSI network and that those hosts should use `genericstartup.com` as their DNS domain. The default routers are specified for each subnet.

Example 3.11

```
option domain-name-servers "dns1.genericstartup.com", "dns2.genericstartup.com";
option domain-name "genericstartup.com";
# default lease = 30 days, max lease = 45 days
default-lease-time 2592000;
max-lease-time 3888000;
# Server subnet
subnet 192.168.11.0 netmask 255.255.255.0 {
    range 192.168.11.1 192.168.11.251;
    # 192.168.11.252 reserved for DHCP server
    # 192.168.11.253 reserved for DNS server
    # 192.168.11.254 reserved for router interface
    option routers 192.168.11.254;
    option subnet-mask 255.255.255.0;
```

```
    # default lease = 90 days, max lease = 120 days
    default-lease-time 7776000;
    max-lease-time 10368000;
}
# Staff subnet 1
subnet 192.168.12.0 netmask 255.255.255.0 {
    range 192.168.12.1 192.168.12.253;
    # 192.168.12.254 reserved for router interface
    option routers 192.168.12.254;
    option subnet-mask 255.255.255.0;
}
# Staff subnet 2
subnet 192.168.13.0 netmask 255.255.255.0 {
    range 192.168.13.1 192.168.13.253;
    # 192.168.13.254 reserved for router interface
    option routers 192.168.13.254;
    option subnet-mask 255.255.255.0;
    }
# Staff subnet 3
subnet 192.168.14.0 netmask 255.255.255.0 {
    range 192.168.14.1 192.168.14.253;
    # 192.168.14.254 reserved for router interface
    option routers 192.168.14.254;
    option subnet-mask 255.255.255.0;
    }
# Staff subnet 4
subnet 192.168.15.0 netmask 255.255.255.0 {
    range 192.168.15.1 192.168.15.253;
    # 192.168.15.254 reserved for router interface
    option routers 192.168.15.254;
    option subnet-mask 255.255.255.0;
    # default lease = 12 hrs, max-lease-time = 24 hrs
    default-lease-time 43200;
    max-lease-time 86400;
 }
```

Global Values for Options

Example 3.11 demonstrates that options can be specified as DNS names as well as IP addresses. The ISC server resolves any domain names in the configuration file and uses the corresponding IP address as the value for the associated option. If a domain name corresponds to more than one IP address, and the option allows more than one IP address to be sent, all the IP addresses for that domain name are sent.

Example 3.11 illustrates that global values for options can be overridden with new values for specific subnets. The global values for default lease time and maximum lease time values are set to 30 days and 45 days, respectively; those values are set to 90 days and 120 days, respectively, within the server subnet declaration.

Extending a Lease and Moving Between Subnets

Using the configuration file in Example 3.11, let's take a closer look at some of the examples from Chapter 2. The section "Using DHCP to Configure Computers" describes the steps in configuring a GSI client, `desktop1`, when it is first connected to the 192.168.12.0 subnet of the GSI network. When the DHCP server, `dhcpserve`, receives the initial broadcast message from `desktop1`, the server determines that `desktop1` is connected to the 192.168.12.0 subnet. From the configuration file, the DHCP server determines that `desktop1` should receive an address in the range 192.168.12.1 to 192.168.12.253. To accompany that selected address, the DHCP server selects the subnet-specific values for the subnet mask and default router, and it selects global values for the domain name and domain name servers. Because `desktop1` did not request a specific lease time, the DHCP server chooses a lease time of 30 days, and the server returns the parameters in Table 3.2 to `desktop1`. At the same time, the server records the information about the allocated address and lease time on disk.

TABLE 3.2 Initial Parameters for `desktop1`

Option Name	Option Value
IP address	192.168.12.25
Subnet mask	255.255.255.0
Default router	192.168.12.254
Domain name	genericstartup.com
Domain name servers	dns1.genericstartup.com, dns2.genericstartup.com
Lease time	30 days (2,592,000 seconds)

Extending a GSI Lease

The section "Restarting `desktop1`" in Chapter 2 describes the sequence of events that occur when `desktop1` restarts while it is still connected to the 192.168.12.0 subnet. In that situation, `desktop1` broadcasts a DHCP message to confirm its address, 192.168.12.25. The DHCP server on `dhcpserve` receives the confirmation request, and it consults its configuration and lease data. Based on the entry for the 192.168.12.0 subnet, the server confirms that 192.168.12.25 is a valid address for the network segment to which `desktop1` is currently connected. The server then consults the lease data and confirms that `desktop1` has a valid lease on 192.168.12.25. Having established the validity of the requested address, the DHCP server extends the lease on the address to the default value of 30 days and returns an acknowledgment. After receiving the acknowledgment, `desktop1` records the new lease time and begins to use its old address again.

Moving Between GSI Subnets

Suppose, now, that `desktop1` is re-located to a new subnet, as described in the section "Moving `desktop1` to a New Network Segment" in Chapter 2. As in the previous example, `desktop1` first broadcasts a message to confirm its address. But in this case, the DHCP server determines that 192.168.12.25 is not a valid address for the segment to which it is attached. The server examines the address recorded in the DHCP message by the relay agent, 192.168.13.254, and identifies the 192.168.13.0 subnet as the source of the message. Because 192.168.12.25 is not in the range of addresses on that subnet, the server returns a message denying `desktop1` the use of the requested address.

After receiving the negative reply from the DHCP server, `desktop1` restarts the DHCP process as it would if it had no valid address. It broadcasts an initial message to locate `dhcpserve`, which allocates an IP address, 192.168.13.37, from the range of addresses available on the new network to which `desktop1` is now attached. `desktop1` records the new address and begins using the network with its newly assigned address.

Other Configuration Information

The file in Example 3.11 is a complete, although minimal, configuration file for the GSI network. In practice, the configuration file would likely include additional information, such as global parameters for other GSI servers and statically assigned addresses, as described in Chapter 15, "Configuring a DHCP Server."

Summary

Network architects can use DHCP to automate the management of IP host configuration. The network architect's role is to configure the DHCP server with a description of the network architecture and rules for IP host configuration. The DHCP server then uses that configuration information to determine specific configuration parameters for each host. The network architect describes the architecture and host configuration rules to the server through a configuration mechanism, such as a configuration file or an interactive user interface.

The ISC DHCP server uses a configuration file that is read when the server first starts up. The network architect defines global and network-specific configuration parameters within the configuration file. As the ISC server receives DHCP messages, it consults the network description from the configuration file and determines the specific parameters to be passed to the IP host. This chapter introduces some of the most widely used features of the ISC server configuration file. Chapter 9 describes the DHCP options in more detail, and Chapter 15 elaborates on the format and use of the configuration file.

4

Configuring TCP/IP Stacks

IN THIS CHAPTER

Correctly operating and maintaining a network that uses DHCP requires an understanding of the fundamentals of TCP/IP networking. This chapter provides an overview of the TCP/IP suite and a summary of the parts of the TCP and IP protocols that are relevant to DHCP. If you are experienced with the use and design of TCP/IP, you might want to skim this chapter. On the other hand, if you want a comprehensive review of TCP/IP, you may want to consider reading the *Internetworking with TCP/IP* series (Comer, 2000) or *TCP/IP Illustrated* (Stevens, 1994).

The sections of this chapter examine each of the layers of the TCP/IP suite and describe some of the characteristics and features of each layer individually. One of the important functions of DHCP is to transmit TCP/IP software configuration parameters, and this chapter identifies some of those parameters.

The TCP/IP Protocol Suite

The TCP/IP protocol suite is a collection of related computer communication protocols that, when used together, provide network communication services among applications. TCP/IP is the protocol suite used on the Internet. Because TCP/IP is vendor-independent and can be used on many hardware/software platforms, it has become the most widely used protocol suite on corporate intranets.

TCP/IP is designed around a five-layer model, as shown in Figure 4.1. Together, the hardware and software that make up these layers form the TCP/IP protocol *stack*. Each layer includes configuration parameters that control the functions of the protocols in that layer. Many of these configuration parameters have default values that allow the protocol to function in most cases. Other parameters, such as the IP address and subnet mask, must be set to specific values for each computer and have no valid default values. In any case, even one incorrectly set parameter could cause the entire stack to perform poorly, experience intermittent failures, or simply not work at all.

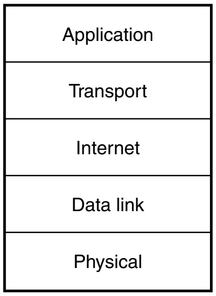

FIGURE 4.1 The five layers of the TCP/IP model.

The Physical Layer

The *physical layer* delivers data encoded in a physical representation such as electrical current, radio waves, or light. The data is carried through a medium such as copper wire or optical fiber. Some physical-layer implementations are point-to-point connections, where two network nodes are connected across a single connection. Other physical-layer implementations are broadcast networks, where many network nodes are connected to the same physical medium and each network node sees all messages that are transmitted on the network.

The Data Link Layer

The *data link layer* of the TCP/IP protocol suite is responsible for delivering messages between network nodes that are attached to the same network segment, or *link*. When one network node wants to send a message to another network node attached to the same network segment, it puts the message into a frame. The frame is used to attach some information to the message. Each frame at the link layer has at least a protocol identifier. This identifier tells the recipient how it should interpret the message. In the case of TCP/IP, the identifier indicates that the message in the frame is an IP packet.

Link-layer framing can be different for different physical-layer types. Point-to-point links, such as modem connections, typically use Point to Point Protocol (PPP).

Broadcast links, such as traditional ethernet, typically use ethernet framing or IEEE 802.3 framing. (It is beyond the scope of this book to describe these different framing standards in detail.) On a point-to-point network, there are only two nodes on the link—the two endpoints. When one of the two devices receives a message, it knows the message came from the other device, and it knows that it is the intended recipient of the message.

Each node on a broadcast network segment must have a unique *link-layer address* (also referred to as a *hardware address* or *MAC address*). On a broadcast link, every node receives every message that is transmitted on the link, and no node knows what other nodes are connected to the link, so each frame must contain both a source and destination link-layer address. The source address is the link-layer address of the node that is sending the message. The destination address can be either the link-layer address of the node that is the intended recipient of the message, a multicast address, or a broadcast address.

When a node on a broadcast link receives a message, it checks the destination link-layer address. If the destination address is its own address, or if it is the broadcast address, or if it is a multicast address for a multicast group to which the node is subscribed, the node accepts the packet; otherwise, it ignores the address. This is usually done on the NIC.

NOTE

Many physical link layers that people think of as broadcast networks are actually point-to-point networks that simulate broadcast networks, using link-layer framing. For example, a twisted-pair ethernet network is actually a large collection of point-to-point links. Each node is connected by a pair of wires (sometimes two pairs) or an optical fiber to a hub or switch. A *hub* forwards every packet it receive to every node that is connected to it. A *switch* keeps a list of all the nodes it has heard from and the ports to which they are attached, and when it receives a message for a particular node, it forwards that message to the port to which the recipient is connected. If it has never heard from the intended recipient, it ignores the message. This allows a twisted-pair ethernet network to look just like a classic ethernet network at the link layer, even though at the physical layer it's very different.

Address Translation

In order for an IP node to send an IP packet to another IP node on the same link, it must translate the destination IP address into a link-layer address and then put the IP packet into a link-layer frame and transmit that packet to the recipient's link-layer address. This translation can be accomplished in several ways, including through static translation tables and dynamic mechanisms that obtain the hardware address through the network itself.

An IP node sending an IP packet on an ethernet network uses a combination of static and dynamic translation. If the IP destination address is an IP broadcast address on the link to which the node is attached, it translates this to the ethernet all-stations address by using a static translation. Otherwise, it uses Address Resolution Protocol (ARP) to dynamically determine the link-layer address of the recipient.

With ARP, a node sends a datagram to the ethernet all-stations address that makes the request "please tell me the link-layer address that corresponds to this IP address." Every node on the link receives this request. If a node on the link has the IP address mentioned in the request, it replies with its link-layer address. The sender uses this link-layer address to send IP datagrams to the destination. The sender records the destination hardware address in a local cache for future use. The entries in this cache must be deleted periodically, to ensure that the hardware addresses are updated. The lifetime of an ARP cache entry can be specified through DHCP.

The Internet Layer

The *internet layer* of the TCP/IP protocol suite is responsible for end-to-end delivery of protocol messages between computers. *Internet Protocol* (*IP*) is the protocol from the TCP/IP suite that implements the internet layer. Logically, an internet can be thought of as a collection of independent network segments or physical networks, interconnected by *routers* that are attached to two or more network segments. *Hosts* connected to the network segments communicate by exchanging network messages through the interconnecting routers.

The *IP datagram* is the basic message unit for data delivered by IP; a host forms an IP datagram and identifies the destination to which the datagram should be delivered, and the IP software on the host and routers cooperates to deliver the datagram to the destination.

Network (IP) Addresses

Each computer that uses IP is assigned an *IP address*, which performs two functions:

- It uniquely identifies the computer.
- It specifies the network segment to which the computer is connected.

In contrast, link-layer addresses provide unique identification only and give no information about location.

As Figure 4.2 shows, an IP address includes a *network number* and a *host identifier*. As a special case, each network segment in an internet has its own network address, with a unique network number and a host identifier of zero. The IP address for a computer is composed of the network number from the network segment to which it is attached and a host part that is unique among all the computers on that segment.

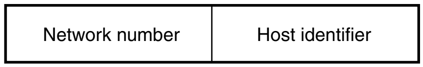

FIGURE 4.2 The structure of an IP address.

Composing the IP address from a network number and a host identifier produces some interesting features:

- It means IP datagrams can be forwarded to their destination by examining only the network number and not the entire IP address. The amount of information needed to forward datagrams depends on the number of network segments in the internet—not on the number of attached computers.
- A computer's IP address depends on the network segment to which it is attached. Thus, when a computer is first attached to an internet, it must be given an address from the network segment to which it is attached. If that computer is then moved to a different network segment, it must be given a new IP address.
- The host identifier of a computer's IP address must be unique among all the computers on its network segment. If two computers from the same network use the same host identifier, they both use the same IP address, and their network connections will be unreliable or unusable.

The related problems of correctly configuring a computer with an address that depends on its location within the network and avoiding simultaneous use of the same address by different computers were the initial motivation for DHCP. Prior to the development of DHCP, a network administrator generally had to walk around with a slip of paper or a spreadsheet listing available addresses. To add a new computer to the network without DHCP, he or she had to consult the list of available addresses, select an unused address, carefully mark the address as "in use," and manually configure the computer with that address.

Not only is this procedure time-consuming and error-prone, it doesn't scale well. Imagine the potential for conflict in a large organization where multiple network administrators try to assign IP addresses simultaneously from the same spreadsheet or sheet of paper.

Subnetting

The original IP specification (RFC 790) defines three major *address classes*, which identify the ways in which an IP address is split into a network number and a host identifier. The format of each address class determines the number of networks available in that class and the number of hosts that can be attached to each network. RFC 988 defines an additional address class, D, which is used for multicast (described later in this chapter). The format of the four address classes and the number of networks and hosts for each class are shown in Figure 4.3.

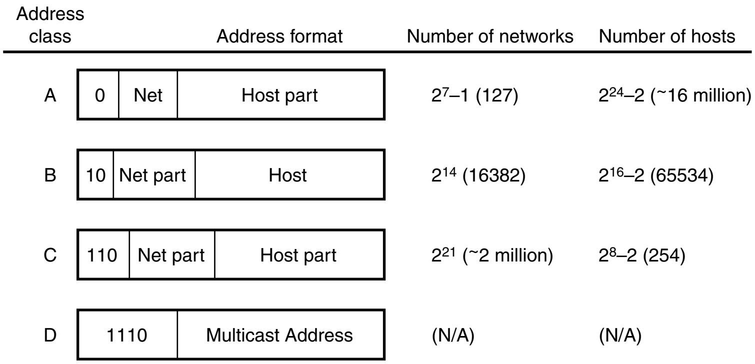

FIGURE 4.3 The structure of Class A, B, C, and D addresses.

The IP address classes aren't a good match with some network architectures. For example, if a network segment with 400 computers uses a Class B address that can accommodate more than 65,000 computers, the segment wastes more than 99% of the available addresses.

Subnetting (as described in RFC 950, RFC 1878, RFC 1519) is a technique for dividing Class A, B, or C addresses into smaller groups of IP addresses that more closely match the addressing requirements for network segments. In a subnetted network, the part of an IP address to the right of the network number is split into a subnet address and the host identifier. The subnet address locates the network segment within the collection of segments that share the same network number, and the host identifier identifies the specific host on that network segment. Figure 4.4 illustrates the format of a subnetted IP address.

In the original IP addressing scheme, the division between the network address and the host identifier is defined by the first three bits of an IP address. How is the format of a subnetted IP address defined? Each subnetted address must have an associated *subnet mask*, which identifies which bits of the address make up the subnet address and which are used as the host address. A subnet mask is a 32-bit number with a 1 in every bit from the IP address that is to be used as the network number or the subnet address and a 0 in every bit from the IP address that is to be used as the host identifier.

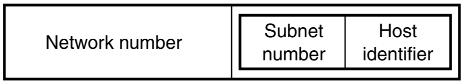

FIGURE 4.4 The structure of a subnetted IP address.

> **NOTE**
>
> The subnet bits in a subnet mask are contiguous and include both the network number field and the subnet field. The notation /24 indicates that the most significant 24 bits of the subnet mask are set to 1.

The subnet mask is applied to IP addresses to determine whether two IP addresses are part of the same subnet or different subnets. The subnet address can be extracted from an IP address by computing the bitwise logical AND of the address and the subnet mask. Two IP addresses are part of the same subnet if the resulting subnet addresses are equal.

As discussed in the next section, computers that use IP as part of the IP datagram delivery process use this equality test. The subnet mask is, therefore, just as important to the correct operation of IP as the IP address; if a computer has an incorrect subnet mask, it might be unable to correctly process some IP datagrams.

The use of a subnet mask has been extended in classless internet domain routing (CIDR) addressing, as described in RFC 1519. CIDR addressing allows the use of a subnet mask that is shorter than the network number field in the associated address. A CIDR address identifies a block of network addresses; for example, the CIDR address 192.168.4.0/22 refers to the four class C network addresses 192.168.4.0 through 192.168.7.0.

Datagram Delivery

Hosts and routers cooperate to deliver datagrams from the sending host to the destination host. The original sender of a datagram and each router along the datagram's path to its destination examine the datagram to determine its destination and then forward the datagram to the next router on the way to the destination. The sender must make an initial decision about delivery: Is the destination of this datagram on the same network segment? If so, the source can deliver the datagram directly to the destination. Otherwise, the source must deliver the datagram to a router for forwarding to the destination.

The sender determines whether the destination is on the same network segment by applying its subnet mask to both its own address and the destination address. If the resulting IP addresses match, the destination is on the same network segment. Otherwise, the source must forward the datagram to a router. Thus, a computer that uses TCP/IP must be configured with the address of at least one router in order to deliver datagrams to destinations on different network segments.

A computer uses a *routing table* that contains the addresses of routers to be used for specific destinations. Often, a computer has one or more entries in its routing table and uses these entries if no specific router for a destination exists. These entries, called *default routes*, identify *default routers* and must be inserted into the routing table manually or automatically through DHCP or by another mechanism such as *router discovery* (RFC 1256).

Multiple IP Networks on a Network Segment

Although the discussion of IP addressing and datagram delivery in the previous sections describes a network segment as having a single IP network number, it is possible to assign multiple IP networks to a single physical network segment. These IP networks, sometimes referred to as *shared* or *overlay* segments, function as though they are assigned to independent network segments. Datagrams that are sent from a computer on one IP network to a computer on the other network on the same segment must be forwarded through a router, even though both computers are on the same physical network. Routers attached to the network segment have separate routing table entries for each IP network, each of which points to the same network interface.

Assigning multiple IP networks to a single network segment complicates the management of DHCP servers because the server must know which IP networks are associated with a common network segment. In addition, the network architect may have rules about which IP network on a network segment a computer should use, and the server must follow those rules.

Multicast

IP includes a form of datagram delivery called *multicast*, in which a datagram is delivered to more than one destination. Multicast differs from broadcast in that a multicast datagram may be delivered to a subset of the computers on an internet, rather than to all the computers on a single IP network.

Multicast is used in applications in which many hosts want to receive copies of datagrams sent by a source without the overhead of sending a separate copy of the data to each destination. Examples of applications that use multicast include digitized audio/video conferencing, other collaborative applications, and routing protocols.

Other Internet-Layer Parameters

Several other IP functions have operational parameters that must be correctly configured.

TTL

Each datagram has a *time to live* (*TTL*) field that is used to detect situations such as a datagram caught in a routing loop. The TTL may be explicitly specified to the IP software. If it is not, the IP software uses a default value for the TTL field in datagrams. The default value can be changed to accommodate, for example, a larger internet.

MTU

Another IP function that can be configured is the *maximum transfer unit* (*MTU*), which is used for frames transmitted on the network segment to which a computer is attached. The MTU is the largest data payload that can be carried in a frame on the local network. Usually, the MTU can be determined from the network interface and from the type of hardware used in the local network. In some circumstances—for example, in the case of a bridged network with dissimilar hardware technologies—computers might need to use a smaller MTU than is allowed by the local hardware. This MTU value can be configured in the IP software.

PMTU

The *path maximum transfer unit* (*PMTU*) is used to control the size of segments transmitted by TCP. The PMTU value is selected to avoid fragmentation of datagrams carrying TCP segments. The IP software determines the PMTU value dynamically by monitoring reports of datagram fragmentation. When the IP software detects that datagrams are being fragmented, it reduces the PMTU value until fragmentation no longer occurs. To ensure that the largest possible PMTU value is discovered, the IP software periodically probes with larger datagrams and watches for fragmentation. The details of the PMTU mechanism are controlled through configuration parameters in the IP software.

Summary of IP Software Parameters

Table 4.1 lists all the parameters that can be configured in the IP software. This list was derived from several sources, including the Host Requirements documents (RFC 1122, RFC 1123, and RFC 1127) and the protocol specifications for PMTU discovery (RFC 1191) and router discovery (RFC 1256).

NOTE

The Host Requirements documents (RFC 1122, RFC 1123, and RFC 1127) summarize and analyze all the protocols in the TCP/IP suite. They include clarifications to protocol specifications from other RFCs, “best practices” and hints to implementers, summaries of important information about protocols, and interactions among protocols. Anyone implementing stacks, evaluating products, building applications, or otherwise using TCP/IP in nontrivial ways should read the Host Requirements documents.

TABLE 4.1 Configurable IP Parameters

Parameter Name	Values
IP Parameters, Per Host	
Be a router	on/off
Nonlocal source routing	on/off
Policy filters for nonlocal source routing	a list of filters
Maximum datagram reassembly size	an integer
Default TTL	an integer
PMTU aging timeout	an integer
MTU plateau table	a list of integers
IP Parameters, Per Interface	
IP address	a 32-bit integer
Subnet mask	a 32-bit integer
MTU	an integer
All-subnets-MTU	on/off
Broadcast address type	0.0.0.0/255.255.255.255
Perform mask discovery	on/off
Be a mask supplier	on/off
Perform router discovery	on/off
Router solicitation address	an IP address
Default routers	a list of IP addresses
Static routes, list of:	
destination	an IP address
destination mask	an IP address
type-of-service	an integer
first-hop router	an IP address
Ignore redirects	on/off
PMTU	an integer
Perform PMTU discovery	an integer

The Transport Layer

The two protocols in the *transport layer* of the TCP/IP protocol suite deliver data between applications through an internet, using the IP datagram delivery service:

- *User Datagram Protocol* (*UDP*) provides best-effort, connectionless delivery of discrete messages.
- *Transmission Control Protocol* (*TCP*) provides reliable, connection-oriented delivery of arbitrarily long messages or streams of data.

UDP

UDP provides independent delivery of individual messages. The UDP software accepts outgoing messages from the application layer and adds a header containing the destination port, the source port, and a checksum. The UDP software then passes the message to the IP software, along with the IP address of the destination. The IP software delivers the datagram to the destination computer. At the destination, the UDP software uses the destination port number to deliver the datagram to the receiving program.

Because UDP does not guarantee reliable delivery, the destination sends no acknowledgments or other indications of successful receipt of UDP messages. Reliability in applications that use UDP is the responsibility of application protocols. For example, the *Trivial File Transfer Protocol* (*TFTP*), which uses UDP, employs a *send-and-wait* mechanism in which the receiver sends a reply message to the sender to acknowledge receipt of each message from the sender.

DHCP uses UDP to deliver protocol messages between clients and hosts. UDP has two specific features that are used by DHCP:

- UDP messages can be broadcast and delivered to every computer on a network segment rather than to just a single destination computer.
- UDP messages can be transmitted with a source IP address of 0.0.0.0 if the source computer has not yet been assigned an IP address.

These two features, as described in more detail in Chapter 7, “Transmitting DHCP Messages,” allow a computer to use UDP to locate and communicate with a DHCP server before the computer has an IP address.

TCP

In contrast to the best-effort message delivery provided by UDP, TCP uses acknowledgments and retransmission to provide reliable delivery of data that is guaranteed to be correct and in the correct order, without loss or duplication. An application hands outgoing data to the TCP software, which splits the data into *segments*. TCP then uses IP to deliver these segments to the destination. The TCP software at the destination reconstructs the original message from the individual segments and passes the data to the receiving application in the correct order.

A TCP receiver sends acknowledgment messages to inform the sender what data has been successfully delivered to the receiver. If the sender fails to receive an acknowledgment, it retransmits the data until the data is successfully delivered and the sender receives the acknowledgment.

TCP also includes a form of *flow control* that enables a receiver to slow down the rate at which the sender transmits data. The receiver defines a *receive window*, and the sender transmits data only within that window. The receiver informs the sender of the size of its receive window through a field in the TCP header. If the receiver wants to pause the transmission without shutting down the connection, it sets the receive window to zero. Later, to resume transmission, the receiver sets the receive window to a nonzero value.

The Application Layer

The *application layer* of the TCP/IP protocol suite includes specific protocols that define interactions between application programs. These application protocols use the transport layer to exchange messages, which are formatted according to the rules of the application protocol. Most application protocols are specific to an application; only a few application protocols are shared among different applications.

Some protocols that are required or crucial to the operation of a TCP/IP intranet are application protocols. The *Domain Name System* (*DNS*), through which human-friendly names such as `desktop1.genericstartup.com` are translated into IP addresses, uses an application protocol that is carried by either UDP or TCP. As mentioned earlier in this chapter, in the section "UDP," TFTP, which is used to transmit software to diskless systems, is an application protocol that uses UDP. Routing protocols, such as *Routing Information Protocol* (*RIP*) and *Open Shortest Path First* (*OSPF*), are carried in UDP and TCP, respectively.

The Client/Server Model

Most protocols in the application layer are based on the *client/server model*, in which client applications contact a server to perform application-specific functions. The server application starts first, and it waits for incoming messages from client applications. Clients then send messages to the server with requests for some function or data. DHCP is based on this client/server model.

In a client/server system, the client is configured with the IP address of the server and contacts the server through that address when required by the application. A DHCP server can inform a DHCP client of the addresses of servers for many different application protocols, including DNS, *Network Time Protocol* (*NTP*), *Simple Mail Transport Protocol* (*SMTP*), *Post Office Protocol* (*POP*), and *Network News Transport Protocol* (*NNTP*).

Summary

The TCP/IP protocol suite is the basis for network communications on the Internet. TCP/IP includes several layers of protocols, each of which implements specific services and which, taken together, provide communication between application programs. Two major styles of communication exist: best-effort datagram delivery through UDP and reliable, data-stream delivery through TCP. Both TCP and UDP depend on IP for end-to-end message delivery through an internet.

To accommodate the widest possible variety of network hardware and computer systems, the protocols in the TCP/IP suite have parameters that can be configured. Some of these parameters are required for correct operation of the protocols, and others enhance the performance of applications that use TCP/IP. Configuration of the most fundamental of these parameters, the IP address, is crucial because incorrect assignment to one computer could affect the operation of other computers as well.

PART II

DHCP Theory of Operation

IN THIS PART

5

DHCP Client/Server Model

IN THIS CHAPTER

Expanding on the specific examples of DHCP in operation given in the previous chapters, this chapter summarizes DHCP theory and principles and explains why some DHCP features are included in the specification in their current form. This chapter describes the following:

- Some of the goals and constraints within which DHCP was designed
- The relationship between DHCP and other, related protocols
- The client/server architecture DHCP uses

DHCP Goals and Design Decisions

As earlier chapters explain, DHCP's primary goal is to automatically configure networked computers that use TCP/IP. Because TCP/IP is used in such diverse environments, the TCP/IP stack includes several parameters that can be configured for a specific computer and network (see RFC 1122). After using DHCP to obtain this TCP/IP stack configuration information, a networked computer can exchange packets with other computers on the network.

Administrative Control, Correctness, and Reliability

A network administrator can use DHCP to control the configuration of individual computers according to a set of particular requirements. DHCP provides a way to convey configuration parameters to the computers being managed without dictating policies about how those computers should be configured.

For example, a DHCP server is not required to respond to every DHCP message it receives. If the network policy does not enable automatic configuration of computers not previously connected to the network, the network administrator can simply configure the DHCP server to ignore `DHCPDISCOVER` messages from hosts that are not already known to the DHCP server. This design goal enables the network administrator to use DHCP in a variety of situations and to implement a range of IP address and configuration policies. A DHCP server acts as an "administrative assistant," interpreting the configuration policies the network administrator develops and passing network along specific parameters to individual computers, based on those policies.

Assigning IP Addresses Dynamically

The most obvious policy a network administrator develops for a DHCP server is one that controls the assignment of IP addresses to computers. Dynamic assignment of IP addresses enables new computers to join a network and obtain configuration parameters without manual intervention. The danger in dynamic address assignment is that two computers might obtain the same IP address, and this prevents both computers from using the network. Such duplicate address assignment can be *very* difficult to find. Therefore, DHCP is designed to eliminate any possibility that the same IP address might be assigned to two computers at the same time.

In conjunction with dynamic address allocation, DHCP enables automatic address reuse. Because only a limited number of IP addresses are available in any IP subnet, the DHCP server can run out of addresses to allocate, unless it can recover addresses that are no longer in use. The solution to address reuse in DHCP is to assign addresses for a finite period of time, known as a *lease*. When the lease on an IP address expires, the DHCP server can safely reassign that address to a different client. In this way, a server can reassign the IP addresses from computers that have left the network to computers that subsequently join the network.

When Servers Are Unavailable

Because access to the DHCP service is critical to the operation of an organization's computers (and, therefore, to the function of the organization as a whole), the DHCP specification allows two or more DHCP servers to provide service to the same network. Computers using DHCP broadcast an initial message to locate DHCP servers and must be prepared to receive responses from more than one server. Likewise, when extending a lease on an address, a computer broadcasts its request to find any available DHCP server if the original server doesn't respond. This means that if one DHCP server is not functioning, another server can continue to provide service.

The DHCP specification also anticipates the possibility that no DHCP servers are available. When a computer restarts, if it can't reach a DHCP server, it can use its old IP address as long as the lease on that address has not expired. If it has not been moved to a new network segment, the computer can start using the network even if the DHCP server is down.

Relay Agents

A DHCP client must contact DHCP servers before the client has been assigned an IP address. Without an IP address, the client can only transmit messages on its local network segment by using IP broadcast. Therefore, every network segment on which DHCP service will be provided must have some sort of DHCP service provider. DHCP provides for *relay agents,* which are very simple DHCP message forwarders that can be deployed on individual network segments. Relay agents enable DHCP service to be provided by just a few servers in a centrally managed location.

Avoiding Manual Configuration and Reducing Changes to Configuration

DHCP is an effective tool for managing thousands of computers, in part because it provides automated configuration services for new computers that were not previously connected to the network. That is, a computer can use DHCP with no manual configuration on the part of the user or the network administrator. A computer equipped with a DHCP client can find a DHCP server, exchange DHCP messages with that server, and use the information from the DHCP server to configure its TCP/IP stack. All this occurs as soon as the machine is started, without the network administrator taking special action for individual computers.

DHCP retains a client's configuration across client and server restarts. Servers must record assigned addresses and leases to permanent storage before responding to the DHCP client so that there is a reliable record of the IP address assigned to each client. Servers use this record to confirm each client's address assignment whenever the DHCP client is restarted. The DHCP specification also recommends that servers keep address assignment information after the lease has expired. This means that each client usually gets the same address, even if its lease expires. Clients with local permanent storage record their configuration information for reconfirmation when they restart the computer.

DHCP also gives the network administrator the opportunity to update client configurations automatically. Each time a DHCP client contacts the server, the server can update the client's configuration parameters. For example, when a new DNS server is installed, the network administrator updates the DHCP server configuration file, and each DHCP client is informed of that server the next time it restarts. Even a DHCP client that remains running and is not restarted periodically will learn of the new DHCP server when it contacts the server to extend the lease on its IP address.

Identifying Clients

A server must be able to identify which DHCP client sent a DHCP message so that it can match that message with an existing entry in its list of assigned addresses or create a new entry for that client. DHCP provides two different ways for a client to identify itself: the link-layer address of the interface that the DHCP client is configuring and the client identifier.

The *link-layer address* is also known as the *MAC address* in networks that use IEEE 802 technology. The link-layer address is unique among the devices on a network segment; in fact, for most network technologies, the link-layer address is unique among all computers using that same technology. The link-layer address is automatically configured and available without requiring user intervention. The link-layer address has some limitations as a DHCP identifier, however. It lacks flexibility, the DHCP identity of a client changes if its network interface adapter changes.

For clients that want to use a different identifier, DHCP defines the *client identifier* option. This option tells the server to use the value in the option to identify the client, rather than using the client's link-layer address. Some sites, for example, configure computers to use the computer's fully qualified domain name as the client identifier.

Whether the server uses the client's link-layer address or a client identifier to identify the DHCP client, the value of the identifier is assumed to be unique only on the network segment to which the client is attached. The server uses the combination of the network segment address and the client identifier to look up the client's information in the server's database of assigned addresses and configurations.

New Functions in DHCP

Several features that were not included in the original protocol specification for DHCP (that is, RFC 2131) are under development. The features described in this section are all close to being accepted as IETF standards. To check on the status of any of these features, see the Dynamic Host Configuration Working Group (DHCWG) Web page, `www.ietf.org/html.charters/dhc-charter.html`.

Communication Between Servers

Although the DHCP specification enables the existence of multiple servers, it doesn't include protocols for communication among DHCP servers. As later chapters discuss in more detail, DHCP servers must exchange information if a network administrator is to take full advantage of redundant servers. The DHCWG has been working on the development of an interserver protocol and, at the time of this writing, has a draft specification for the "DHCP Failover Protocol" ready for final review. Chapter 10, "Failover Protocol Operation," describes this interserver protocol.

Automatic DNS Updates

A natural link exists between DHCP and DNS. When a server assigns a new IP address to a DHCP client, DNS entries for that client must be added or updated. But when DHCP was first designed, the DNS database could be updated only via manual entries. No protocol existed through which the DNS database could be updated automatically. The DHCWG decided that this problem was outside the scope of its charter and declined to develop a solution. Now that dynamic DNS updates are defined in RFC 2136, the DHCWG has developed a formal definition of the interactions between DHCP and DNS, which is described in Chapter 11, "DHCP/DNS Interaction."

Server-Initiated Messages

The definition of DHCP in RFC 2131 includes only message exchanges initiated by DHCP clients. In some situations where the server has new information to deliver to the client immediately, it might be desirable for a server to initiate a message exchange. Initially, the DHCWG rejected proposals for server-initiated message exchanges as being overly complex to manage and impossible to scale to large DHCP deployments. Recently, a specification titled "DHCP Reconfigure Extension" has been accepted by the DHCWG and is awaiting final IETF review for acceptance as part of the DHCP specification.

New DHCP Options

As the deployment of DHCP continues to expand and new applications for DHCP are developed, the need for carrying additional information in DHCP messages has been identified. The DHCWG accepts and reviews proposals for new DHCP options to carry this information. As the new options are accepted as part of the standard, the specifications for those options are published as RFCs. To get the latest information on newly defined options and the options under development, go to the DHCWG Web page.

Functions Not Included in DHCP

Several functions or features were not incorporated into the DHCP protocol specification (see RFC 2131) because these functions do not comply with the working group's original design goals.

One IP Address per Interface

The DHCP specification restricts the use of DHCP to a single IP address for each network interface on a device. When DHCP was designed, few TCP/IP stacks were configured that had more than one IP address on a single interface, and it was not necessary to assign more than one IP address to an interface.

NOTE

Obtaining multiple IP addresses for a single interface can be done by using a different client identifier option for each IP address. A DHCP server treats messages with different client identifiers as though they came from different DHCP clients, even though the link-layer addresses in those messages might be identical. So, to assign multiple addresses to an interface, the network administrator should arrange for the DHCP client to make multiple requests to the server, each with a different client identifier.

DHCPv6, the version of DHCP under development for IPv6, enables the assignment of multiple IP addresses to a single interface. This feature of DHCPv6 is discussed in more detail in Chapter 25, "DHCP for IPv6."

Router Configuration

The DHCP specification clearly states that the protocol is not intended to configure routers. The DHCWG adopted that restriction on the scope of DHCP early on, to avoid the potential of added complexity in support of router configuration. Because routers are typically static, and because it is critical to the operation of a network that routers be correctly configured, interest in expanding DHCP to support router configuration is minimal.

NOTE

Walt Lazear of Mitre Corporation has done some research into what he calls *self-discovering networks.* In such a network, a self-configuring router is connected between a network segment that already has DHCP service and routing and one or more network segments that do not. The router has no preconfigured knowledge of its configuration; it uses the DHCP protocol to contact a DHCP server, which provides it with configuration information. Using this information, it configures all its interfaces and begins routing. At the same time, it configures its own DHCP server, which the next router in the chain can then use to configure itself. By deploying these routers in a lattice, network configuration information can propagate from a central server out to the edges of the network without requiring any intervention on the part of a knowledgeable user; it is the ultimate in plug-and-play networking.

Related TCP/IP Protocols

Clients can use other TCP/IP protocols, including those listed in Table 5.1, to obtain some of the configuration information DHCP provides.

TABLE 5.1 Other TCP/IP Configuration Protocols

Protocol	Function
Bootstrap Protocol (BOOTP)	Provides a static IP address, some IP stack parameters and addresses of some network servers. DHCP is based on BOOTP, and DHCP servers are easily extended to provide service to BOOTP clients.
Reverse Address Resolution Protocol (RARP)	Provides statically assigned IP address.
Dynamic RARP (DRARP)	Provides automated assignment of a static IP address.
Internet Control Message Protocol (ICMP)	Provides the subnet mask and default router.
Service Location Protocol (SLP)	Acts as a dynamic directory for identifying and locating network services; can provide flexible and dynamic configuration of services that are also provided through DHCP options.

RARP (see RFC 903)) and DRARP (see RFC 1931) provide a computer with its IP address, based on its link-layer address. Both RARP and DRARP use the same message format as Address Resolution Protocol (ARP; see RFC 826), and they return the IP address assigned to a link-layer address. A computer that uses RARP or DRARP to obtain its IP address broadcasts a `request` message that contains its link-layer address on its network segment. A server, which must be connected to the same network segment, determines the computer's IP address based on its link-layer address, and it returns that address in a `reply` message.

RARP and DRARP use different methods to determine what IP address to return to a requesting computer. RARP uses a table of link-layer addresses and IP addresses, which the network administrator creates and edits manually. DRARP assigns addresses to new clients automatically, without direct intervention by the network administrator.

Although RARP and DRARP provide only IP addresses, many computers use Trivial File Transfer Protocol (TFTP; see RFC 1350) to obtain further configuration information from the server that answered the RARP or DRARP request. For example, Sun diskless workstations use TFTP to download an appropriately configured Unix kernel after obtaining an IP address from the RARP or DRARP server. Note that at this writing, newer Sun systems can use DHCP rather than RARP.

ICMP (RFC 792) includes two mechanisms that a computer can use to obtain protocol stack parameters. A local authority, such as a router, uses the ICMP subnet mask message (RFC 950) to inform a computer of the appropriate subnet mask. A computer uses the ICMP router discovery messages (RFC 1256) to learn about routers on its local network segment. Two types of router discovery messages exist. The first, which the computer broadcasts, asks for a response from routers on the network segment. The second, which routers broadcast, announces that the router on the network segment is available. A computer uses a router discovered through the router discovery mechanism as its default router.

A computer can find network services, such as DNS and printers, by using Service Location Protocol (SLP; see RFC 2165). SLP is configured either with a central server or as a distributed service. In either case, a computer looking for a particular service formulates a request for the service and submits the request through SLP. The response, from either the SLP server or the computers providing the requested service, returns to the requesting computer information that describes the service and the address of the computer that is providing the service.

NOTE

Taken together, RARP/DRARP, the ICMP subnet mask and router discovery messages, and SLP provide most of the important configuration information a TCP/IP host requires. As explained in Chapter 3, "Configuring the DHCP Server," a computer needs an IP address, the subnet mask, the address of at least one default router, and the address of a DNS server before it can effectively use TCP/IP.

A network administrator might find that DHCP is a better choice than these other protocols for computer configuration management. Most importantly, DHCP provides all these configuration functions through a single service. A network administrator needs to manage only one DHCP server, rather than separate RARP/DRARP, ICMP, and SLP servers. Another advantage of using DHCP is that it includes the leasing mechanism for automated recovery and reliable reassignment of IP addresses. Finally, DHCP can provide other TCP/IP stack parameters in addition to an IP address, subnet mask, and default router, and DHCP does not require a server on every network segment.

The DHCP Client/Server Architecture

In the DHCP client/server model, the clients are the computers that use DHCP services to obtain IP addresses and parameters. DHCP servers, managed by network administrators, hand out the configuration information. DHCP clients initiate all client/server transactions and are responsible for handling all the details of each transaction, including generating transaction identifiers and retransmitting lost protocol messages.

The DHCP model of centralized administration came about for at least two reasons: to minimize client configuration before the client uses DHCP and to give network administrators full control over the configuration of networked computers. BOOTP also influenced the architecture and the details of DHCP. The client/server organization in DHCP is identical to that in the BOOTP model. The DHCP message formats, including the fixed-format header area and the individual option formats, are almost identical to those in BOOTP. DHCP also uses relay agents in the same way as BOOTP, to forward DHCP messages from clients to servers, avoiding the need for a DHCP or BOOTP server on every network segment.

NOTE

Compatibility with BOOTP relay agents was the deciding factor in reusing the BOOTP message formats for DHCP. When DHCP was designed, router vendors were just beginning to include BOOTP message forwarding in their products. Rather than delay DHCP deployment until a new type of relay agent was developed and integrated into routers, the DHCWG decided to retain the BOOTP message format so that DHCP could use the installed base of BOOTP relay agents.

In one sense, clients are more complicated than servers. Clients must maintain internal state information and generate the appropriate sequences of messages for client/server transactions. Servers do not have to maintain protocol state information for clients (although they must, of course, keep track of the IP address assignments); they can simply respond whenever they receive a request from the client.

In another sense, servers are more complex than clients. A server must read and parse a configuration file that describes the network architecture and the local DHCP policies, and it must store the information about assigned addresses in some kind of persistent storage, such as a disk file or database. Most importantly, a server must be implemented carefully so that it responds quickly to client messages.

Summary

DHCP provides hands-off configuration of networked computers through network messages the computer exchanges with a centralized server. The server manages the client's configuration, assigns it an IP address and provides other configuration parameters that are specified by the network administrator's policies.

The client/server model in DHCP assigns all responsibility for initiating transactions to the client. Servers are essentially stateless and can respond to individual client messages independently. A server does maintain pertinent information about the addresses assigned to clients, but it need not track the state of each client as it obtains and uses its configuration.

Other protocols, including RARP/DRARP, ICMP, and SLP, also provide some of the services DHCP provides. DHCP has the advantage of providing configuration management through a single service, which reduces administrative overhead.

As this book was written, the working group was completing the definition of several new functions and options for DHCP, including a protocol through which DHCP servers can exchange information about clients, coordination of updates to DNS information with assignment of addresses through DHCP and new options to carry additional configuration information.

6

The Format of DHCP Messages

IN THIS CHAPTER

DHCP clients and servers communicate by exchanging messages as described in the protocol specification. All DHCP messages share a common format, which is described in this chapter. Future chapters describe how DHCP messages are transmitted by using UDP and the specific options that can be carried in a DHCP message.

DHCP was developed from BOOTP (see RFC 951) and uses a message format that is based on the BOOTP specification. Because DHCP shares UDP ports 67 and 68 with BOOTP, DHCP messages include a special option in the `option` field that differentiates them from BOOTP messages.

DHCP Message Format Overview

All DHCP messages include a *fixed-format* section and a *variable-format* section. The fixed-format section consists of several fields that are the same in every DHCP message. The variable-format section holds *options*, which carry additional configuration parameters. The contents of the fixed-format section and the format of the variable-format section vary according to the type of DHCP message.

The fixed-format section is divided into several fields that carry information such as the following:

- The network and link-layer addresses of the client
- The IP address of the server
- Control information about the message itself

These fields appear in every DHCP message, although not all fields are used in every type of message.

The type of each DHCP message, as well as other information not defined in the fixed-format section, is specified by *DHCP options* in the variable-format section. The length of each DHCP option, as well as the contents and format of the data in the option, depends on the definition of that option.

By default, DHCP messages contain 576 or fewer bytes (including the IP and UDP headers). However, a client can indicate to a server that it is prepared to accept messages larger than 576 bytes. If the server's response requires more than 576 bytes and the server can send a larger message, it might take advantage of the client's willingness to accept a larger message. In such cases, each message is carried in a single UDP datagram.

The Fixed-Format Section

The fixed-format , illustrated in Figure 6.1, appears in every DHCP message. In the figure, each row represents 32 bits of the fixed-format section. Individual fields are delimited by vertical bars and are labeled with the name of the field. Some of the larger fields are summarized in the figure, and their lengths are given explicitly. The fields in the fixed-format section are summarized in Table 6.1.

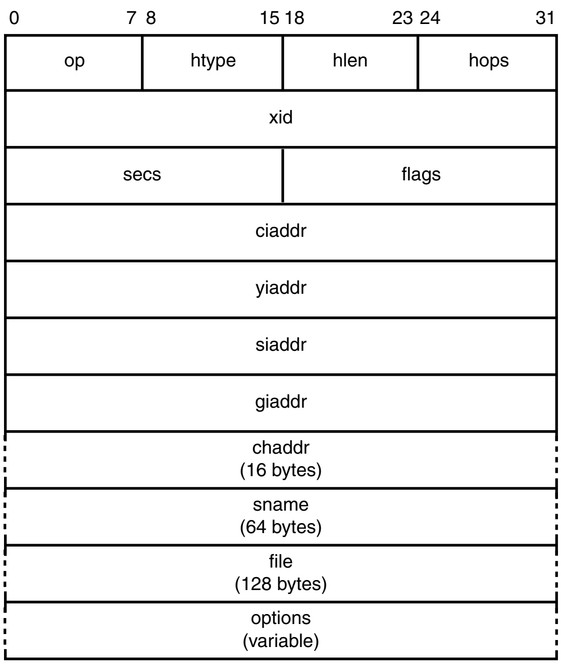

FIGURE 6.1 The fields in the fixed-format section of a DHCP message.

TABLE 6.1 A Summary of the DHCP Message Fields

Field	Description
`op`	Message operation code; set to `1` in messages sent by a client and `2` in messages sent by a server. The two possible values for `op` are carried forward from BOOTP for backward compatibility and are sometimes called `BOOTREQUEST` and `BOOTREPLY`, respectively.
`htype`	Link-layer address type; definitions are taken from the IANA list of ARP hardware types (see RFC 1700). For example, the ethernet type is specified when `htype` is set to `1`.
`hlen`	Link-layer address length (in bytes); defines the length of hardware address in the `chaddr` field.
`hops`	Number of relay agents that have forwarded this message.
`xid`	Transaction identifier; used by clients to match responses from servers with previously transmitted requests.
`secs`	Elapsed time (in seconds) since the client began the DHCP process.
`flags`	Flags field; the least significant bit, called the *broadcast bit*, can be set to `1` to indicate that messages to the client must be broadcast (see the section "Using the Broadcast Flag" in Chapter 7, "Transmitting DHCP Messages," for details).
`ciaddr`	Client's IP address; set by the client when the client has confirmed that its IP address is valid.
`yiaddr`	Client's IP address; set by the server to inform the client of the client's IP address (that is, "your" IP address).
`siaddr`	IP address of the next server for the client to use in the configuration process (for example, the server to contact for TFTP download of an operating system kernel).
`giaddr`	Relay agent (or gateway) IP address; filled in by the relay agent with the address of the interface through which DHCP message was received.
`chaddr`	Client's link-layer address.
`sname`	Name of the next server for client to use in the configuration process.
`file`	Name of the file for the client to request from the next server (for example, the name of the file that contains the operating system for this client).

The `options` Section

The `options` section of a DHCP message carries a sequence of options that convey additional configuration information between the client and the server. Each option includes an option code, an option length, and option data, as illustrated in Figure 6.2. The option code identifies the specific option and the information carried in the option. The option data is the actual information, and the option length is the length, in bytes, of the option data field.

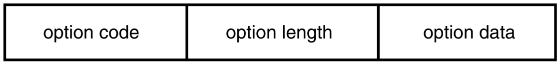

FIGURE 6.2 The format of the DHCP message `options` section.

The first 4 bytes in the `options` section define the format of the remainder of the section. The 4 bytes are set to the decimal values 99, 130, 83, 99 (or 63, 82, 53, 63 in hexadecimal). These values are referred to as a *magic number* in the BOOTP specification (see RFC 951) and a *magic cookie* in the DHCP options and BOOTP extensions (originally defined in RFC 1048 and included in RFC 2132).

The options are stored sequentially in the `options` section, without word alignment or other formatting restrictions. The option data is formatted according to the specification of the particular option. Examples of data formats include a single IP address, a list of IP addresses, and a character string.

The remainder of this section describes some examples of options, and the following section illustrates the use of DHCP message fields and options in messages exchanged between a client and a server. The options described in the remainder of this section illustrate some specific option data formats.

OPTION ORIGIN

One of the reasons for using a variable-format section is to allow the definition of new options as new configuration requirements are defined. Many of the options in RFC 2132 carry the addresses of servers for applications and services that hadn't been invented when the first version of DHCP was published as a standard in 1993. Today, new options are being developed to adapt DHCP to new technologies, such as DSL and cable modems, and to allow DHCP to operate with other protocols, such as DNS.

The IETF defines new options through a process described in RFC 2132 and subsequently modified in RFC 2939 (and also in BCP 29). Each new option (or group of related options) is described in a separate document and considered for adoption independently. In the review process, which is based on the IETF standards process for new Internet protocols, a new option is first defined in an Internet Draft. The draft is then reviewed by the DHC working group of the IETF and is revised based on input from the working group. After the draft has passed the working group review, it is submitted to the IESG for acceptance as an Internet Standard. When the new option is accepted as a standard, it is assigned an option number.

The `DHCP message type` Option

This section describes the `DHCP message type` option. DHCP client/server message exchanges are composed of messages of different types, representing the steps in the transaction. DHCP messages are often referred to by their type. In the example of DHCP operation described in Chapter 2, the client first broadcasts a `DHCPDISCOVER` message (or just `DISCOVER`), and the server replies with a `DHCPOFFER` message.

The DHCP message type option, which is included in every DHCP message, identifies the type of DHCP message being sent. The DHCP message type option includes the DHCP message type option code (53), the length of the data field (1), and the message type, encoded as a single byte with one of the following values:

Message Type	Option Value
DHCPDISCOVER	1
DHCPOFFER	2
DHCPREQUEST	3
DHCPDECLINE	4
DHCPACK	5
DHCPNAK	6
DHCPRELEASE	7
DHCPINFORM	8
DHCPFORCERENEW	9

Figure 6.3 illustrates the format of the DHCP message type option.

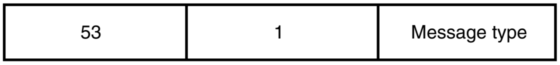

FIGURE 6.3 The format of the DHCP message type option.

NOTE

The DHCP message type option also differentiates DHCP messages from BOOTP messages. Because DHCP and BOOTP use different messages and semantics, a server must be able to distinguish between DHCP messages and BOOTP messages so that it can process each appropriately. Although DHCP and BOOTP messages share the same format and are delivered to the same UDP port, only DHCP messages include the DHCP message type option. Thus, a server can use the presence of the DHCP message type option to determine that a message is a DHCP message.

The subnet mask Option

The subnet mask option carries the subnet mask that the client should use for its local network segment. Whereas the client's IP address appears in the ciaddr field in the fixed-format section of the DHCP message, the subnet mask is carried in an option.

NOTE

The allocation of a field in the fixed-format section of the message to the client's IP address while using an option for the `subnet mask` was first employed in BOOTP and carried over to DHCP. The reason for this design decision is purely historical.

The `subnet mask` option has an option code of 1, a length of 4, and a subnet mask encoded as a 32-bit IP address. The subnet mask is encoded in network-byte ordering, with the most significant byte immediately following the `length` field. Figure 6.4 illustrates the format of the `subnet mask` option.

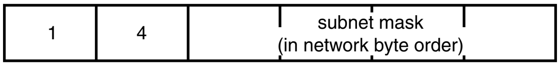

FIGURE 6.4 The format of the DHCP `subnet mask` option.

The `router` Option

Before a host can exchange IP datagrams with other hosts on different network segments, it must know the address of at least one router on its own network segment. The `router` option carries a list of routers, sometimes known as *default routers*, which are connected to the same network segment as the client. The `router` option can carry the addresses of more than one router, in order to provide backup routers that a client can use in case one router fails. The `router` option has an option code of 2, the length (four times the number of routers listed in the option), and the addresses of the routers.

Figure 6.5 provides an example of the `router` option. In the figure, the option includes two router addresses; more router addresses can be carried if desired. Although the routers are listed in order of preference, in practice the client can decide how to choose among the specified routers.

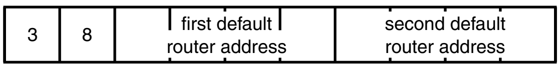

FIGURE 6.5 The format of the `router` option.

The `DNS server` Option

If a host is to support the translation of domain names into IP addresses, it must know the address of a DNS server to which it can send name-resolution requests. The `DNS server` option carries a list of addresses of DNS servers that the client can use, in the same format as the `router` option. The `DNS server` option includes the option code, 6, the length (four times the number of DNS server addresses carried in the option), and the addresses of the DNS servers.

Figure 6.6 shows the format of a `DNS server` option that is carrying the address of one DNS server. Like the `router` option, the `DNS server` option can carry more than one server address.

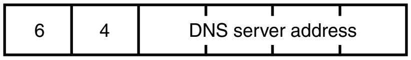

FIGURE 6.6 The format of the `DNS server` option.

The `requested IP address` Option

When a DHCP client is allocated an IP address, it must check that address every time it starts or connects to a network. As described in the example in the section "Restarting `desktop1`" in Chapter 2, the client contacts the DHCP server with a DHCP message that contains the address assigned to the client. The address is carried in the `requested IP address` option. The format of this option includes the option code (50), the length (4), and the client's requested address, as shown in Figure 6.7.

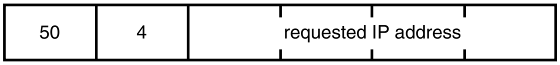

FIGURE 6.7 The format of the `requested IP address` option.

The `end` Option

The `end` option indicates the end of the options in the `options` section of a DHCP message. The `end` option is formatted in a slightly different manner from the other options described in this section; the `end` option has a fixed length and includes neither a `length` field nor data. The format, as shown in Figure 6.8, is a single byte set to the value 255.

NOTE

The `end` option is not required by the DHCP specification, and it might not appear in the `options` section of every DHCP message. DHCP clients and servers must be prepared to correctly process DHCP messages that do not include an `end` option.

FIGURE 6.8 The format of the `end` option.

Examples of Message Formats

To illustrate the format of DHCP messages, this section uses the example from the section "Using DHCP to Configure Computers" in Chapter 2 and presents the details of the `DHCPREQUEST` message, sent from the client to the server, and the details of the `DHCPACK` message, sent in reply from the server to the client. Although the `DHCPREQUEST` message does not begin the protocol exchange between client and server (that is the purpose of the `DHCPDISCOVER` message), it is the most frequently observed DHCP message type.

The `DHCPREQUEST` Message Format

The client constructs its `DHCPREQUEST` message with its hardware address in the `chaddr` field and its IP address in the `requested IP address` option. The resulting message is shown in Figure 6.9.

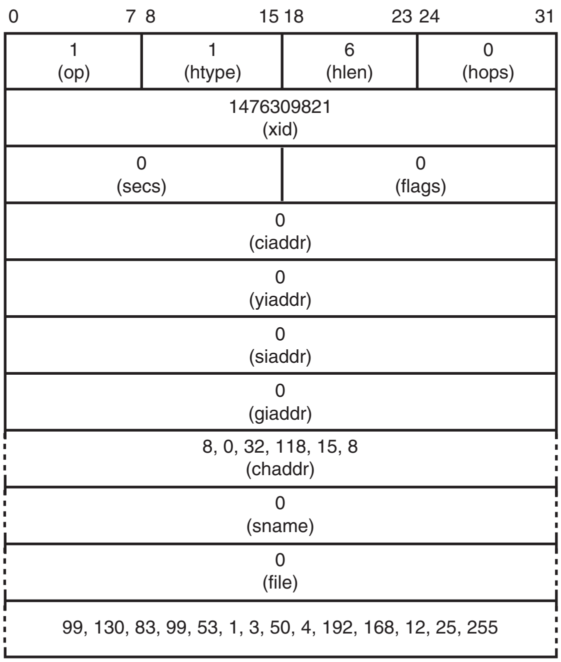

FIGURE 6.9 An example of a `DHCPREQUEST` message.

In this message, the client has filled in the fields and options shown in Table 6.2.

TABLE 6.2 Fields in the Sample `DHCPREQUEST` Message

Field or Option Name	Value	Comments
`op`	1	A message from a client
`htype`	1	Indicates an ethernet address
`hlen`	6	The length (in bytes) of the link-layer address
`hops`	0	Indicates that the message has not been forwarded by relay agents
`xid`	1476309821	A random number chosen by the client to identify this request
`secs`	0	The time elapsed (in seconds) since the client began sending DHCP requests
`flags`	0	Indicates that no flags are set
`ciaddr`	0	(unused in this message)
`yiaddr`	0	(unused in `DHCPREQUEST`)
`siaddr`	0	(unused in `DHCPREQUEST`)
`giaddr`	0	Contains the address of the relay agent if the message was forwarded by a relay agent
`chaddr`	8:00:20:76:0f:08	The client's Ethernet address
`sname`	0	(unused in `this message`)
`file`	0	(unused in this message)
DHCP message type	3	A `DHCPREQUEST` message
requested IP address	192.168.11.25	An address for which the client has a lease
`end`	255	The end of options in `option` section

Here is the `DHCPREQUEST` message, as it is transmitted to the server, in hexadecimal format:

```
     00  01  02  03    04  05  06  07    08  09  0A  0B    0C  0D  0E  0F
000  01  01  06  00    57  FE  B3  3D    00  00  00  00    00  00  00  00
010  00  00  00  00    00  00  00  00    00  00  00  00    08  00  20  76
020  0F  08  00  00    00  00  00  00    00  00  00  00    00  00  00  00
...
0E0  00  00  00  00    00  00  00  00    00  00  00  00    63  82  53  63
0F0  35  01  03  32    04  C0  A8  0B    19  FF  00  00    00  00  00  00
```

The `DHCPACK` Message Format

In response to the `DHCPREQUEST` message, the server constructs a `DHCPACK` message, which confirms that the client's IP address is appropriate for the network segment to which it is attached. In this example, the server also includes other configuration parameters: the subnet mask, the address of a default router, and the address of a DNS server. Figure 6.10 illustrates the contents of the `DHCPACK` message.

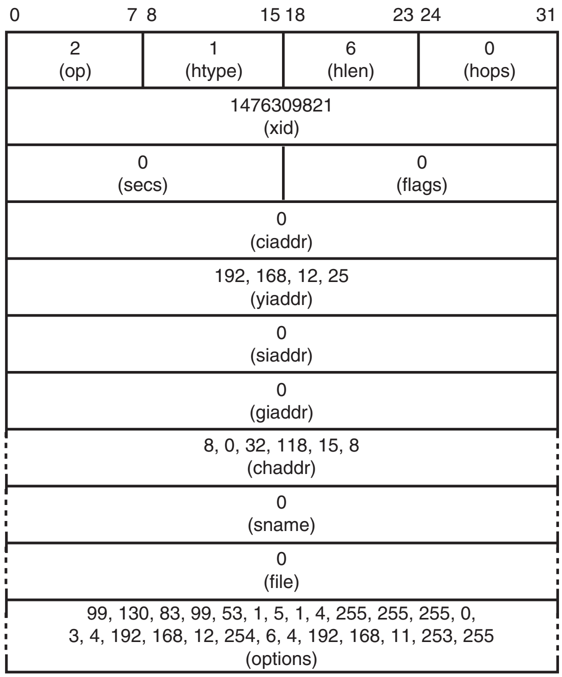

FIGURE 6.10 An example of a `DHCPACK` message.

The fields and options the server sends in the `DHCPACK` message shown in Figure 6.10 are shown in Table 6.3.

TABLE 6.3 Fields in the Sample `DHCPACK` Message

Field or Option Name	Value	Comments
`op`	2	A message from a server
`htype`	1	Indicates an ethernet address
`hlen`	6	The length (in bytes) of the hardware address
`hops`	0	Indicates that the message has not been forwarded by relay agents
`xid`	1476309821	Copied from the client's `DHCPREQUEST` message
`secs`	0	(unused in `DHCPACK`)
`flags`	0	Indicates that no flags are set
`ciaddr`	0	(unused in `DHCPACK`)
`yiaddr`	192.168.11.25	The address that is confirmed by the server

TABLE 6.3 Continued

Field or Option Name	Value	Comments
`siaddr`	192.168.11.252	The server's IP address
`giaddr`	0	(unused in DHCPACK)
`chaddr`	8:00:20:76:0f:08	The client's hardware address
`sname`	0	(unused in `DHCPACK`)
`file`	0	(unused in `DHCPACK`)
`DHCP message type`	5	This is a `DHCPACK` message
`Server identifier`	192.168.11.252	The IP address of the DHCP server
Lease time	2592000	Lease time in seconds (30 days)
subnet mask	255.255.255.0	The subnet mask for the client
router	192.168.11.254	The default router for the client
DNS server	192.168.11.253	The DNS server for the client
Domain name	`genericstartup.com`	The default domain name for the client
end	255	The end of the options in the `option` section

Here is the `DHCPACK` message in hexadecimal format:

```
     00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
000  02 01 06 00  57 FE B3 3D  00 00 00 00  00 00 00 00
010  C0 A8 0B 19  C0 A8 0B FC  00 00 00 00  08 00 20 76
020  0f 08 00 00  00 00 00 00  00 00 00 00  00 00 00 00
...
0E0  00 00 00 00  00 00 00 00  00 00 00 00  63 82 53 63
0F0  35 01 05 36  04 C0 A8 0B  FC 33 04 00  27 8D 00 01
100  04 FF FF FF  00 03 04 C0  A8 0B FE 06  04 C0 A8 0B
110  FD 0F 12 67  65 6E 65 72  69 63 73 74  61 72 74 75
120  70 2E 63 6F  6D FF
```

DHCP MESSAGE FORMAT SPECIAL CASES

A couple of special cases affect the format of options in a DHCP message. The first occurs when multiple instances of the same option appear in the `options` area. The motivation for allowing multiple instances of an option comes from the limitation on the data carried in an option. Because the size of the data must be specified in a single byte, an option carries, at most, 255 bytes of data. In order to allow a DHCP server or client to send an option longer than 255 bytes, multiple instances of any option are allowed in a single message. DHCP specifies that the data from instances of a single option are concatenated and interpreted as a single instance of the option. Thus, a list of 70 DNS servers can be split into two DNS server options, each carrying the addresses of 35 servers in 140 bytes.

The situation in which long lists of items are carried in an option leads to the second special case. Suppose the collection of options being sent in a DHCP message exceeds the default maximum `options` section size of 312 bytes. In anticipation of this problem, DHCP enables redefinition of the `sname` and `file` fields from the `fixed-format` section of the DHCP message to hold options rather than a server name and configuration file name. Using the two fields from the `fixed-format` section allows for 192 additional bytes of options, an increase of more than 60%.

Design Constraints

One might look at the format of a DHCP message and ask, "How did anyone come up with *this* format?" In the right situation, Ralph Droms, who is one of the authors of this book and was part of the initial DHCP design process, might be persuaded to admit that the DHCP message format is, indeed, arcane and bordering on ugly. The primary constraint around which the DHCP message format was designed is backward compatibility with BOOTP messages. When the design choices were made (in 1990–1991), some router vendors were including BOOTP relay agents in their products. The DHCP designers retained the original BOOTP format for DHCP in order to leverage the availability of these BOOTP relay agents and allow for easy DHCP deployment. To ensure that the BOOTP relay agents would correctly forward DHCP messages, the BOOTP format was retained. Rather than encode new DHCP functions in the `op` field (which might have caused deployed BOOTP relay agents to reject the message), the DHCP functions were encoded as options (which BOOTP relay agents never touch).

DHCP messages differ from BOOTP messages in four important ways:

- DHCP messages include a `flags` field, which is in an area defined as "unused" by the BOOTP specification. The definition of the `flags` came about as an engineering solution (or workaround) for certain TCP/IP stacks; Chapter 7 explains the details of the `flags` field.
- DHCP allows the `options` section (called the `vendor extensions` field in BOOTP) to be at least 312 bytes long, whereas BOOTP allows only 64 bytes of options.
- The `DHCP message type` option identifies DHCP messages.
- The `sname` and `file` fields in a DHCP message can be used to hold additional options.

DHCP also retains the format of options, originally defined in RFC 1048, and the convention of reserving option codes 128–254 for local use. Therefore, a network architect can safely use the local-use options without conflicting with standard options, which use option codes 1–127. However, this numbering convention is becoming problematic because the option codes are being exhausted. When BOOTP first adopted the convention about a decade ago, only a handful of options existed. Therefore, reserving only the first 127 codes for globally defined options seemed appropriate.

NOTE

Although option codes 128–254 are supposed to be reserved for definition and local use by network administrators, some vendors of DHCP clients use those option codes for options that are specific to their particular clients. When vendors use option codes 128–254 for their own clients, they create a potential conflict with local use of those option codes. Rather than using option codes 128–254 for their clients, vendors should use vendor-specific options, a procedure discussed in more detail in Chapter 8, "DHCP Message Exchanges."

Summary

DHCP messages include a fixed-format section and a variable-format section. The DHCP message format is based on the message format used by BOOTP, primarily to ensure backward compatibility with BOOTP relay agents. Several different DHCP messages exist, and all share the same format. The fixed-format section of a DHCP message carries information that identifies the client to the server and conveys some configuration information from the server to the client. The variable-format section holds options that carry additional configuration parameters.

The type of DHCP message is determined by the `DHCP message type` option. The presence of this option also differentiates DHCP from BOOTP messages. If a DHCP/BOOTP format message includes a `DHCP message type` option, it is a DHCP message; otherwise, it is interpreted as a BOOTP message. Other options carry protocol stack configuration parameters, the addresses of application servers, and other configuration information.

7

Transmitting DHCP Messages

IN THIS CHAPTER

- Using UDP for DHCP
- Relay Agents
- Reliable Delivery of DHCP Messages
- Other Transmission Methods
- Authenticated DHCP Messages

DHCP clients, servers, and relay agents use UDP ports 67 and 68 when transmitting DHCP messages. Use of these ports is derived from BOOTP. Port 67 is referred to as BOOTPS (RFC 1700) or the *DHCP server po*rt, and port 68 is known as BOOTPC or the *DHCP client port.*

Use of UDP raises an interesting question: How can a DHCP client use UDP before the client has a valid IP address? DHCP solves this problem by using the limited broadcast IP address, 255.255.255.255, as the destination IP address and the "this host" address, 0.0.0.0, as the source IP address in any message sent by a DHCP client that does not have an IP address.

Use of the limited broadcast IP address limits the delivery of messages from a client to servers that are connected to the same network segment as the client. DHCP uses relay agents, often running in routers, to forward broadcast messages from clients to servers and to forward replies from servers back to clients.

This chapter explains DHCP message delivery, including the use of UDP and broadcast, relay agents, and retransmission of lost DHCP messages.

Using UDP for DHCP

The client sends all DHCP messages to UDP port 67 (the DHCP server port). The client broadcasts messages if it does not yet have an address. If the client knows the address of a DHCP server and has an IP address of its own, it sends DHCP messages directly to the server.

Broadcast Messages

If the client doesn't yet have a valid IP address, it broadcasts messages to the limited broadcast IP address, 255.255.255.255, and to the appropriate link-layer broadcast address. The client uses port 68 (the DHCP client port) as the UDP source port, 0.0.0.0 as the IP source address, and its own link-layer address as the frame source address. This use of IP and link-layer addresses complies with RFC 1122.

NOTE

DHCP assumes that every client uses a physical layer and data link layer that together support broadcast. Use of DHCP to configure DHCP clients that are connected to a network segment whose physical or data link layer do not support broadcast has been proposed, but standards have not been developed to date. DHCP servers and relay agents communicate using unicast datagrams, so this restriction applies only to the network segment to which a DHCP client is attached.

In the DHCP message header, the client puts its link-layer address in the `chaddr` field and sets the `ciaddr` field to `0`. The `ciaddr` field is used only if the client has confirmed that its IP address is usable on its local network. The client also sets the `op` field to `BOOTREQUEST`.

Figure 7.1 shows a DHCP message that is broadcast by the client `desktop1`.

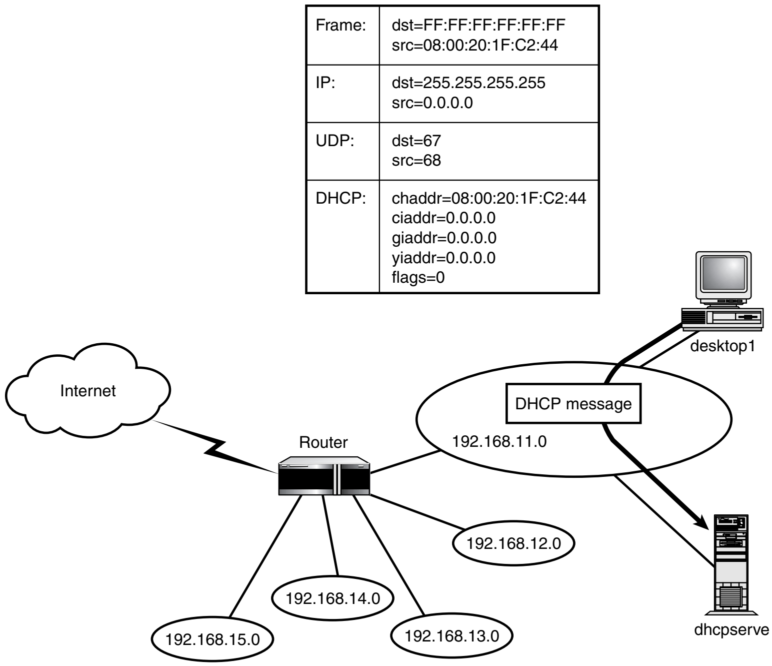

FIGURE 7.1 A DHCP message that is broadcast by the client `desktop1`.

Unicast Messages

When the client has a valid IP address, it sends DHCP messages to the DHCP server by using IP unicast, using its own IP address as the IP source address. The client then fills in the `chaddr` field with its link-layer address and the `ciaddr` field with its IP address. For example, a client uses IP unicast when it is extending the lease on an address.

Server Response Messages

Servers send DHCP messages directly to clients with the following information:

- UDP destination port 68
- UDP source port 67
- The IP destination address set to the client's IP address
- The link-layer destination address set to the client's link-layer address

Servers also set the `op` field to `BOOTREPLY`.

Figure 7.2 shows a DHCP message sent by a server in response to the message from the DHCP client in Figure 7.1.

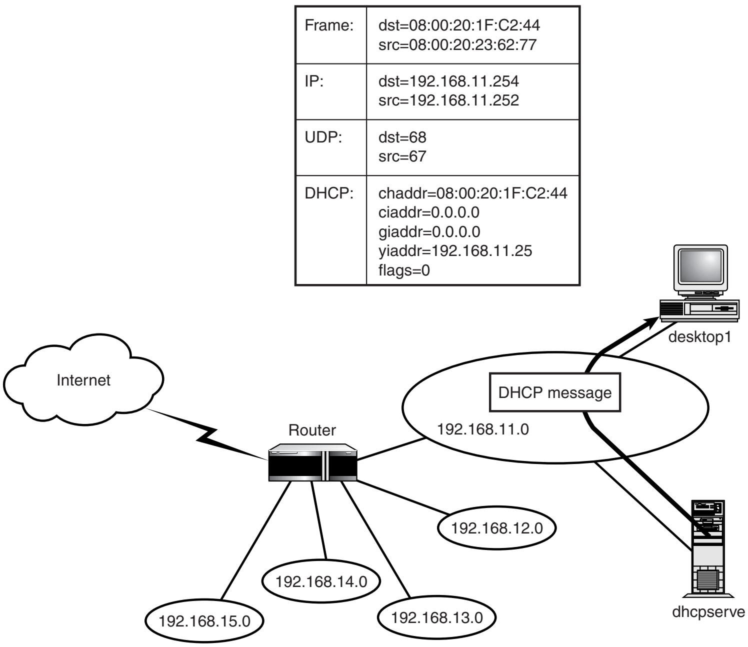

FIGURE 7.2 A DHCP message sent by `dhcpserve` in response to a message from `desktop1`.

NOTE

If the server sends a DHCP message to a client before the client has confirmed its IP address, the client will not respond to link-layer address resolution protocols such as ARP. The DHCP server or relay agent may need to explicitly create an entry in its ARP table to provide a mapping between the IP address it has assigned to the client and the client's link-layer address, so that the IP stack on the server will not have to ask the DHCP client for its link-layer address.

Using Broadcast for Delivery to Clients

Some TCP/IP implementations do not accept incoming IP datagrams that have unicast destination addresses before the IP address has been configured. For such implementations, the server sends DHCP messages to the IP broadcast address, 255.255.255.255, and to the local hardware broadcast address.

Using the Broadcast Flag

Sending DHCP messages to the client's IP address is the preferred delivery mechanism because it avoids interrupting other computers on the network with a broadcast message. Therefore, servers use unicast for DHCP messages unless a client explicitly requests the use of broadcast.

A client can specify that servers use broadcast by setting the broadcast flag in the `flags` field of the DHCP message header. If a server receives a DHCP message with the broadcast flag set to 1, the server broadcasts responses to the client.

On some operating systems, it is not possible to send unicast messages to clients that do not yet have confirmed IP addresses. In these cases, the DHCP server broadcasts the message to the client even if the client does not set the broadcast flag to 1.

Relay Agents

Relay agents forward DHCP messages from clients to servers in cases where there is no DHCP server on the network segment to which the client is connected. A relay agent listens to the DHCP server port and receives broadcast messages from DHCP clients. When the relay agent receives a client message, the relay agent does the following:

1. If the `giaddr` field in the message contains 0.0.0.0, the relay agent inserts the address of the network interface on which the message was received into the `giaddr` field.
2. It increments the hop count.
3. It appends any relay agent options.
4. It forwards the message to the DHCP servers that were configured by the network administrator.

Figure 7.3 shows the original message that is broadcast by the DHCP client and the message that is forwarded by the relay agent to the DHCP server. In this example, the DHCP message on network segment 192.168.12.0 is essentially the same as the message in Figure 7.1. The contents of the message from the relay agent to the DHCP server illustrate the details of the forwarded message.

Frame:	dst=08:00:20:23:62:77 src=08:00:20:A3:99:4C
IP:	dst=192.168.11.252 src=192.168.12.254
UDP:	dst=67 src=68
DHCP:	chaddr=08:00:20:1F:C2:44 ciaddr=0.0.0.0 giaddr=192.168.12.254 yiaddr=0.0.0.0 flags=0

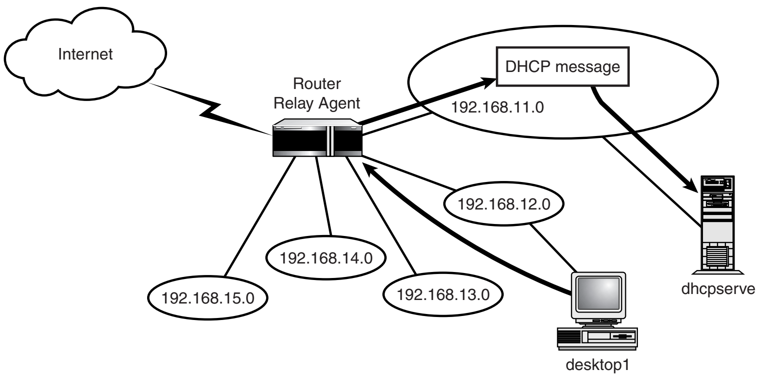

FIGURE 7.3 A DHCP message forwarded from `desktop1` to `dhcpserve` by a relay agent in a router.

Relay Agent Options

RFC 3046 defines the `relay agent information` option, which allows relay agents to add information to a message sent by a DHCP client to a DHCP server. The `relay agent information` option is used to pass additional information related to the client between the relay agent and the server. This information is encoded as suboptions within the `relay agent information` option. Several suboptions can be used to carry specific pieces of information between a relay agent and a server. The section "`relay agent information`" in Chapter 9 defines the `relay agent information` option and suboptions in more detail and gives examples of the uses for the `relay agent information` option.

Forwarding Destinations

The destination to which a relay agent forwards DHCP messages must be explicitly configured for the relay agent. Many relay agents can be configured with more than one forwarding destination. This enables the relay agent to forward separate copies of the client DHCP message to multiple DHCP servers.

NOTE

Network architects usually configure relay agents to forward client messages using IP unicast. Thus, the DHCP message from the client is broadcast only over the network segment to which the client is attached, and it is subsequently delivered directly to the server or servers in a unicast message. This means that DHCP broadcasts are limited to the networks to which clients are attached—there are no DHCP broadcasts on the network segment to which the DHCP server is connected unless there are DHCP clients on that network segment.

Response Delivery

DHCP servers use relay agents to deliver responses to DHCP clients. However, a DHCP server must explicitly forward DHCP messages through the appropriate relay agent. To determine how to deliver responses, a server examines the `giaddr` field in the message from the client. If the `giaddr` field is set to 0, a relay agent did not forward the message and the client must be connected to the same network segment as the server. In this case, the server delivers the message directly to the client. If the `giaddr` field is not set to 0, the server sends the response to the IP address in the `giaddr` field—the address of the relay agent that originally forwarded the client's DHCP message to the server.

Figure 7.4 shows the message the server sent to the relay agent and the message the relay agent delivered to the client in response to the sample message in Figure 7.3.

Messages from a server to a relay agent are sent to the DHCP server port. When a relay agent receives a message on the DHCP server port that has the `op` field set to `BOOTREPLY`, it forwards that message to the appropriate client. The relay agent forwards the server message to the client on the network segment identified by the address in the `giaddr` field. Because this field contains the IP address of the interface on which the client's message was originally received, the relay agent can deliver the response to the client through the same interface. The relay agent must also honor the broadcast flag and broadcast the response to the client if the broadcast flag is set to `1`.

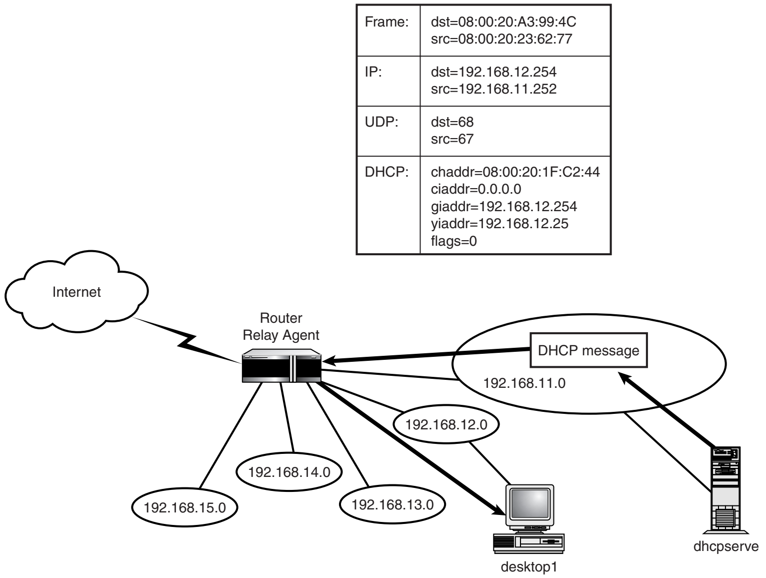

FIGURE 7.4 A response from `dhcpserve` forwarded by a relay agent in a router to `desktop1`.

Multiple Relay Agents

A DHCP message can be forwarded through more than one relay agent. In this case, the second relay agent finds that the `giaddr` field in the message already contains the address of the first relay agent. If a relay agent receives a DHCP message with a nonzero `giaddr` field, it forwards the message as usual but does not modify the contents of the `giaddr` field. When a DHCP server receives a message that has been forwarded by multiple relay agents, the `giaddr` field contains the address of the relay agent that first received the message. The server sends its response directly to the address specified in the `giaddr` field, bypassing the intermediate relay agents.

NOTE

RFC 3046 specifically prohibits any relay agent except the first from adding a `relay agent information` option. Network administrators at sites that use the `relay agent information` option should be sure that only the first relay agent in a relay agent chain needs to add a relay agent information option.

Relay Agent Implementation

A relay agent does not have to keep track of information about specific DHCP messages it forwards. However, it must be configured with the address or addresses to which the client messages should be forwarded. All the other information the relay agent needs is contained in the DHCP message, which means the relay agent can be a stateless device. A relay agent is therefore simple to implement, doesn't consume many resources, and can support any number of DHCP clients without scaling problems.

Most commercial routers provide a relay agent function; although it is not required that a router perform this function, relay agents in dedicated routers are the most common configuration.

Reliable Delivery of DHCP Messages

Because UDP does not guarantee delivery, DHCP provides for reliable delivery. DHCP clients, not servers, are responsible for managing message delivery. A client uses the response from a server as an implicit acknowledgment of receipt of the original message from the client. If the client does not receive a response to a message, the client retransmits the message as necessary until it does receive a response, or until it decides that no server is responding to DHCP messages.

The DHCP specification defines the amount of time a client must wait before retransmitting a message. The client waits 4 seconds before the first retransmission. It then doubles the waiting time between retransmissions, up to a maximum of 64 seconds. The particular client implementation determines how many retransmissions to send before giving up on the transmission of a message.

NOTE

There is no provision in DHCP for a DHCP server to respond to a `DHCPDISCOVER` message by saying "I can't provide service." So when a DHCP server is not able to provide service to a DHCP client, it simply ignores requests from that client. This can happen either because the DHCP server has no IP addresses to allocate or because it has been configured not to respond to some clients.

There is one exception to the retransmission rules: The `DHCPRELEASE` message is never retransmitted. Because it is an advisory message, the only harm that can come from it being lost is that the DHCP server will not free up the lease until it has expired.

Reliable delivery of a `DHCPFORCERENEW` message, which is initially sent by a server, is provided by the server. The server retransmits the `DHCPFORCERENEW` message, doubling the retransmission delay to provide exponential backoff, until it receives a

`DHCPREQUEST` message from the client. The specific retransmission delays and the number of times the server should retransmit the `DHCPFORCERENEW` message before giving up are not defined by DHCP.

Reliable delivery of a `DHCPLEASEQUERY` message is the responsibility of the original sender. The server responds to a `DHCPLEASEQUERY` message with either a `DHCPKNOWN` or a `DHCPUNKNOWN` message. If the sender doesn't receive a response, it retransmits the `DHCPLEASEQUERY` message, using exponential backoff to delay retransmissions.

`DHCPFORCERENEW` and `DHCPLEASEQUERY` are relatively new additions to DHCP and not all clients and servers include code that implements these messages. Clients and servers that do not have code for `DHCPFORCERENEW` and `DHCPLEASEQUERY` will simply discard those messages without responding to the sender. Servers and clients sending `DHCPFORCERENEW` and `DHCPLEASEQUERY` messages should take into account that they might never receive responses from some recipients of these messages and compensate by reducing the number and frequency of retransmissions.

Avoiding Message Collisions

Under some circumstances, a DHCP server may receive incoming messages faster than it can respond to them. It is easy to imagine a scenario in which this might happen: For example, if power fails in a building, then when the power comes back on, all the computers in that building might restart at the same time. Computers that have DHCP clients initiate their DHCP protocol exchanges at almost the same time. If a network segment has many identical or similar machines, many messages will be delivered to a server at once.

DHCP includes two mechanisms to decrease the likelihood of overloading a server with messages from clients. First, a client is required to delay its initial DHCP message by a random time between 0 and 10 seconds. Second, a DHCP client adjusts the delay time (in seconds) between retransmissions of a message with a random value in the range of –1 to +1 to avoid synchronization with other clients.

Transaction IDs

A DHCP client inserts a 32-bit identification number, or *transaction ID,* in the `xid` field of every DHCP message. The DHCP specification gives the client significant freedom in choosing transaction IDs; the goal is to minimize the chance that two DHCP clients will use the same transaction ID simultaneously. The client can reuse the same transaction ID when it retransmits a DHCP message, or it can choose a different transaction ID for retransmissions. Similarly, the client can choose subsequent transaction IDs sequentially, starting with an initial random transaction ID, or it can choose each transaction ID at random.

The server copies the transaction ID from incoming DHCP messages into the corresponding responses. A client then matches the transaction ID in a server response to the transaction ID from the client's most recently transmitted message. If the transaction IDs match, the client accepts the response to its previous message; otherwise, the client discards the message and continues waiting for the correct response. Inappropriate delivery of a server response to a client may occur, for example, if a client sets the broadcast flag to `1`, requesting that responses be broadcast. These responses are delivered to every DHCP client on the network segment and must be filtered out by the clients that did not send the original message.

NOTE

The ISC DHCP client and other DHCP clients check the returned client link-layer address and the contents of the `xid` field to further reduce the likelihood of a collision.

Other Transmission Methods

A DHCP client uses broadcast and relay agents to send messages to DHCP servers when the client does not have an IP address. When the client has an address and knows the address of a DHCP server, it uses unicast and transmits messages directly to servers through the normal IP datagram delivery service. This section discusses how DHCP messages are delivered. The use of these messages is discussed in more detail in Chapter 8, "DHCP Message Exchanges."

`DHCPREQUEST` Messages

DHCP uses `DHCPREQUEST` messages at four stages of a lease's lifetime:

- During the initial selection of the lease
- When confirming the validity of an IP address after a restart
- When the lease is renewed at specific points during its lifetime
- When the lease is rebound near the end of its lifetime

When the `DHCPREQUEST` message is sent during the initial selection phase, it indicates which lease offer the client has selected and implicitly indicates that addresses offered by servers the client has not selected may be reclaimed for assignment to other clients. This `DHCPREQUEST` message is broadcast and must use the `server identifier` option to indicate which server it has selected. It can also include other options that specify desired configuration values. The `requested IP address` option must be set to the value of `yiaddr` in the `DHCPOFFER` message from the server.

This broadcast DHCPREQUEST message can be relayed through DHCP/BOOTP relay agents. It is important that BOOTP relay agents forward the DHCPREQUEST message to the same set of DHCP servers that received the original DHCPDISCOVER message. To help ensure that they do, the DHCPREQUEST message uses the same value that's in the DHCP message header's `secs` field and is sent to the limited broadcast IP address.

A DHCP client also broadcasts a DHCPREQUEST message when it restarts or reconnects to the network, if it still holds a valid lease on an IP address. This DHCPREQUEST message contains the client's current IP address, which is to be confirmed by the DHCP server, but does not include the `server identifier` option.

NOTE

Some relay agents take advantage of the ability to use more than one DHCP server to provide backup reliability and load balancing by selectively forwarding DHCP messages to different servers. For example, a relay agent might forward messages that have the `secs` field set to `0` to a primary server and other messages to a backup server. To ensure that messages are forwarded to the correct server, the client must set the `secs` field in the DHCPREQUEST message to the same value used in the DHCPDISCOVER message.

A DHCP client sends a DHCPREQUEST message to request an extension of the client's lease on an IP address. The client sends the extension request to the IP address of the server from which the address was originally obtained (the client gets the server's IP address from the `server identifier` option when it receives it in a DHCPACK message). The client uses the client's own valid IP address as the source address in the message.

If the server from which the client requests a lease extension has moved to a new network or is temporarily shut down, the client will not receive a response from the server. In this case, the client sends a DHCPREQUEST message to 255.255.255.255.The message is delivered to every DHCP server that is providing service on the client's network segment. Because the client still has a valid IP address at this time, it uses that address as the source address in the message.

DHCPINFORM Messages

A client that does not need to obtain an address through DHCP uses the DHCPINFORM message. Computers with manually configured IP addresses, such as large systems or mission-critical servers, can use DHCPINFORM to obtain other network information, such as the location of print servers, time servers, name servers and so on. The client can either use unicast to send a message to a server known by the client or use broadcast to send the message to all available servers. In either case, the client uses its own IP address as the source address in the message.

DHCPRELEASE Messages

Clients also use unicast to deliver DHCPRELEASE messages to the server from which the address was originally obtained. The client puts the address it is releasing in the ciaddr field. However, because the DHCP specification specifies that the client terminates its lease on an address when it first transmits the DHCPRELEASE message, the server cannot deliver a response to the client, which no longer has a valid IP address. Thus, servers do not respond to DHCPRELEASE messages, and clients do not wait for responses to DHCPRELEASE messages nor do clients retransmit DHCPRELEASE messages.

DHCPFORCERENEW Messages

The DHCPFORCERENEW message is the only DHCP message that is sent by the server without a message being sent by the client. A server can send a DHCPFORCERENEW message to any DHCP client that has an IP address assigned to it through DHCP. Because the client is guaranteed to have an IP address, the server uses the client's address as the destination address in a unicast IP datagram.

DHCPLEASEQUERY Messages

The DHCPLEASEQUERY message is typically sent by a device or an application rather than from a DHCP client. Because the device or computer on which the requesting application is running must already have an IP address, the sender transmits the DHCPLEASEQUERY message to the server in a unicast IP datagram.

Authenticated DHCP Messages

In some situations, network administrators and users might want to reliably identify the other participants in DHCP message exchanges. For example, a cable system subscriber might want to be sure that she trusts the DHCP server from which her computer is obtaining configuration information. The administrator of a wireless LAN in a university library might want to limit access to the network by restricting the assignment of IP addresses to computers that have been identified as authorized to use the university network.

RFC 3118 defines an extension to DHCP through which DHCP clients and servers can authenticate the identity of other DHCP participants and can verify that the content of a DHCP message has not been changed during delivery through the network. Using the authentication mechanisms described in RFC 3118, a client can verify that a DHCP server can be trusted to provide valid configuration information. A DHCP server can use the mechanisms in RFC 3118 to decide whether a request for DHCP information comes from a client that is authorized to use the network.

Protocol Design

The design goals for the authentication mechanism in DHCP are as follows:

- Provide a mechanism through which clients and servers can reliably identify each other and confirm that the contents of DHCP messages were not altered while in transit
- Avoid changing the current protocol and maintain backward compatibility with existing clients, servers, and relay agents
- Allow for multiple authentication mechanisms and algorithms
- Allow for automated selection of authentication tokens
- Minimize manual configuration of DHCP clients

The authentication mechanism and the `authentication` option are a framework for the definition of multiple authentication protocols. The `authentication` option carries an identification of the protocol and the algorithm used within the option. The authentication mechanism also includes additional information to prevent spoofing of identities through replay of authenticated DHCP messages.

RFC 3118 defines two authentication mechanisms: a simple plain-text token that is used to identify clients and servers and a keyed-hashing technique that is based on *Hashed Message Authentication Code* (HMAC; see RFC 2104). These two authentication mechanisms use the same format for the `authentication` option, as shown in Figure 7.5.

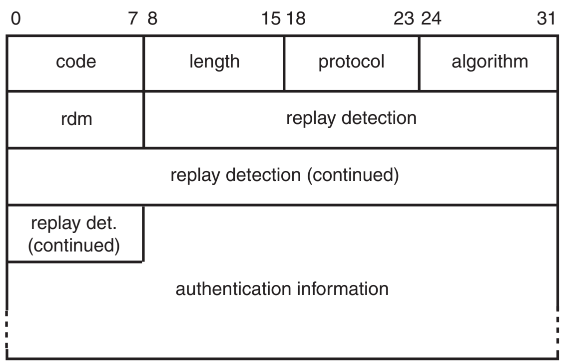

FIGURE 7.5 The format of the DHCP `authentication` option.

The difference between the two mechanisms is in the contents of the `authentication information` field. The fields in the `authentication` option are described in Table 7.1.

TABLE 7.1 Fields in the DHCP `authentication` Option

Field	Description
`code`	90
`length`	Number of bytes in the `protocol`, `algorithm`, `replay detection method`, `replay detection`, and `authentication information` fields
`protocol`	The protocol in use in this option, which can have the following values: `0`: authentication token (see the section "The Authentication Token Protocol") `1`: delayed authentication (see the section "Delayed Authentication Protocol")
`algorithm`	The specific algorithm used with the protocol from the `protocol` field; interpreted according to the definition of the protocol
`replay detection method` (RDM)	The method used to detect replay of DHCP messages
`replay detection`	The information used to detect a replay
`authentication`	Additional information as required by the `information` protocol and algorithm; interpreted according to the definition of the protocol

The Authentication Token Protocol

The Authentication Token Protocol provides minimal identification of DHCP clients and servers. The protocol is intended for protection against DHCP servers that are inadvertently started or incorrectly configured and for simple segregation of clients and servers in shared networks. The Authentication Token Protocol provides no defense against an active intruder who can simply examine local network traffic to determine the appropriate authentication token to use to gain unauthorized service.

The protocol number for the Authentication Token Protocol is 0, and the `algorithm` field must be set to `0`. The `replay detection method` field is set to `0`, indicating that the replay detection is a simple increasing counter. Each message must have a counter value that is strictly greater than that of the previous message. For example, the time of day in NTP format (Network Time Protocol; see RFC 1305) might be used as the replay detection information. The receiver checks each message and discards any messages whose replay detection value is not strictly greater than the value from the previous message.

The authentication information is a simple plain-text string. For example, it might be the name of the server or of the organization that manages the server. The format of the `authentication` option when used to carry the Authentication Token Protocol is shown in Figure 7.6.

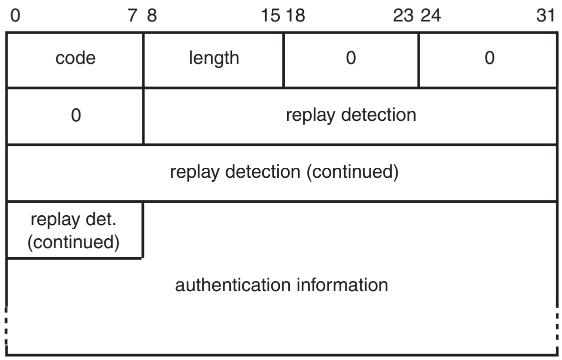

FIGURE 7.6 The format of the DHCP Authentication Token protocol.

The Delayed Authentication Protocol

The second protocol defined in RFC 3188 is called Delayed Authentication Protocol. In this protocol, the authentication is delayed until the server sends a `DHCPOFFER` message in response to a client's `DHCPDISCOVER` message. The delay enables the server to announce to the client what algorithm and key the server will accept, without requiring the client to divulge any information.

This protocol assumes that DHCP clients and servers are provided with a shared secret key through some mechanism that is independent of the authentication protocol. The mechanism enables the use of multiple keys so that a mobile DHCP client that might frequently contact different DHCP servers can use a different key for each DHCP server it knows about.

Authentication Option Formats

A client first sends a `DHCPDISCOVER` message with an `authentication` option, requesting use of the Delayed Authentication Protocol. Figure 7.7 illustrates the format of the `delayed authentication` option in a `DHCPDISCOVER` message.

The `algorithm` field specifies the algorithm to be used to generate the authentication information in subsequent messages. RFC 3118 defines a single algorithm, which is explained in more detail in the following section that is identified by the value `1` in the `algorithm` field. The `RDM` field specifies the replay detection method to use. Delayed Authentication Protocol uses the same counter mechanism for replay detection as the authentication token protocol.

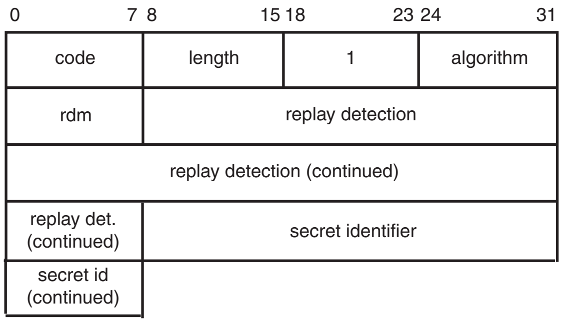

FIGURE 7.7 The format of the `delayed authentication` option in a `DHCPDISCOVER` message.

Subsequent messages between the client and server contain the `authentication` option in the format shown in Figure 7.8. In this version of the `authentication` option, `secret ID` is an identifier for the secret the sender uses to generate the *message authentication code* (MAC). The secret identifier enables the client and server to agree on the use of one of possibly several shared secrets. The `MAC` field is generated from the contents of the DHCP message, using the HMAC and MD5 algorithms (see RFC 1321).

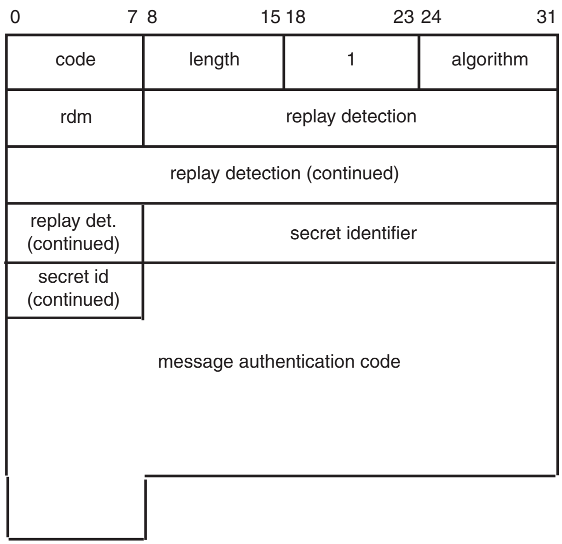

FIGURE 7.8 The format of the `delayed authentication` option in subsequent messages.

Computing the `MAC` Field

The sender computes the `MAC` field for the `delayed authentication` option by using the HMAC and MD5 algorithms. The entire UDP payload of the DHCP message, with two exceptions, is used as input to the HMAC-MD5 algorithm. Because the `giaddr` and `hops` fields may be altered by a relay agent, those fields are not included in the MAC, and their contents are set to `0` for computation by the MAC. In addition, if a `relay agent information` option appears at the end of the DHCP packet, the bytes in this option are not included in the HMAC computation.

The `secret ID` field of the `delayed authentication` option is set to the identifier of the shared secret that the sender uses to generate the MAC. The `RDM` field is set to `0`, and the replay detection field is set to a 64-bit monotonically increasing counter. The current time of day, in NTP format (see RFC 1305), is a good value for the `counter` field.

Validating a Message

To validate an incoming message, the receiver first checks that the value in the `replay detection` field is greater than the value from the previous message, and it discards any messages that fail this test. Next, the receiver uses the contents of the `secret ID` field from the `delayed authentication` option to identify the key used to generate the MAC in the message. The receiver then computes the MAC for the message by using the algorithm described in the previous section. It sets the contents of the `MAC` field in the `authentication` option, the `giaddr` and the `hops` fields in the `fixed-format` section of the message to `0` for the computation, and it ignores the `relay agent information` option if one exists. If the MAC value the receiver computes does not match the contents of the `MAC` field in the `authentication` option, the receiver discards the message. The identification and authentication in the Delayed Authentication Protocol are based on the assumption that the receiver and the sender of a DHCP message are the only two DHCP participants that know the shared secret identified by the `secret ID` field in the message. If the receiver successfully validates the incoming message, the receiver can infer that the message was sent by the sender identified in the message because only that sender knows the key used to generate the MAC value in the message. The receiver can also infer that the content of the message was not altered in transmission because the key would be required to recompute a new MAC value to match the contents of the message after any changes were made.

Using Delayed Authentication When Obtaining a New IP Address

When the `delayed authentication` option is used while obtaining a new IP address, the client uses the option format shown in Figure 7.7. At present, only one algorithm is defined, and the only valid value for the `algorithm` field is `1`, which selects the HMAC-MD5 MAC computation algorithm.

After receiving the `delayed authentication` option in the `DHCPDISCOVER` message, a server selects a key and composes the `delayed authentication` option to insert in its `DHCPOFFER` response. The server must be configured with enough information so that it can select a key and an identifier for the key for clients with which it has not communicated previously. When the server selects a key and an identifier for the client, it records that information along with any other information it keeps about the client. The server computes the MAC for the `DHCPOFFER` message according to the procedure described in the section, "Computing the `MAC` Field," and inserts the MAC to the client as the `MAC` field in the `authentication` option. The server then sends the `DHCPOFFER` message to the client.

The client validates incoming messages as described in the section, "Validating a Message," and discards any messages that do not pass the validation tests. The client then chooses one of the `DHCPOFFER` messages, and it looks up the secret key identified in the message in its local database of secret keys. The client composes a `DHCPREQUEST` message with a `delayed authentication` option that contains a value for the `MAC` field computed by the secret key used for the selected server. Finally, the client sends the `DHCPREQUEST` message as specified in RFC 2131.

Any server that receives the `DHCPREQUEST` message validates the incoming message. The server selected in the `DHCPREQUEST` message constructs a `DHCPACK` message that contains all the options required by RFC 2131. The server includes the `authentication` option in the `DHCPACK` message, which is composed as described previously. The server sends the `DHCPACK` message to the client, which validates the incoming `DHCPACK` message, extracts the assigned IP address and other configuration parameters, and uses those configuration parameters to configure its protocol stack.

NOTE

The sequence of messages exchanged between the client and server for authenticated DHCP is the same as the sequence described in RFC 2131. This enables backward compatibility with clients and servers that do not include an implementation of the `authentication` option, and it minimizes the impact of the `authentication` option on the DHCP specification.

Using Delayed Authentication When Confirming an IP Address

When confirming an IP address, the client uses the same secret it used when it obtained its configuration information in the `INIT` state to compose an `authentication` option to include with its `DHCPREQUEST` message. The client then sends the `DHCPREQUEST` message.

The client performs the validation test on responses it receives and discards messages that fail. Messages that pass the validation test are processed as specified in RFC 2131:

- A DHCPACK message confirms that the client can continue using its address, and the client uses any configuration parameters from the DHCPACK message.
- A DHCPNAK message forces the client into the INIT state.
- If the client receives no responses that pass the validation test, it can continue to use its previous address until the lease on that address expires.

Using Delayed Authentication When Extending the Lease on an IP Address

At the time specified to extend the lease on its address, the client composes a DHCPREQUEST message and includes the authentication option with the MAC value computed using the secret it recorded when it initially obtained the address. The client sends the DHCPREQUEST message and performs the validation test on responses it receives. The client discards messages that fail the validation test.

If the client receives a DHCPACK message that passes the validation test, it uses the configuration information from the message to configure its protocol stack. If the client receives no responses or none of the received responses passes the validation test, the client behaves as though it received no responses to its DHCPREQUEST message.

Summary

DHCP uses UDP to transmit protocol messages. To deliver messages from a client that doesn't have an IP address, DHCP specifies the use of the limited broadcast IP address, 255.255.255.255, and the "this host" IP address, 0.0.0.0, in DHCP messages. When a client has an IP address and knows the address of a server, it uses unicast to transmit messages to the server.

Broadcast messages from a DHCP client can be delivered only to servers on the same network segment. DHCP uses relay agents to forward messages between clients and servers on different network segments. Relay agents are often implemented in routers. DHCP's design ensures that the relay agent is essentially stateless because it need not store information about messages it forwards.

DHCP clients are responsible for reliable delivery of protocol messages. Clients use responses from servers as acknowledgments of receipt and retransmit messages for which responses are not received. Clients use randomized exponential backoff to determine how long to wait before retransmitting a lost message. This retransmission strategy reduces network congestion if many DHCP clients are on a network segment and smoothes the load on a server that serves a large network with a heavy client load.

RFC 3118 defines an authentication mechanism for DHCP messages. DHCP clients and servers can use authenticated DHCP messages to reliably identify each other and to avoid denial of service and other attacks through DHCP.

8

DHCP Message Exchanges

IN THIS CHAPTER

- Client States
- Working with Multiple Servers
- Other Message Exchanges

Chapter 2, “An Example of DHCP in Operation,” and Chapter 3, “Configuring the DHCP Server,” explain how DHCP clients and servers communicate. Chapter 7, “Transmitting DHCP Messages,” describes how DHCP messages are transmitted by using UDP. This chapter takes a detailed look at the DHCP messages the protocol exchanges with clients and servers. It covers the typical life cycle of a client: from initial configuration, restarting, and reconfiguring after moving to a new network segment, to notifying the server that it is leaving the network.

The examples in this chapter are based on the Generic Startup, Inc. (GSI) network described in Chapter 2, which includes one DHCP server and several DHCP clients. The client and the server in these examples are both on the same subnet, which is assigned the network address 192.168.11.0. The description of each message exchange includes packet traces of the messages. The packet traces were generated with the network analysis tool `snoop`, which is available with Sun’s Solaris operating system, and were edited to delete extraneous information and to focus on the parts of the messages relevant to DHCP.

This chapter concentrates on the messages themselves, simply describing the decisions that a DHCP server makes when interacting with a DHCP client. DHCP server address, leasing, and configuration policies are covered in Chapter 15, “Configuring a DHCP Server,” and Chapter 16, “Client Identification and Fixed-Address Allocation.”

Client States

DHCP client operation is specified in RFC 2131 as a *state machine*—a set of possible states and a list of inputs for each state that result in transition to a different states. RFC 2131 also provides a *state transition diagram* to illustrate DHCP client behavior. This section introduces the various client states. A closer look at the way a client behaves, along with details regarding state transitions, is provided later in this chapter.

When a client does not have a valid IP address, it is said to be in the `INIT` state. Figure 8.1 illustrates the client states and state transitions. During the initial configuration process, the client normally moves to the `SELECTING` state; when it is successfully configured with an IP address, it moves to the `BOUND` state. When the client restarts, it goes to the `INIT-REBOOT` state, and after it confirms that its IP address is still valid, it moves to the `BOUND` state. If a server sends a `DHCPNAK` message to the client, the client reverts to the `INIT` state.

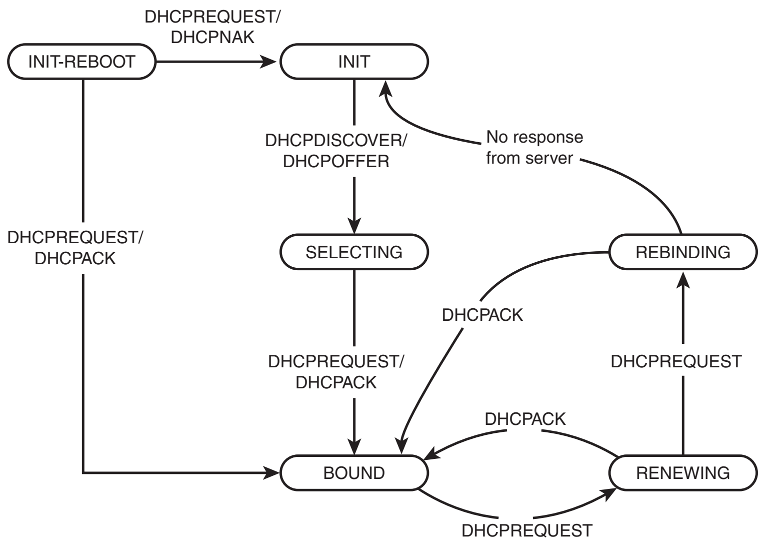

FIGURE 8.1 A finite state machine for a DHCP client.

NOTE

The DHCP specification in RFC 2131 does not explicitly describe the behavior of a client that restarts with an expired lease on its most recent IP address. RFC 2131 simply states that a client goes to the `INIT-REBOOT` state if it restarts with a previously assigned address. It does not specifically require that the lease on the address has not expired.

Section 3.2 of RFC 2131 specifies that a client can use a previously assigned address if it doesn't receive a confirming `DHCPACK` message from a DHCP server in response to its `DHCPREQUEST` message. This behavior implies that the lease on the previously unassigned address has not expired.

Some clients go to `INIT-REBOOT` state with a previously assigned address, regardless of the state of the lease on that address. A client implementing this behavior sends an initial `DHCPREQUEST` message and proceeds to the `BOUND` state if it receives a `DHCPACK` message in response. If the client does not receive a `DHCPACK` message and the lease has not expired, the client uses its previously assigned address. If the lease has expired, the client either reverts to the `INIT` state or abandons DHCP initialization.

Some time before the client's lease on the IP address expires, the client enters the `RENEWING` state and attempts to extend its lease by sending a unicast message to the server from which it obtained its IP address. The client waits for some time, and if it receives no response to its unicast renewal request, it enters the `REBINDING` state and broadcasts a message to extend its lease from any available server. If the lease expires without the client successfully renewing its lease, the client reverts to the `INIT` state.

Obtaining an Initial Configuration

When a computer configured to use DHCP is connected to a network, it determines whether it has a valid IP address. A client may be without a valid address because it is new and has not had an IP address assigned to it, because the lease on its previous address expired, or because a server told the client that its IP address is invalid. In these cases, the client is in the `INIT` state because it does not have a valid address.

When the client starts in the `INIT` state, the client and server exchange four messages through which the client locates available DHCP servers, and a DHCP server assigns an address and other configuration to the client. The next sections describe these messages in detail.

The `DHCPDISCOVER` Message

To obtain an IP address and other configuration parameters, the client finds a DHCP server or servers. The client broadcasts a `DHCPDISCOVER` message, and the message is delivered to all the DHCP servers on the same network segment as the client. The `DHCPDISCOVER` message is also received by relay agents on the client's network segment and forwarded to other DHCP servers on other networks. Example 8.1 is based on the network configuration in Figure 2.3 in Chapter 2.

Example 8.1 shows the output from `snoop` looking at the `DHCPDISCOVER` message from `desktop1`.

Example 8.1

```
ETHER:  ----- Ether Header -----
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:7c:fb:89, Sun
IP:   ----- IP Header -----
IP:   Protocol = 17 (UDP)
IP:   Source address = 0.0.0.0, OLD-BROADCAST
IP:   Destination address = 255.255.255.255, BROADCAST
UDP:  ----- UDP Header -----
UDP:  Source port = 68
UDP:  Destination port = 67 (BOOTPS)
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP: Transaction ID = 0xc8206f1c
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 0.0.0.0
DHCP: Client hardware address (chaddr) = 08:00:20:7C:FB:89
DHCP: ----- (Options) field options -----
DHCP: Message type = DHCPDISCOVER
```

As described in Chapter 7, this message is sent to the Ethernet broadcast address FF:FF:FF:FF:FF:FF and to the IP limited broadcast address 255.255.255.255. The client uses the IP address 0.0.0.0 as the UDP source address and in the `ciaddr` address. The `DHCP message type` option identifies this message as a `DHCPDISCOVER` message, and the client does not include additional DHCP options.

The `DHCPOFFER` Message

After the server receives the `DHCPDISCOVER` message from the client, it finds an address to assign to the client and puts it in a `DHCPOFFER` message. The server also includes in the `DHCPOFFER` message other configuration parameters for the client, as defined by the server's configuration file. After the server has completed the `DHCPOFFER` message, it sends the message back to the client.

Example 8.2 shows the `DHCPOFFER` message that is sent in response to the previous `DHCPDISCOVER` message.

Example 8.2

```
ETHER:  ----- Ether Header -----
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:76:f:8, Sun
IP:   ----- IP Header -----
IP:   Protocol = 17 (UDP)
IP:   Source address = 192.168.11.252
```

Example 8.2 Continued

```
IP:   Destination address = 255.255.255.255, BROADCAST
UDP:  ----- UDP Header -----
UDP:  Source port = 67
UDP:  Destination port = 68 (BOOTPC)
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP: Transaction ID = 0xc8206f1c
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 192.168.11.25
DHCP: Client hardware address (chaddr) = 08:00:20:7C:FB:89
DHCP: ----- (Options) field options -----
DHCP: Message type = DHCPOFFER
DHCP: DHCP Server Identifier = 192.168.11.252
DHCP: IP Address Lease Time = 2592000 seconds
DHCP: Renewal (T1) Time Value = 1296000 seconds
DHCP: Rebinding (T2) Time Value = 2268000 seconds
DHCP: Subnet Mask = 255.255.255.0
```

This message is sent to Ethernet broadcast address ff:ff:ff:ff:ff:ff and to the newly assigned IP address 192.168.11.25. The server copies the transaction identifier, $c8206f1c_{16}$, from `desktop1`'s `DHCPDISCOVER` message, so the client can identify the `DHCPOFFER` response. The server gives the client a lease time of 30 days and indicates that the client should try to extend its lease after 15 days.

The `DHCPREQUEST` Message

After `desktop1` receives the `DHCPOFFER` message from `dhcpserve`, it sends a `DHCPREQUEST` message, asking for the configuration information from `dhcpserve`. The `DHCPREQUEST` message is shown in Example 8.3.

Example 8.3

```
ETHER:  ----- Ether Header -----
ETHER:  Destination = FF:FF:FF:FF:FF:FF, (broadcast)
ETHER:  Source      = 8:0:20:7c:fb:89, Sun
IP:   ----- IP Header -----
IP:   Protocol = 17 (UDP)
IP:   Source address = 0.0.0.0, OLD-BROADCAST
IP:   Destination address = 255.255.255.255, BROADCAST
UDP:  ----- UDP Header -----
UDP:  Source port = 68
UDP:  Destination port = 67 (BOOTPS)
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP: Transaction ID = 0xc8206f1d
```

Example 8.3 Continued

```
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 0.0.0.0
DHCP: Client hardware address (chaddr) = 08:00:20:7C:FB:89
DHCP: ----- (Options) field options -----
DHCP: DHCP Server Identifier = 192.168.11.252
DHCP: IP Address Lease Time = 2592000 seconds
DHCP: Renewal (T1) Time Value = 1296000 seconds
DHCP: Rebinding (T2) Time Value = 2268000 seconds
DHCP: Subnet Mask = 255.255.255.0
DHCP: Message type = DHCPREQUEST
DHCP: Requested IP Address = 192.168.11.25
```

In this `DHCPREQUEST` message, `desktop1` asks for the address and other configuration parameters that `dhcpserve` supplied in the `DHCPOFFER` message. `desktop1` uses a new transaction identifier, and broadcasts the message using the FF:FF:FF:FF:FF:FF link-layer and 255.255.255.255 IP-layer broadcast addresses.

The `DHCPACK` Message

After receiving the `DHCPREQUEST` message, `dhcpserve` checks the requested address and configuration parameters to ensure that the address is still available and the parameters are correct. `dhcpserve` records the assigned address and sends the `DHCPACK` message shown in Example 8.4 to `desktop1`.

Example 8.4

```
ETHER:  ----- Ether Header -----
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:76:f:8, Sun
IP:   ----- IP Header -----
IP:   Source address = 192.168.11.252
IP:   Destination address = 255.255.255.255, BROADCAST
UDP:  ----- UDP Header -----
UDP:  Source port = 67
UDP:  Destination port = 68 (BOOTPC)
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP: Transaction ID = 0xc8206f1d
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 192.168.11.25
DHCP: Client hardware address (chaddr) = 08:00:20:7C:FB:89
DHCP: ----- (Options) field options -----
DHCP: Message type = DHCPACK
DHCP: DHCP Server Identifier = 192.168.11.252
```

Example 8.4 Continued

```
DHCP: IP Address Lease Time = 2592000 seconds
DHCP: Renewal (T1) Time Value = 1296000 seconds
DHCP: Rebinding (T2) Time Value = 2268000 seconds
DHCP: Subnet Mask = 255.255.255.0
```

Figure 8.2 shows a time line of the messages exchanged by `desktop1` and `dhcpserve` as the client obtains its initial address.

When `desktop1` receives this `DHCPACK` message, it records the assignment information and configures its TCP/IP software with the IP address and other parameters. Then, `desktop1` can begin using TCP/IP.

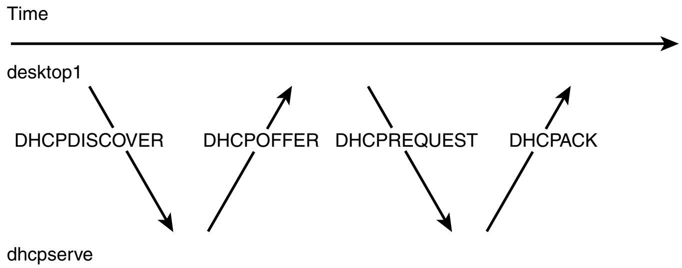

FIGURE 8.2 A time line of messages exchanged between a client and a server to assign an initial address.

Confirming an IP Address When Restarting

Every time `desktop1` restarts—for example, after it is powered on—it checks whether it has recorded an address with a lease that has not expired. After `desktop1` is installed and assigned its initial address, it typically has an address it can reuse. If `desktop1` has an IP address whose lease has not expired, it goes into `INIT-REBOOT` state and attempts to confirm that its address is still valid.

The `INIT-REBOOT DHCPREQUEST` Message

The client sends the IP address to be confirmed in a `DHCPREQUEST` message, which is received and checked by all DHCP servers that are configured for the network segment to which the client is attached. Example 8.5 shows a `DHCPREQUEST` message sent by a client in the `INIT-REBOOT` state.

Example 8.5

```
ETHER:  ----- Ether Header -----
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:7c:fb:89, Sun
IP:   ----- IP Header -----
IP:   Protocol = 17 (UDP)
IP:   Source address = 0.0.0.0, OLD-BROADCAST
IP:   Destination address = 255.255.255.255, BROADCAST
UDP:  ----- UDP Header -----
UDP:  Source port = 68
UDP:  Destination port = 67 (BOOTPS)
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP: Transaction ID = 0xc8206f1f
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 0.0.0.0
DHCP: Client hardware address (chaddr) = 08:00:20:7C:FB:89
DHCP: ----- (Options) field options -----
DHCP: IP Address Lease Time = 2592000 seconds
DHCP: Renewal (T1) Time Value = 1296000 seconds
DHCP: Rebinding (T2) Time Value = 2268000 seconds
DHCP: Subnet Mask = 255.255.255.0
DHCP: Message type = DHCPREQUEST
DHCP: Requested IP Address = 192.168.11.25
```

This message is broadcast because `desktop1`'s address may be invalid, even if the lease on that address has not expired. For example, if `desktop1` moves to a new office, or if the network architect assigns a new address to the network to which `desktop1` is attached, `desktop1` has an address that does not match the local IP subnet.

When `dhcpserve` receives `desktop1`'s `DHCPREQUEST` message, it extracts `desktop1`'s requested address from the `options` section and checks that the address is from an IP subnet assigned to the network segment to which `desktop1` is attached. Unless `desktop1` has been moved, the address is usually correct, so `desktop1` is allowed to use it.

The `DHCPACK` Message

`dhcpserve` replies to `desktop1` with a `DHCPACK` message. `dhcpserve` puts all the configuration parameters in the response message and returns the message to `desktop1`. Example 8.6 shows the details of a `DHCPACK` message.

Example 8.6

```
ETHER:  ----- Ether Header -----
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:76:f:8, Sun
IP:   ----- IP Header -----
IP:   Source address = 192.168.11.252
IP:   Destination address = 255.255.255.255, BROADCAST
UDP:  ----- UDP Header -----
UDP:  Source port = 67
UDP:  Destination port = 68 (BOOTPC)
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP: Transaction ID = 0xc8206f1f
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 192.168.11.25
DHCP: Client hardware address (chaddr) = 08:00:20:7C:FB:89
DHCP: ----- (Options) field options -----
DHCP: Message type = DHCPACK
DHCP: DHCP Server Identifier = 192.168.11.252
DHCP: IP Address Lease Time = 2592000 seconds
DHCP: Renewal (T1) Time Value = 1296000 seconds
DHCP: Rebinding (T2) Time Value = 2268000 seconds
DHCP: Subnet Mask = 255.255.255.0
```

When `desktop1` receives the `DHCPACK` message, it uses the parameters from the message to set its IP address and protocol software configuration. Then it is ready to use the network. Figure 8.3 shows a time line of the messages exchanged between `desktop1` and `dhcpserve` as the client rechecks its IP address when restarting.

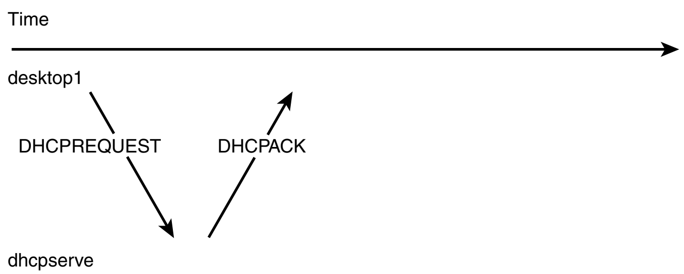

FIGURE 8.3 A time line of messages exchanged between a client and a server when the client restarts.

If `desktop1` receives no response to its broadcast message when it first starts up, it may be that DHCP servers are inaccessible due to a power outage or a temporary network problem. Rather than keep DHCP clients from using the network if the servers do not respond, a client can use its previous address if the lease on that address is still valid.

Extending a Lease

Suppose `desktop1` continues running without being turned off or restarting. Eventually, the lease on the address assigned to `desktop1` will run out. The leasing mechanism is designed to give DHCP servers a reliable way to reclaim unused addresses, not to take away addresses from computers that are still in use. DHCP provides a way for a computer to extend the lease on its address without interrupting network use.

A DHCP client extends its lease by sending a message to a server, requesting more time on the lease. The request for an extension is sent in a `DHCPREQUEST` message, and the client can ask for a lease of whatever length it chooses. At this point, the DHCP server decides how long an extension to grant and returns the new lease duration to the client in a `DHCPACK` message.

The choice of lease length is up to the server. In fact, the server can choose not to extend the lease or it can ignore lease extension requests altogether.

The Lease Extension Request

A DHCP client is said to be in the `RENEWING` state when it begins asking for a lease extension. Example 8.7 shows an example of a `DHCPREQUEST` message from `desktop1`, asking for an extension of its current lease on 192.168.11.0. Note that the message is sent directly to `dhcpserve` and includes the length of the extension `desktop1` wants as well as the other protocol parameters that `desktop1` is using.

Example 8.7

```
ETHER:  ----- Ether Header -----
ETHER:  Destination = 8:0:20:76:f:8, Sun
ETHER:  Source      = 8:0:20:7c:fb:89, Sun
IP:   ----- IP Header -----
IP:   Protocol = 17 (UDP)
IP:   Source address = 192.168.11.25
IP:   Destination address = 192.168.11.252
UDP:  ----- UDP Header -----
UDP:  Source port = 68
UDP:  Destination port = 67 (BOOTPS)
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP: Transaction ID = 0xc8206f1c
```

Example 8.7 Continued

```
DHCP: Client address (ciaddr) = 192.168.11.25
DHCP: Your client address (yiaddr) = 192.168.11.25
DHCP: Client hardware address (chaddr) = 08:00:20:7C:FB:89
DHCP: ----- (Options) field options -----
DHCP: DHCP Server Identifier = 192.168.11.252
DHCP: IP Address Lease Time = 2592000 seconds
DHCP: Renewal (T1) Time Value = 1296000 seconds
DHCP: Rebinding (T2) Time Value = 2268000 seconds
DHCP: Subnet Mask = 255.255.255.0
DHCP: Message type = DHCPREQUEST
```

The Lease Extension Response

After receiving the `DHCPREQUEST` message from `desktop1`, the server confirms that its configured lease policy allows it to extend the lease on the address assigned to `desktop1`. If the lease can be extended, the server determines the appropriate configuration parameters for `desktop1` and sends a `DHCPACK` message with options for those parameters. Example 8.8 shows `dhcpserve`'s response to `desktop1`'s request to extend the lease.

Example 8.8

```
ETHER:  ----- Ether Header -----
ETHER:  Destination = 8:0:20:7c:fb:89, Sun
ETHER:  Source      = 8:0:20:76:f:8, Sun
IP:   Protocol = 17 (UDP)
IP:   Source address = 192.168.11.252
IP:   Destination address = 192.168.11.25
UDP:  ----- UDP Header -----
UDP:  Source port = 67
UDP:  Destination port = 68 (BOOTPC)
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP: Transaction ID = 0xc8206f1c
DHCP: Client address (ciaddr) = 192.168.11.25
DHCP: Your client address (yiaddr) = 192.168.11.25
DHCP: Client hardware address (chaddr) = 08:00:20:7C:FB:89
DHCP: ----- (Options) field options -----
DHCP: Message type = DHCPACK
DHCP: DHCP Server Identifier = 192.168.11.252
DHCP: IP Address Lease Time = 2592000 seconds
DHCP: Renewal (T1) Time Value = 1296000 seconds
DHCP: Rebinding (T2) Time Value = 2268000 seconds
DHCP: Subnet Mask = 255.255.255.0
```

In this response, `dhcpserve` agrees to `desktop1`'s request for an extension of 2592000 seconds. When `desktop1` receives the `DHCPACK` message from `dhcpserve`, it records the lease duration and other parameters from the message. Figure 8.4 shows the time line of these messages.

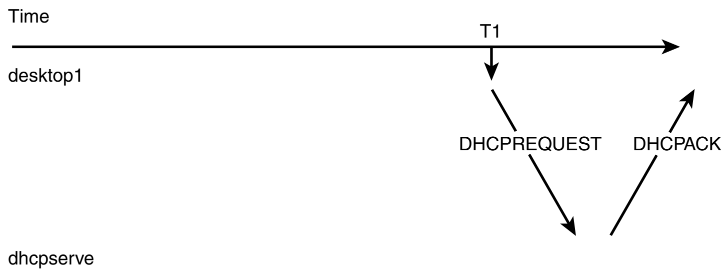

FIGURE 8.4 A time line of messages exchanged by a client and a server to extend a lease.

The lease is extended without interrupting other applications. This process does not affect the user; as long as a DHCP server that can extend the lease is available, a DHCP client can run indefinitely without any impact from the DHCP leasing mechanism.

The time at which `desktop1` begins asking to extend its lease is called *T1*. The DHCP server can explicitly tell a client when to extend its lease by configuring T1, which is also known as the *renewal time*. If the server does not give the client a value for T1, it defaults to one-half the original lease.

In Example 8.8, if `desktop1` does not receive a response to its lease extension request, it retransmits the request according to the rules described in Chapter 7. As long as the lease extension is received before the lease expires, `desktop1` can continue using the network.

Extending a Lease from a Different Server

If a client in the `RENEWING` state fails to contact its original DHCP server, the client enters the `REBINDING` state, called *T2,* at a later time.

In the `REBINDING` state, the client broadcasts its `DHCPREQUEST` messages to all DHCP servers, rather than unicasting them only to the server that gave it its current lease. At this point, if another server receives the `DHCPREQUEST` message and can extend the lease, it does so, and the client can continue operating with its lease.

NOTE

This mechanism for finding alternate servers for extending a lease was included in DHCP to enhance reliability. If the server from which a DHCP client received its lease is unavailable for an extended period, the client automatically locates another server. If the DHCP server that granted a client its lease is moved to a new computer system, with a new IP address, the client automatically finds the server at its new address when it cannot contact it at the old address.

Using alternate DHCP servers implies that the servers can coordinate the information about clients and leases. If `dhcpserve` gives an address to `desktop1`, other servers must learn about that address and its lease before they can extend the lease. The DHCP specification in RFC 2131 does not define a standard way for servers to exchange lease information. Chapter 10, "Failover Protocol Operation." And Chapter 17, "Setting Up a Reliable DHCP Service," discuss some alternatives for using multiple DHCP servers.

In the `REBINDING` state, after a server has received the broadcast `DHCPREQUEST` message from the client, it unicasts the `DHCPACK` message to the client. When the client receives a `DHCPACK` message, it records the new lease and the responding server's address. The next time the client extends its lease, it uses this new server.

When a Lease Expires

If a client is unable to contact a server to renew its lease before the lease expires, the client must stop using its IP address and go back to the `INIT` state. When the client's lease on an address expires while the client is using the network, all active TCP connections are dropped, and the user might have to reconnect to network applications that were in use. Data (and tempers!) can be lost when the client's lease expires, so it is important for the client to extend the lease before it expires. Ways of preventing lease expiration are discussed in Chapter 17.

Moving to a New Network

If `desktop1` moves to a new network segment while it is powered off, DHCP allows it to quickly discover that it has moved. When it starts up after the move, it broadcasts a `DHCPREQUEST` message for its old IP address. When `dhcpserve` receives the `DHCPREQUEST` message, it compares the new network segment to the requested address and determines that the address will not work on that network segment. For example, if `desktop1` moves from the 192.168.11.0 subnet to the 192.168.12.0 subnet, it broadcasts its previous address—say, 192.168.11.25—in a `DHCPREQUEST` message. The relay agent on the 192.168.12.0 subnet forwards the message to the server. The server checks its configuration information and finds that the client's requested address, 192.168.11.25, is not on the same network segment as the 192.168.12.0 subnet, which means that `desktop1`'s address is invalid.

`dhcpserve` notifies `desktop1` that its address is invalid by sending a `DHCPNAK` message. After `desktop1` receives the `DHCPNAK` message, it discards its old address and enters the `INIT` state. At that point, the client tries to obtain a new, valid address as though it just started up with no IP address. So, `desktop1` goes through the process described in the section "Obtaining an Initial Configuration" to locate a server and obtain an address that works on its new subnet. Figure 8.5 illustrates this sequence of messages.

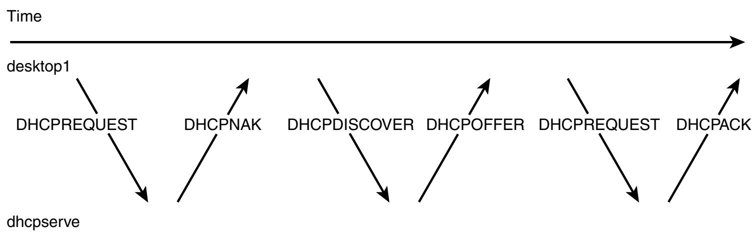

FIGURE 8.5 Messages used by a DHCP client to obtain an address from a new subnet.

NOTE

Most kinds of network hardware provide ways to notice when a move may have occurred. Ethernet has a *carrier*, which is always present when a node is connected to the Ethernet network. If the node notices that the carrier has gone away and then comes back, it can assume that the node might have changed networks. Some DHCP clients, such as the DHCP client in Mac OS X, watch for this kind of transition, and if it happens, they immediately go into the `INIT-REBOOT` state to reconfirm their leases. This is not required by the DHCP specification, but is a useful optimization.

Working with Multiple Servers

Although the examples thus far in this chapter focus on the messages exchanged between a client and a single server, the DHCP specification is, in fact, written to accommodate the use of more than one server. More than one server may be involved in all client/server exchanges—not just as alternate servers for lease extensions, as described in the section "Extending a Lease." The model of operation described in RFC 2131 allows multiple, independent DHCP servers to be configured on an organization's network, and it requires that a DHCP client be prepared to receive multiple responses to its broadcast messages.

Obtaining an Initial Address

When a DHCP client broadcasts a `DHCPDISCOVER` message, it might receive responses from more than one server. Each responding server offers an address and other configuration information for the client. The client listens for these responses after sending the `DHCPDISCOVER` message, and it chooses one address from among those offered. The client uses whatever criteria it wants to select a server. In practice, however, most clients simply choose the first server to respond.

NOTE

It might be the case that one of the responding servers already has an active lease for the client. For example, a DHCP client with no local permanent storage sends an initial `DHCPDISCOVER` every time it restarts because it has no record of its previous address. Similarly, a client that crashed and lost its record of any previous address broadcasts a `DHCPDISCOVER` when it restarts. Unfortunately, a server cannot indicate to a client that the client already has a lease for the address offered by the server. So the client may choose a different server and unnecessarily switch addresses.

What about the servers and the offered addresses that are not selected? The client indicates which server it has selected in its `DHCPREQUEST` message by including the server's IP address in the `server identifier` option. The `DHCPREQUEST` message is then broadcast and delivered to all DHCP servers. The selected server responds to the client, and the other servers return the offered addresses to their pools of available addresses. Figure 8.6 illustrates this sequence of messages.

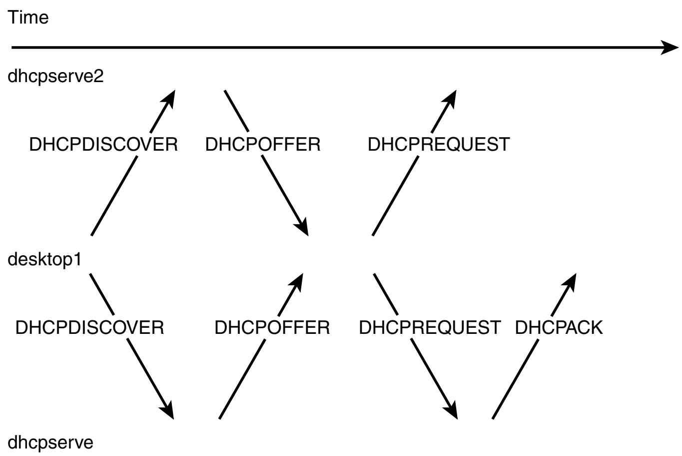

FIGURE 8.6 A time line of messages exchanged between a client and multiple servers when the client restarts.

Of course, the broadcast `DHCPREQUEST` message may not reach all the DHCP servers. How long should a server reserve an offered address, waiting for the message from the client? Reserving the address forever is, of course, a bad idea. If only a relatively small number of addresses are available, the server will eventually run out of addresses.

In fact, the DHCP specification enables the server to decide whether to reserve an address. If, for some reason, it assigns an address to Client B that it previously offered to Client A before Client A requested the address, the server sends a `DHCPNAK` message in response to Client A's request. When the client receives the `DHCPNAK`, it restarts its configuration process.

NOTE

The ISC DHCP server does not attempt to make use of `DHCPREQUEST` messages that are intended for other servers. Instead, it reserves addresses offered to clients for two minutes. The server also returns addresses offered but not subsequently requested to the end of the list of available addresses. As long as enough addresses are available on the network, even a `DHCPREQUEST` message from the client more than two minutes after the address was offered will probably be honored.

Restarting

A client may receive replies from multiple servers in response to its `DHCPREQUEST` message when restarting. In this case, the client's job is easy: It simply records the lease and configuration parameters from the first response and discards subsequent responses.

A potential problem exists with the consistency of the lease information after servers extend a client's lease. Because the client does not respond to the servers' `DHCPACK` messages, the servers don't know which response the client accepted. If the servers offer different leases, they will have different information about when the client's lease expires. This problem can occur when the client restarts and when it broadcasts a request to extend its lease. This problem is discussed as part of the larger issue of exchanging DHCP information among servers in Chapter 10.

Broadcasting to Extend a Lease

A broadcast `DHCPREQUEST` message that a client sends to find an alternate server to extend its lease might trigger replies from multiple servers. The client's response is much the same as if it receives multiple responses to its initial message when restarting: The client simply records the lease and server identifier from the first reply it receives, and it discards the rest.

Other Message Exchanges

Three additional message sequences are possible:

- The sequence used by a client that is configured with an IP address through another mechanism
- The sequence used by a client to terminate its lease before the lease expires
- The sequence used by a server to force a client to renew its lease before time T1

These messages are used less frequently than the messages described earlier in this chapter.

Obtaining Configuration Information with an IP Address Not Obtained Through DHCP

Some network devices are manually configured with an IP address or they obtain their addresses through another protocol, such as Point to Point Protocol (PPP; see RFC 1661).

A network administrator might want to provide other configuration parameters to these devices through DHCP. Using DHCP for configuration parameters such as DNS servers and NTP servers enables the administrator to automatically change the addresses of those servers on all managed computers, minimizing manual configuration and ensuring consistency.

A client with an IP address sends a `DHCPINFORM` message to a server to obtain other configuration parameters. When the server receives the `DHCPINFORM` message, it determines all the appropriate parameters, using the same policies it employs for computers that use dynamic address assignment. The server returns those parameters to the client in a `DHCPACK` message.

NOTE

`DHCPINFORM` doesn't have a mechanism through which a DHCP server can schedule the time at which the client recontacts the server. If the DHCP server administrator changes network configuration parameters but the client does not use `DHCPINFORM` to obtain the new parameters, the client may be using the old, incorrect parameters until the next time it is restarted (which could be quite a long time).

If the `DHCPFORCERENEW` message is available on both the server and the client, the server can use it to cause the client to contact the server immediately rather than wait for the next time the client is restarted.

Terminating a Lease

A client can give up its lease on an address before the lease expires by sending a DHCPRELEASE message. A user might want to send a DHCPRELEASE message from a computer that is about to be moved from one subnet to another, so the server knows that the computer's old address is immediately available for reassignment. Some DHCP clients can be configured to send a DHCPRELEASE each time the computer is shut down.

The client sends its current address in the DHCPRELEASE message and does not wait for a response. The client stops using its old address as soon as it sends the DHCPRELEASE message, so the server cannot send a response to the client. And, as a practical matter, a user won't want to wait for the server to respond while the computer is shutting down.

Updating a Client's Configuration

A DHCP server can use the DHCPFORCERENEW message to update a client's IP address and configuration immediately, without waiting for the client to contact the server with a DHCPREQUEST or DHCPREBIND message. The server sends the DHCPFORCERENEW message directly to the DHCP client, as described in the section "DHCPFORCERENEW Messages" in Chapter 7. After the client receives the message, it goes to the RENEWING state (refer to Figure 8.1). The client then contacts the server by sending a DHCPREQUEST message. If the server only needs to update the client's parameters, and not its IP address, it can simply respond with a DHCPACK message that contains the new parameters. Otherwise, it responds with a DHCPNAK message, forcing the client into the INIT state. At this point, the client continues by broadcasting a DHCPDISCOVER message. Finally, the server responds, as described in the section "Obtaining an Initial Configuration," with the client's new IP address and configuration information.

Summary

Several interactions between DHCP clients and servers involve specific sequences of messages. These interactions are designed to allow the clients to operate without prior knowledge of the location of DHCP servers and to accommodate responses from more than one server. The interactions also minimize the effect of network and server failures on the client. The goal is to provide a robust protocol that enables clients to function in the event of network problems.

When a DHCP client does not have an IP address, it uses a two-phase process to locate a DHCP server and obtain an initial address. After the client has obtained an address, it confirms that its address is still valid whenever the client restarts. A client can contact the server and extend the lease on its address while the client is still running. A client that has already established a lease with a server and merely wants to confirm the lease when rebooting can do so by using a one-phase process instead of the two-phase process used initially.

Three other message sequences are less central to the usual client/server interaction. A client can send a `DHCPRELEASE` message to terminate its lease on an address before the lease expires. If the client already has an IP address, it can use `DHCPINFORM` to obtain other configuration information such as server addresses. A server can use a `DHCPFORCERENEW` message to cause a client to contact the server before the time at which the client was scheduled to extend its lease.

Both this chapter and Chapter 7 describe the mechanisms through which clients and servers communicate in DHCP. Chapter 9, "DHCP Options," looks at the specific information that clients and servers exchange in the `options` section of DHCP messages.

9

DHCP Options

IN THIS CHAPTER

- DHCP-Specific Options
- Host Configuration Parameters Options
- TCP/IP Stack Configuration Parameters
- Service Parameter Options
- `subnet selection`
- `authentication`
- `relay agent information`

Chapter 6, "The Format of DHCP Messages," introduces *options* as the mechanism through which a DHCP message type is identified and configuration parameters are transmitted between DHCP servers and clients. This chapter describes the DHCP options, which are defined in RFC 2132. The next few chapters explain how options are used in DHCP messages.

The options described in this chapter are organized into several categories: options that are specific to DHCP, options that provide configuration parameters for the DHCP client, options that carry TCP/IP stack parameters, and application and service parameter options.

As illustrated in Chapter 8, "DHCP Message Exchanges," all DHCP options except for `END` and `PAD` share the same three-part format: a 1-byte option code, a 1-byte `length` field, and the data that is carried by the option. Options carry data in different formats, including IP addresses, character strings, and integers. The descriptions in this chapter give the option code, the acceptable `length` field values, and the format of the data in each option.

NOTE

The DHCP options listing servers and other network devices identify servers by their IP addresses rather than by domain names. Thus, if one of those servers is relocated or for some other reason assigned a new IP address, the DHCP server databases must be updated to reflect that new address. Also, DHCP clients do not have a valid address for a server that has been assigned a new IP address until the client contacts the DHCP server when it restarts or extends its lease.

NOTE

The DHCP specification calls for the use of 7-bit USASCII characters as used by the NVT (network virtual terminal) and defined in the Telnet protocol (which is described in RFC 854). The DHCP RFCs refer to NVT ASCII characters and NVT ASCII strings. This chapter simply uses characters and strings.

DHCP-Specific Options

Although both DHCP and BOOTP clients and servers can use the options described in other sections of this chapter, the options in this section are specific to DHCP operation.

NOTE

The description of each option in this chapter includes a table that lists the option code, the range of values of the `length` field, and the interpretation of data values, along with a short text description of the option. Most of these options are defined in RFC 2132; specific references are included for options that are defined elsewhere.

`DHCP message type`

Option code:	53	
Length:	1	
Data:	DHCPDISCOVER	1
	DHCPOFFER	2
	DHCPREQUEST	3
	DHCPDECLINE	4
	DHCPACK	5
	DHCPNAK	6
	DHCPRELEASE	7
	DHCPINFORM	8

DHCPFORCERENEW	9
DHCPLEASEQUERY	10
DHCPKNOWN	11
DHCPUNKNOWN	12

NOTE

At the time this book was written, the `DHCPLEASEQUERY`, `DHCPKNOWN`, and `DHCPUNKNOWN` message types had not yet been assigned code values.

DHCP client/server transactions use several different message types. The `DHCP message type` option identifies a specific type of DHCP message. Chapter 7, "Transmitting DHCP Messages," explains how the different message types are used.

`client identifier`

Option code:	61
Length:	*n*
Data:	identifier (*n* bytes)

DHCP servers use the value of the `client identifier` option to distinguish between DHCP clients. If the `client identifier` option is present, the DHCP server uses it; otherwise, the server uses the contents of the `htype` and `chaddr` fields of the DHCP message. A DHCP client's identifier must be unique among all the client identifiers on the IP network to which the client is attached. The server treats a DHCP client identifier as an opaque value and does not interpret it in any way.

RFC 2132 suggests that client identifiers be composed of a 1-byte type field, followed by the identifier itself, similarly to the combination of `htype` and `chaddr` fields in the `fixed-format` section of a DHCP message. If this format is used, the type value in the client identifier is either selected from the ARP hardware types defined in STD2 or is set to `0`. In the latter case, the client identifier is an arbitrary string, such as a domain name, rather than a link-layer address.

NOTE

Although RFC 2132 suggests the typed format for client identifiers, this is not required, and some DHCP clients simply send a text string with no type identifier.

`server identifier`

Option code:	`54`
Length:	`4`
Data:	IP address

The `server identifier` option gives the IP address of the server involved in the DHCP transaction. This option is used by the server to send the server's IP address to the client, and it is used by the client to identify the server to which the client intends to deliver a `DHCPREQUEST` message. If a server has more than one network interface, it uses the IP address of the interface on which it received the DHCP message to which it is responding.

The `server identifier` option differs from the `siaddr` field in the `DHCP message header` section. A client uses the server identifier to determine the source of a DHCP message delivered to the client and to indicate for which server a broadcast DHCP message is intended. The `siaddr` field holds the IP address of the server that the client should contact to obtain additional bootstrap services, such as additional configuration information or an operating system kernel, through a network protocol such as TFTP.

`requested address`

Option code:	`50`
Length:	`4`
Data:	IP address

The `requested IP address` option contains the IP address that the client requests when it does not have explicit confirmation that its current address is valid. A client includes its previous IP address in a `requested IP address` option when sending a `DHCPREQUEST` message during restart.

`lease time`

Option code:	51
Length:	4
Data:	lease time

The value of the `lease time` field indicates the duration of the lease for an address assigned to a client. The `lease time` value is an unsigned 32-bit number that represents the length of the lease, in seconds. The reserved value $FFFFFFFF_{16}$ indicates a lease that never expires (in other words, a lease that is of infinite duration).

When a server sends a message to a client, the `lease time` option represents the length of the lease the server selected according to the network's policies. When a client sends a message to a server, the `lease time` option represents the length of the lease the client is requesting from the server. The lease time that the server supplies is authoritative, and the client must honor it, regardless of the lease time it requests.

`lease renewal time` (T1)

Option code:	58
Length:	4
Data:	T1

T1 represents the point in time when a client begins extending the lease on its address. T1 and T2 are always specified as a number of seconds relative to the current time. Beginning at time T1, the client unicasts `DHCPREQUEST` messages to the server from which the lease on the address was obtained. This option specifies the value in `T1` to the client as a 32-bit unsigned integer representing T1 in seconds. If the server does not use this option to specify T1 to the client, the client uses half the initial lease duration for T1.

`lease renewal time` (T2)

Option code:	59
Length:	4
Data:	T2

T2 is the time when a client begins finding a new server through which it can extend the lease on its address. Beginning at time T2, the client broadcasts `DHCPREQUEST` messages to locate a server that is willing to extend its lease. This option specifies `T2` as a 32-bit unsigned integer representing T2 in seconds. If the server does not use this option to specify T2 to the client, the client uses seven-eighths of the initial lease duration for T2.

Some clients ignore the values of T1 and T2 that are specified by the server in these options. Instead, these clients compute T1 to be one-half of the duration of the lease and T2 to be seven-eighths of the duration of the lease. If you find that a client is not attempting to extend its lease at the time specified in the `T1` option, the client might be one that ignores values of T1 and T2 that are sent from the server.

`vendor class identifier`

Option code:	`60`
Length:	*n*
Data:	vendor class identifier

A DHCP client uses the `vendor class identifier` option to pass information about the client's vendor type and configuration. The server uses this option to interpret the contents of the `vendor-specific options` field, and (optionally) to select specific configuration parameters for a client. The identifier is a string of opaque byte values that is not terminated with a null character.

`vendor-specific information`

Option code:	`43`
Length:	*n*
Data:	vendor-specific information

The `vendor-specific information` option carries information that is interpreted according to the client vendor type, as specified in the `vendor class identifier` option. This option enables a vendor to define new options used only by its clients, without going through the standards process or consuming limited option code space.

If more than one option is carried in the `vendor-specific information` option, then the options are encoded in the same way as DHCP options and encapsulated in the `vendor-specific information` option data area. For example, if a vendor defines two options with option codes 125 and 126, a `vendor-specific information` option that carries those vendor codes is encoded as follows:

```
43   9   125   4   192   168   7   4   126   1   1
```

In this example, the option code is 43, specifying the vendor-specific information option. The length is 9, giving the length of all the data for the option. The first encapsulated option is the vendor's option 125, with a length of 4 and data 192.168.7.4. The second encapsulated option is 126, with a length of 1 and data 1.

parameter request list

Option code:	55
Length:	*n*
Data:	*n* option codes

A DHCP client uses the parameter request list option to request specific parameter values from a server. Each byte in the parameter request list is a DHCP option code that the client wants the server to provide. The server includes in its response to the client values for the requested option, along with other options that are required by DHCP.

NOTE

The exact behavior of the parameter request list option is not defined. Some DHCP servers return just the options that are listed in the parameter request list only if there is a parameter request list option. Some DHCP servers return options that are not listed in the parameter request list option anyway. The safest things a DHCP client can do are either to specify in the parameter request list every option that it wants to receive or to not send a parameter request list option at all.

message

Option code:	56
Length:	*n*
Data:	*n* characters

DHCP servers and clients use the message option to transmit an error message to a DHCP message recipient. The format of the contents of the message option is unspecified and is typically a character-string message that is displayed to a user or recorded in a log file.

maximum DHCP message size

Option code:	57
Length:	2
Data:	length

A client or server uses the `maximum DHCP message size` option to advertise that it will accept incoming messages that are larger than the default maximum size for DHCP message (576 bytes). The length is stored as an unsigned 16-bit integer and must not be less than 576.

`option overload`

Option code:	52	
Length:	1	
Data:	1	`file` field holds options
	2	`sname` field holds options
	3	both fields hold options

If the `option overload` option is present in a DHCP message, the message recipient concatenates the specified fields with the `options` field and interprets the options in the resulting list. See the section "Examples of Message Formats" in Chapter 6 for a more detailed explanation.

`TFTP server name`

Option code:	66
Length:	*n*
Data:	*n* characters

The `TFTP server name` option identifies a TFTP server for the client to use in the next phase of its bootstrap process, when the `sname` field in the DHCP header has been used for DHCP options. The name is a string that is not terminated with a null character.

`bootfile name`

Option code:	67
Length:	*n*
Data:	*n* characters

The `bootfile name` option identifies a bootfile name for the client to use when the `file` field in the DHCP header is used for DHCP options. The name is a string that is not terminated with a null character.

`pad`

Option code: `0`

The `pad` option carries no information and is skipped when the `options` field is interpreted. It can be used, for example, to pad the `options` section to the BOOTP standard of 64 bytes.

`end`

Option code: `255`

The `end` option indicates the end of the options carried in the `options` field. Because the end of the `options` field in the DHCP message can be inferred from the length of the UDP message in which the DHCP message is transmitted, some DHCP clients and servers do not send the `end` option.

> **NOTE**
>
> Only the `pad` and `end` options do not include `length` or `data` fields.

Host Configuration Parameters Options

These options provide configuration parameters that apply to the host.

`host name`

Option code:	`12`
Length:	*n*
Data:	*n* characters

The `host name` option gives the client's name. This name can be only the client's name or the client's name, qualified with the local domain name. The `host name` option is a string that is not terminated with a null character.

A client uses the `host name` option to inform the DHCP server of the name the client is using. A server uses the `host name` option to inform the client of the name it should use for itself. RFCs 2131 and 2132 are unclear as to how clients and servers should react if they disagree about the name the client should use.

`domain name`

Option code:	`15`
Length:	*n*
Data:	*n* characters

The `domain name` option specifies the name of the client's domain for resolving names in the Domain Name System (DNS). The name is a string of characters that is not terminated with a null character.

NOTE

The `domain name` option specifies only a single domain for name resolution. The `domain search` option, which is about to be accepted as a standard DHCP option, specifies a list of domain names used in name resolution.

`client FQDN`

Option code:	`81`
Length:	*n*
Data:	`flag` (1 byte)
	`rcode1` (1 byte)
	`rcode2` (1 byte)
	`domain name` (n-3)

The `client FQDN` option is used by both DHCP clients and DHCP servers, in order to negotiate whether and how the client and server will update the domain name server with the client's address information. A DHCP server only sends an FQDN option to a DHCP client in response to a message from the client that contains an FQDN option.

DHCP clients use the `client FQDN` option to tell DHCP servers what hostname or fully qualified domain name (FQDN) the client wants to use. When a DHCP server receives a `client FQDN` option from a client, it can respond to the client with a `client FQDN` option that either confirms the client's choice or provides the client with the name the server wants the client to use.

The first byte of the `client FQDN` option is a `flag` byte. Four bits in the `flag` byte have been assigned codes:

Bit	Code
1	`S`
2	`O`
4	`E`
8	`N`

If the client wants the DHCP server to update its A record, the client sets the `S` bit in the `flag` byte. If the client intends to update its own A record, it clears the `S` bit. If the client does not want the DHCP server to do any DNS update at all for the client, the client sets the `N` bit; otherwise, it leaves the `N` bit clear. DHCP clients never set the `O` bit.

If the DHCP server receives a `client FQDN` option from the client and it supports the `client FQDN` option, it always sends a `client FQDN` option back to the client. If the server is configured to update the client's A record, it sets both the `O` and the `S` bits to inform the client of its policy. DHCP servers never set the `N` bit.

DHCP clients and servers can both set the `E` bit. The DHCP server always sets the `E` bit the same way that it is set in the client's message. All new DHCP clients must set the `E` bit, but older DHCP clients might not set it. If it is set, the `E` bit indicates that the domain name is encoded in DNS wire format. If it is clear, the `E` bit indicates that the domain name is encoded as in the DNS presentation format (for example, the ASCII string `foo.example.com` is in presentation format).

The `rcode1` and `rcode2` bytes are present for backward compatibility with old clients, but they are not used.

When the client sends an FQDN option, it puts the unqualified hostname or fully qualified domain name that it wants to use in the domain name field.

When the server responds to the client, it either sends the client the same name that the client sent or, if it wants to provide the client with a different name, it sends that name instead.

Chapter 23, "Configuring DHCP–DDNS Interactions," describes how DHCP clients and servers do DNS updates and how the `client FQDN` option is used. The `client FQDN` option has only recently begun to be implemented, and many clients do not support it. Clients that do support it usually also send a `hostname` option. No standard says which option the server should use if both are present, although common sense suggests that the `client FQDN` option should be used.

`time offset`

Option code:	2
Length:	4
Data:	32-bit signed integer

The `time offset` option specifies which time zone offset from *coordinated universal time* the client should use. (Coordinated universal time is often abbreviated UTC, which is the abbreviation for the French name, Universal Temp Coordiné.) The offset is a signed 32-bit integer that expresses the offset in seconds. A positive offset indicates a location east of the zero meridian, and a negative offset indicates a location west of the zero meridian.

`bootfile size`

Option code:	13
Length:	2
Data:	16-bit unsigned integer

The `bootfile size` option gives the size of the client's bootfile. The size is an unsigned 16-bit integer that specifies the number of 512-byte blocks in the bootfile.

`root path`

Option code:	17
Length:	*n*
Data:	*n* characters

The `root path` option gives the filename (that is, the full pathname) of the directory that is being used as the client's root disk partition. Diskless clients use it to mount a network disk from the server that is identified in the `siaddr` or `sname` fields.

`swap server`

Option code:	16
Length:	4
Data:	IP address

The `swap server` option gives the IP address of a server that provides a swap space service (that is, a service such as providing swap storage through the network for a diskless workstation) for the client.

`extensions path`

Option code:	18
Length:	*n*
Data:	*n* characters

The `extensions path` option gives the filename (that is, the full pathname) of a file that contains additional options for the client. The name is a string that is not terminated with a null character. The client obtains a copy of the file identified in the `extensions path` option by using TFTP from the server that is identified in the `siaddr` or `sname` fields. After retrieving the file, the client interprets the contents of the file by using the same syntax used in the contents of the `options` field.

`merit dump file`

Option code:	14
Length:	*n*
Data:	*n* characters

The `merit dump file` option gives the name of a file where the client should dump a core image if the client crashes. The name is a string that is not terminated with a null character.

NOTE

DHCP options that specify filenames do not specify a particular format for the filenames. The format of the filenames is dictated by the needs of the client and of the servers that the client will be contacting.

`user class`

Option code:	77
Length:	*n*
Data:	*n* characters

The `user class` option (as described in RFC 3004) gives textual information that a DHCP client can use to identify the type or category of user or applications it represents. The data in the `user class` option is structured as a sequence of separate user class values. Each value consists of a single byte that gives the length of the value, followed by the actual data values. A DHCP server uses the `user class` option values to select the configuration information to be assigned to the client.

TCP/IP Stack Configuration Parameters

The following sections describe several types of TCP/IP stack configuration parameters. Most of the parameters apply to the IP layer—either to all IP traffic on the client or to the traffic on a specific interface. Other parameters configure the link layer and the TCP layer.

IP Layer Parameters for the Client

Each of the options in the following sections supplies a configuration parameter that applies to all IP traffic on the client.

`router`

Option code:	3
Length:	*n*
Data:	list of IP addresses

The `router` option lists the addresses of default routers for the client to add to its routing table. The routers are listed by their IP addresses, in order of preference. The `length` field gives the total length of the list of routers and must be a multiple of 4; if the list contains *r* routers, the length is 4*r*.

`default IP time to live`

Option code:	23
Length:	1
Data:	1 byte

The `default IP time to live` option gives the default value for the `TTL` (Time to Live) field in the IP header of datagrams that the client transmits.

`IP forwarding enable/disable`

Option code:	19	
Length:	1	
Data:	0	disable forwarding
	1	enable forwarding

The `IP forwarding enable/disable` option controls whether a client that has more than one interface should forward IP datagrams between its interfaces.

`maximum datagram reassembly size`

Option code:	22
Length:	2
Data:	16-bit integer

The `maximum datagram reassembly size` option specifies the largest fragmented IP datagram the client is prepared to reassemble.

`Nonlocal Source Routing Options`

Two options —`nonlocal source route enable/disable` and `policy filter`— control the forwarding of IP datagrams with nonlocal source routes (see Section 3.3.5 of RFC1122).

`nonlocal source route enable/disable`

Option code:	20	
Length:	1	
Data:	0	disable forwarding of datagrams with nonlocal source routes
	1	enable forwarding of datagrams with nonlocal source routes

The `nonlocal source route enable/disable` option controls whether the client forwards datagrams with nonlocal source routes.

`policy filter`

Option code:	21
Length:	*n*
Data:	list of filters

The `policy filter` option controls the filters applied to datagrams with nonlocal source routes. Each filter includes an IP address and a subnet mask. The client discards an IP datagram with a source route whose next hop does not match one of the addresses (with its associated subnet mask applied) given in the `policy filter` option. The `length` field for the `policy filter` option must be a multiple of 8; if *f* filters exist, the length is `8f`.

Path Maximum Transmission Unit (PMTU) Options

The options described in the following sections control the PMTU mechanism in the client.

`PMTU aging timeout`

Option code: 24

Length: 4

Data: 32-bit integer

The `PMTU aging timeout` option specifies the timeout value to be used when PMTU values are being aged. The option gives the timeout value in seconds.

`PMTU plateau table`

Option code: 25

Length: *n*

Data: list of 16-bit integers

The `PMTU plateau table` option specifies a table of MTU sizes for use in the PMTU mechanism. Each MTU size is an unsigned 16-bit integer. The MTU sizes are listed in order, from smallest to largest. The minimum MTU value must be at least 68. The length of the PMTU plateau table list must be a multiple of 2; if *p* plateau values exist, the length is $2p$.

Options Defining IP Layer Parameters for a Specific Interface

The options in the following sections give configuration parameters that apply to the IP traffic through a specific interface. Because a DHCP client with multiple network interfaces must use DHCP separately for each interface, it can receive customized configuration parameters for each interface.

`subnet mask`

Option code: 1

Length: 4

Data: subnet mask

The `subnet mask` option carries the subnet mask for the interface to which the DHCP message is delivered. The subnet mask is represented as a 32-bit integer in network byte order, with bits set to 1 corresponding to each bit in the address to be used in a network or subnet number.

`broadcast address`

Option code: 28

Length: 4

Data: IP address

The `broadcast address` option specifies the IP broadcast address for the network to which the configured interface is attached. The address is represented as a 32-bit IP address. Examples of broadcast addresses are all 1s (255.255.255.255), 0, or a network broadcast address such as 192.1.1.255 or 192.1.1.0.

`MTU for the interface`

Option code:	26	
Length:	2	
Data:	MTU in bytes for this interface	

The `MTU for the interface` option gives the MTU to be used for this interface, overriding any MTU deduced from the network interface hardware.

`all subnets local`

Option code:	27	
Length:	1	
Data:	1	all subnets have same MTU
	0	some subnets may have smaller MTU

If the `all subnets local` option has value `1`, the client can assume that all network segments in the network to which the client is attached have the same MTU. If this option has value `0`, some network segments may have a smaller MTU.

`perform mask discovery`

Option code:	29	
Length:	1	
Data:	1	perform mask discovery
	0	don't perform mask discovery

If the `perform mask discovery` option has the value `1`, the client should use the ICMP `address mask request` message (see RFC 950) to obtain the subnet mask for this interface.

`supply subnet mask`

Option code:	30	
Length:	1	
Data:	1	supply subnet mask
	0	don't supply subnet mask

If the `supply subnet mask` option has the value `1`, the client should respond to ICMP `address mask request` messages.

`static routes`

Option code:	33
Length:	*n*
Data:	list of IP address pairs

The `static routes` option lists static routes for the client to install in its routing table. Each static route is listed as the address of the destination and the address of the router to use for that destination. If duplicate static routes exist for a destination, the routes are listed in decreasing order of priority. The length of this option must be a multiple of 8; if *s* static routes exist, the length is `8s`.

> **NOTE**
>
> Since the definition of *classless interdomain routing* (CIDR), the `static routes` option is no longer useful. The `classless static routes` option carries routes with subnet masks and should be used in place of the `static routes` option.

Router Discovery Options

Two options—`perform router discovery` and `router solicitation address`—control the use of router discovery (see RFC 1256).

`perform router discovery`

Option code:	31	
Length:	1	
Data:	0	do not perform router discovery
	1	perform router discovery

The `perform router discovery` option specifies whether the client should initiate the router discovery protocol described in RFC 1256.

`router solicitation address`

Option code:	32
Length:	4
Data:	IP address

The `router solicitation address` option gives the address the client should use to transmit router discovery protocol messages.

Link Layer Parameters Options

A few options affect the configuration of link-layer parameters. These options apply only to the link-layer software for the interface over which the DHCP message is received.

`ARP cache timeout`

Option code:	35
Length:	4
Data:	32-bit integer

The `ARP cache timeout` option specifies the lifetime for entries in the ARP cache. The timeout value is represented in seconds.

`Ethernet encapsulation`

Option code:	36	
Length:	1	
Data:	0	Ethernet version 2 encapsulation
	1	IEEE 802.3 encapsulation

The `Ethernet encapsulation` option selects either Ethernet version 2 (defined in RFC 894) or IEEE 802.3 (defined in RFC 1042) encapsulation for IP datagrams.

`trailer encapsulation`

Option code:	34	
Length:	1	
Data:	0	do not use trailers
	1	attempt to use trailers

The `trailer encapsulation` option controls whether the client should negotiate the use of trailers (see RFC 893) by using ARP.

TCP Parameters Options

The options described in the following sections give values for TCP software configuration parameters. These options affect the parameters for all TCP traffic on the client.

`TCP default TTL`

Option code:	37
Length:	1
Data:	8-bit integer

The `TCP default TTL` option sets the TTL that the TCP layer should use when sending TCP segments.

TCP Keepalive Parameters Options

Two options—`TCP keepalive interval` and `TCP keepalive garbage`—control the operation of the TCP keepalive mechanism.

`TCP keepalive interval`

Option code:

38

Length:

4

Data:

0

do not use keepalive

nonzero integer

use keepalive interval

The `TCP keepalive interval` option specifies the time the client should wait before a keepalive segment. The interval is given in seconds. If the data value is zero, the client should not use keepalive segments unless an application specifically requests it.

`TCP keepalive garbage`

Option code:	39	
Length:	1	
Data:	0	do not send garbage byte
	1	send garbage byte

The `TCP keepalive garbage` option specifies whether the client should send a garbage byte in keepalive segments (see section 4.2.3.6 of RFC 1122).

Service Parameter Options

The options described in the following sections provide parameters for a variety of services. Most of these options provide the address of a server or servers for a specific service. Some options give additional information about a service, such as the Network Information Service (NIS) or Network Information Service+ (NIS+).

`time server`

Option code:	4
Length:	*n*
Data:	list of IP addresses

The `time server` option lists addresses of RFC 868 time servers that are available to the client. The length must be a multiple of 4; if *s* time servers are in the option, the length is `4s`.

`name server`

Option code:	5
Length:	*n*
Data:	list of IP addresses

The `name server` option lists addresses of Internet Name Servers (see IEN-116, `www.isi.edu/in-notes/ien/ien116.txt`) that are available to the client. The length must be a multiple of 4; if *s name* servers are in the option, the length is `4s`.

`domain server`

Option code:	6
Length:	*n*
Data:	list of IP addresses

The `domain server` option lists addresses of Domain Name Service (DNS) servers (see RFC 1035) that are available to the client. The length must be a multiple of 4; if *s* DNS servers are in the option, the length is `4s`.

`log server`

Option code:	7
Length:	*n*
Data:	list of IP addresses

The `Log server` option lists addresses of logging servers that are available to the client. The length must be a multiple of 4; if *s* logging servers are in the option, the length is `4s`.

`quotes server`

Option code: `8`

Length: *n*

Data: list of IP addresses

The `quotes server` option lists addresses of quote of the day servers (see RFC 865) that are available to the client. The length must be a multiple of 4; if *s* quote servers are in the option, the length is `4s`.

`LPR server`

Option code: `9`

Length: *n*

Data: list of IP addresses

The `LPR server` option lists addresses of Line Printer Protocol (LPR) servers (see RFC 1179) that are available to the client. The length must be a multiple of 4; if *s* LPR servers are in the option, the length is `4s`.

`Impress server`

Option code: `10`

Length: *n*

Data: list of IP addresses

The `Impress server` option lists addresses of Impress print servers that are available to the client. The length must be a multiple of 4; if *s* impress servers are in the option, the length is `4s`.

`RLP server`

Option code: `11`

Length: *n*

Data: list of IP addresses

The `RLP server` option lists addresses of Resource Location Protocol (RLP) servers (see RFC 887) that are available to the client. The length must be a multiple of 4; if *s* RLP servers are in the option, the length is `4s`.

`SMTP server`

Option code: `69`

Length: *n*

Data: list of IP addresses

The `SMTP server` option lists addresses of Simple Mail Transport Protocol (SMTP; see RFC 821) servers that are available to the client. The length must be a multiple of 4; if *s* SMTP servers are in the option, the length is `4s`.

`POP server`

Option code: `70`

Length: *n*

Data: list of IP addresses

The `POP server` option lists addresses of Post Office Protocol version 3 (POP3; see RFC 1939) servers that are available to the client. If more than one address is in the list, the POP3 servers are listed in order of preference. The length must be a multiple of 4; if *s* POP3 servers are in the option, the length is `4s`.

`NTP server`

Option code: `42`

Length: *n*

Data: list of IP addresses

The `NTP server` option lists addresses of Network Time Protocol (NTP; see RFC 1305) servers that are available to the client. If more than one address is in the list, the NTP servers are listed in order of preference. The length must be a multiple of 4; if *s* NTP servers are in the option, the length is `4s`.

`finger server`

Option code: `73`

Length: *n*

Data: list of IP addresses

The `finger server` option lists addresses of finger protocol (see RFC 1288) servers that are available to the client. If more than one address is in the list, the finger servers are listed in order of preference. The length must be a multiple of 4; if *s* finger servers are in the option, the length is `4s`.

`WWW server`

Option code:	72
Length:	*n*
Data:	list of IP addresses

The `WWW server` option lists addresses of World Wide Web (HTTP; see RFC 1945) servers that are available to the client. If more than one address is in the list, the WWW servers are listed in order of preference. The length must be a multiple of 4; if *s* WWW servers are in the option, the length is `4s`.

`NNTP server`

Option code:	71
Length:	*n*
Data:	list of IP addresses

The `NNTP server` option lists addresses of Network News Transport Protocol (NNTP; see RFC 977), or NetNews, servers that are available to the client. If more than one address is in the list, the NNTP servers are listed in order of preference. The length must be a multiple of 4; if *s* NNTP servers are in the option, the length is `4s`.

`IRC server`

Option code:	74
Length:	*n*
Data:	list of IP addresses

The `IRC server` option lists addresses of Internet Relay Chat (IRC; see RFC 1459) servers that are available to the client. If more than one address is in the list, the IRC servers are listed in order of preference. The length must be a multiple of 4; if *s* IRC servers are in the option, the length is `4s`.

X Window System Options

Two options—`X Window System Font Server` and `X Window System Display Manager`—provide information to the client about X Window System resources.

`X Window System Font Server`

Option code:	48
Length:	*n*
Data:	list of IP addresses

The `X Window System Font Server` option lists addresses of font servers that are available to the client. If more than one address is in the list, the font servers are listed in order of preference. The length must be a multiple of 4; if *s* font servers are in the list, the length is `4s`.

`X Window System Display Manager`

Option code:	49
Length:	*n*
Data:	list of IP addresses

The `X Window System Display Manager` option lists systems that are running the X display manager that are available to the client. If more than one address is in the list, the display manager systems are listed in order of preference. The length must be a multiple of 4; if *s* display manager systems are in the list, the length is `4s`.

`mobile IP home agent`

Option code:	68	
Length:	0	no agents are available
	n	length of list
Data:	list of IP addresses	

The `mobile IP home agent` option is used with the mobile IP protocol (see RFC 2002). The option lists addresses of home agents available to the client. If the length is `0`, no home agents are available. Otherwise, the length must be a multiple of 4; if *s* home agents exist, the length is `4s`.

NIS and NIS+ Options

The four options described next pass information to the client about the local deployment of NIS, NIS+.

`NIS server addresses`

Option code: `41`

Length: *n*

Data: list of IP addresses

This option contains a list of IP addresses of NIS servers. If more than one address is in the list, the servers are listed in order of preference. The length must be a multiple of 4; if *s* servers are in the list, the length is 4*s*.

`NIS domain`

Option code: `40`

Length: *n*

Data: NIS domain

This option gives the client its NIS domain. The name in each option is a string that is not terminated with a null character.

`NIS+ server addresses`

Option code: `65`

Length: *n*

Data: list of IP addresses

This option contains addresses of the NIS+ version 2 servers. If more than one address is in the list, the servers are listed in order of preference. The length must be a multiple of 4; if *s* servers are in the list, the length is 4*s*.

`NIS+ domain`

Option code: `64`

Length: *n*

Data: NIS+ domain

This option gives the client its NIS+ domain. The name in each option is a string that is not terminated with a null character.

NetBIOS over TCP/IP Options

The options described in the following sections inform the client about the use of NetBIOS over TCP/IP, as described in RFC 1001 and RFC 1002.

`NetBIOS address over TCP/IP name servers (WINS)`

Option code:	44
Length:	*n*
Data:	list of IP addresses

The `NetBIOS address over TCP/IP name servers` option lists the addresses of Windows Internet Naming Service (WINS) servers that are available to the client. If more than one address is in the list, the servers are listed in order of preference. The length must be a multiple of 4; if *s* servers are in the list, the length is 4*s*.

`NetBIOS address over TCP/IP datagram distribution server`

Option code:	45
Length:	*n*
Data:	list of IP addresses

The NetBIOS address over TCP/IP datagram distribution (NBDD) server option lists the addresses of NBDD servers that are available to the client. If more than one address is in the list, the servers are listed in order of preference. The length must be a multiple of 4; if *s* servers are in the list, the length is 4*s*.

`NetBIOS address over TCP/IP node type`

Option code:	46	
Length:	1	
Data:	1	B-node: Broadcast—no WINS
	2	P-node: Peer—WINS only
	4	M-node: Mixed—broadcast, then WINS
	8	H-node: Hybrid—WINS, then broadcast

The `NetBIOS address over TCP/IP node type` option specifies the type of node to which the client should configure itself. The data in the option encodes the node type.

`NetBIOS address over TCP/IP scope`

Option code:	47
Length:	*n*
Data:	NetBIOS over TCP/IP scope

The `NetBIOS address over TCP/IP scope` option specifies the NetBIOS over TCP/IP scope for the client. The scope is encoded as a string of characters, which must be selected according to the restrictions in RFC 1002, RFC 1002, and RFC 1035. The resulting character string is not terminated with a null character.

StreetTalk Options

Two options—`StreetTalk server` and `StreetTalk Directory Assistance server`—give information on StreetTalk servers that are available to the client.

`StreetTalk server`

Option code:	75
Length:	*n*
Data:	list of IP addresses

The `StreetTalk server` option lists the addresses of StreetTalk servers for the client. If more than one address is in the list, the servers are listed in order of preference. The length must be a multiple of 4; if *s* servers are in the list, the length is `4s`.

`StreetTalk Directory Assistance server`

Option code:	76
Length:	*n*
Data:	list of IP addresses

The `StreetTalk Directory Assistance server` option lists the addresses of `server` StreetTalk Directory Assistance (STDA) servers that are available to the client. If more than one address is in the list, the servers are listed in order of preference. The length must be a multiple of 4; if *s* servers are in the list, the length is `4s`.

NDS Options

The three options described in the following sections inform the client about the use of *NetWare Directory Services* (*NDS*). These options are defined in RFC 2241.

`NDS servers`

Option code:	85
Length:	*n*
Data:	list of IP addresses

The `NDS servers` option lists the addresses of NDS servers that are available to the client. If more than one address is in the list, the servers are listed in order of preference. The length must be a multiple of 4; if *s* servers are in the list, the length is `4s`.

`NDS Tree Name`

Option code: `86`

Length: *n*

Data: UTF-8-byte string

The `NDS Tree Name` option provides the name of the NDS tree that is available to the client. The names of NDS trees are normally specified as 16-bit Unicode strings. When carried in an `NDS Tree Name` option, an NDS tree name is encoded using *UTF-8* (*UCS Transformation Format-8*). The resulting byte string is not terminated with a null character.

NOTE

ISO/IEC 10646-1 defines a multi-octet character set called the *Universal Character Set* (*UCS*), which incorporates the characters from most of the writing systems that exist today. UCS also defines an encoding that represents each of those characters as a 16-bit number. A string composed of these 16-bit UCS values is called a *UCS string*.

The UCS standard also defines UTF-8 encoding, which represents 7-bit ASCII values in a single byte and then uses multibyte values to represent other characters from the Unicode Standard. UTF-8 is useful because text that uses simple 7-bit values is left unchanged when converted to UTF-8, providing backward compatibility.

`NDS context`

Option code: `87`

Length: *n*

Data: UTF-8-byte string

The `NDS context` option gives the initial NDS context for the client. NDS contexts are normally specified as 16-bit Unicode strings. When carried in an `NDS context` option, an NDS context is encoded by using UTF-8. The resulting byte string is not terminated with a null character.

NOTE

Because of restrictions in the DHCP option format in which the `length` field is encoded as a single byte, a DHCP option can carry only 255 data bytes. However, an NDS context name can be longer than 255 bytes. To accommodate longer NDS context names, the sender splits the context name among multiple occurrences of the `NDS context` option. The receiver then concatenates the `data` fields of the `NDS context` options to reconstruct the complete NDS context name.

NetWare/IP Options

Two options—`NetWare/IP Domain Name` and `NetWare/IP Domain Name`—provide information about the client's configuration for NetWare/IP. These options are defined in RFC 2242.

`NetWare/IP Domain Name`

Option code:	62
Length:	*n*
Data:	*n* characters

The `NetWare/IP Domain Name` option provides the name of the NetWare/IP domain for the client. The name is a string of characters that is not terminated with a null character.

`NetWare/IP Information`

Option code:	63
Length:	*n*
Data:	NetWare/IP information

The `NetWare/IP Information` option carries additional NetWare/IP information for the client.

NetWare/IP Suboptions

The data area of the `NetWare/IP Information` option is composed of one or more suboptions. The suboptions are encoded in the same way as DHCP options; each suboption includes a suboption code, a length, and the data.

The first four suboptions define how the NetWare/IP information is carried. The `NetWare/IP Information` option must include one of these suboptions as the first suboption in the option's information area.

The remaining suboptions carry the NetWare/IP configuration information itself.

`NWIP_DOES_NOT_EXIST`

Suboption code:	1
Length:	0
Data:	N/A

The `NWIP_DOES_NOT_EXIST` suboption indicates that the DHCP server does not have NetWare/IP information for the client.

NWIP_EXIST_IN_OPTIONS_AREA

Suboption code:	2
Length:	0
Data:	N/A

If this suboption is present, it indicates that all the NetWare/IP Information option information is contained in the options area of the DHCP message.

NWIP_EXIST_BUT_TOO_BIG

Suboption code:	4
Length:	0
Data:	N/A

The NWIP_EXIST_BUT_TOO_BIG suboption indicates that the DHCP server has NetWare/IP configuration information for the client, but the server cannot fit the information in the DHCP message.

NWIP_EXIST_IN_SNAME_FILE

Suboption code:	3
Length:	0
Data:	N/A

The NetWare/IP configuration information is in the sname field and, if necessary, in the file field in the fixed-format section of the DHCP message. If this method of transmitting the NetWare/IP configuration information is used, the NetWare/IP Information option appears in the options section, and it contains only the NWIP_EXIST_IN_SNAME_FILE suboption. The sname and file fields then contain (optionally) a NetWare/IP Domain Name option and a NetWare/IP Information option. The NetWare/IP Information option in the sname and file fields contains the NetWare/IP information in the suboptions described later in this section, but it does not include one of these first four suboptions.

NSQ_BROADCAST

Suboption code:	5	
Length:	1	
Data:	0	client should not use a NetWare Nearest Server Query
	1	client should use a NetWare Nearest Server Query

`NSQ_BROADCAST` specifies whether the client should use a NetWare Nearest Server Query to locate a NetWare/IP server.

`PREFERRED_DSS`

Suboption code:	6
Length:	*n*
Data:	list of IP addresses

`PREFERRED_DSS` lists the addresses of NetWare Domain SAP/RIP Service (DSS) servers. The maximum number of addresses carried in the `PREFERRED_DSS` suboption is five. The length must be a multiple of 4; if *s* servers are in the list, the `length` field is 4*s*.

`NEAREST_NWIP_SERVER`

Suboption code:	7
Length:	*n*
Data:	list of IP addresses

`NEAREST_NWIP_SERVER` lists the addresses of the nearest NetWare/IP servers. The maximum number of addresses carried in the `NEAREST_NWIP_SERVER` suboption is five. The length must be a multiple of 4; if *s* servers are in the list, the `length` field is 4*s*.

`AUTORETRIES`

Suboption code:	8
Length:	1
Data:	1 byte

`AUTORETRIES` specifies the number of times a client should attempt to contact a DSS server initially. The number of retries is encoded as an unsigned 1-byte integer.

`AUTORETRY_SECS`

Suboption code:	9
Length:	1
Data:	1 byte

`AUTORETRY_SECS` specifies the number of seconds a client should wait between attempts to contact a DSS server. The number of seconds is encoded as an unsigned 1-byte integer.

`NWIP_1_1`

Suboption code:	`10`	
Length:	`0`	
Data:	`0`	client should not use NetWare/IP version 1.1 compatibility
	`1`	client should use NetWare/IP version 1.1 compatibility

`NWIP_1_1` specifies whether the client should employ NetWare/IP version 1.1 compatibility to contact a NetWare/IP version 1.1 server.

`PRIMARY_DSS`

Suboption code:	`11`
Length:	`4`
Data:	IP address

`PRIMARY_DSS` specifies the address of the primary DSS server for the client.

SLP Options

Two options—`SLP Directory Agent Option` and `SLP Directory Agent Option`—provide information about the client's configuration for Service Location Protocol (SLP; see RFC 2608). These options are described in RFC 2610.

`SLP Directory Agent Option`

Option code:	`78`
Length:	*n*; 1 + 4 bytes for each address
Data:	`mandatory` flag, IP addresses

The `SLP Directory Agent Option` option gives the client a list of IP addresses for SLP servers. The `mandatory` flag controls the client's use of multicast discovery to locate SLP servers.

`SLP Directory Agent Option`

Option code:	`79`
Length:	*n*; 1 + number of bytes in data field
Data:	`mandatory` flag, list of SLP scopes

The `SLP Directory Agent Option` option gives the client a comma-delimited list of the scopes that the client's SLP agent should use. The `mandatory` flag controls whether the list of scopes in this option should override any scopes that are manually configured in the client.

UAP server

Option code:	71
Length:	*n*
Data:	list of IP addresses

The `UAP server` option lists addresses of User Authentication Protocol (UAP) servers that are available to the client. The length must be a multiple of 4; if *u* UAP servers are in the option, the length is $4u$.

name service search order

Option code:	117
Length:	*n*
Data:	list of names services, in the order in which they are to be consulted

The `Name Service Search Order` option specifies the order in which name services should be consulted when resolving hostnames and other information. This option is defined in RFC 2937. Each name service is identified by its corresponding DHCP option code, encoded as a 16-bit data value. The length must be a multiple of 2; if there are *s* name services, the length is $2s$.

subnet selection

Option code:	118
Length:	4
Data:	IP address in the subnet from which the client should be assigned an IP address

The `subnet selection` option allows the client to explicitly identify the network segment from which its address should be assigned. The DHCP server chooses an address for the client based on the IP address supplied in the option. This option is defined in RFC 3011.

When this option is present in a client's message, the IP address in the option is used in place of `giaddr` for the purpose of allocating an IP address. The motivation for this option is to allow DHCP proxy agents to acquire leases for clients on network segments to which the proxy agent is not connected.

authentication

Option code: `90`

Length: *n*

Data:

```
Protocol
Algorithm
Replay detection method
Replay detection
Authentication information
```

The `authentication` option carries information for authenticating the identity of DHCP clients and servers and to ensure that the contents of the DHCP message have not been altered in transit between clients and servers. The method for authentication of DHCP messages is described in the section "Authenticated DHCP Messages" in Chapter 7. The `data` field of the option is composed of several fields that define the authentication protocol in use, the algorithm used to generate the MAC for this message, the method and identifying information used for replay detection and the authenticating MAC.

relay agent information

Option code: `82`

Length: *n*

Data: relay agent information

The `relay agent information` option carries information about a DHCP client from a DHCP relay agent to a DHCP server. This option is defined in RFC 3118.

The data area of the `relay agent information` option is composed of one or more suboptions. The suboptions are encoded in the same way as DHCP options; each suboption includes a suboption code, a length, and the data.

The two suboptions defined in RFC 3046 are used by circuit access units, to pass information about the circuit to which the DHCP client is attached.

AGENT CIRCUIT ID

Suboption code: 1

Length: *n*

Data: circuit ID

The data in the `AGENT CIRCUIT ID` suboption is the circuit access unit's name for the circuit to which the client is attached.

`AGENT REMOTE ID`

Suboption code:	2
Length:	*n*
Data:	remote circuit ID

The data in the `AGENT REMOTE ID` suboption is the remote name for the circuit to which the client is attached.

Summary

The `options` section of a DHCP message carries values for most configuration parameters. These parameters are carried in options, whose formats are described in this chapter. Each option carries a separate configuration parameter, as defined by the option's option code. The data formats for most of the options are defined in RFC 2132. One group of options carries information that is specific to the operation of DHCP, identifying the type of each DHCP message and the server to which the message is directed. Other options carry information for the DHCP client, parameters for the client's TCP/IP software, and addresses of servers such as SMTP, NTP, and NIS servers.

Options range in complexity from the `TCP Default TTL` option to the `authentication` option. The `subnet mask` option, the `default routers` option, and the `DNS server` option are the most commonly used options.

Some options, such as the `Impress server` option, refer to services that are no longer available. Those options are still in the protocol specification, for backward compatibility with earlier versions of DHCP and BOOTP.

The IETF continues to identify and define new options for DHCP. For example, the `relay agent information` option, which was published in January 2001, allows relay agents to add additional information about DHCP clients to messages forwarded to servers. The initial definition of the `relay agent information` option, in RFC 3046, included two suboptions. Since the publication of RFC 3046, several additional suboptions have been defined as new uses for information from relay agents has been identified. Another new option, the `authentication` option, which was published in June 2001, allows clients and servers to confirm the source and validate the contents of DHCP messages.

10

Failover Protocol Operation

IN THIS CHAPTER

- Failover Protocol Overview
- Lease Handling with Failover
- Failover Operational States
- Binding Update Conflicts
- Pool Rebalancing
- Complex Failover Configurations

This chapter describes the DHCP failover protocol. DHCP provides for dynamic IP address allocation. In order to provide dynamic IP address allocation, a DHCP server must maintain a database of IP addresses, and it must maintain dynamic state for each IP address. Because of this, two DHCP servers that want to allocate IP addresses from the same pool must somehow cooperate to synchronize their database of IP addresses. Otherwise, both servers can allocate the same IP address to different DHCP clients. The failover protocol provides a reliable way for two DHCP servers to cooperate in allocating IP addresses out of the same pool.

The failover protocol also provides for disaster recovery. As soon as two failover peers have synchronized for the first time, either peer can safely and completely recover from the total loss of the other peer and all its data, even if the two servers are not communicating at the time of the failure.

NOTE

The failover protocol allows only two DHCP servers to share a particular set of IP addresses; there is no provision for three or more DHCP servers to share the same set of addresses.

This chapter is intended to give an overview of the failover protocol so that a network administrator can understand how it works and successfully operate a failover pair. If you are interested in implementing the protocol, you will find this to be a useful introduction, but it is by no means sufficient by itself to act as a reference for implementers.

To implement the failover protocol, you should consult the protocol specification. At the time of this writing, the failover protocol was under review for publication as an Internet Standard protocol. The specification is available as an Internet Draft titled draft-ietf-dhc-failover-10.txt. You can obtain the latest revision of this document from the DHC working group page on the IETF Web site (`www.ietf.org`). When the specification is accepted as a standard, it will be published as an RFC, which will be listed on the DHC working group Web page at `http://www.ietf.org/html.charters/dhc-charter.html`.

Failover Protocol Overview

The DHCP synchronization protocol is called the *failover protocol* because it was initially intended to provide a way for one DHCP server to act as a primary server and for a second DHCP server to act as a backup. In a very basic failover configuration, the secondary server does not provide DHCP service when it is in contact with the primary; it simply accepts updates from the primary. The secondary will start to provide DHCP service only if it loses contact with the primary. Thus, DHCP service will *fail over* from the primary to the secondary server. In a more advanced configuration, both the primary and secondary servers provide service at the same time, using a well-defined load-balancing algorithm to determine which server answers which requests.

Synchronizing the databases between two DHCP servers is relatively easy, as long as those two servers are able to communicate with each other. The central problem that the failover protocol solves is providing correct, reliable DHCP service in the face of a communication failure. There are several ways in which the primary and secondary server might lose contact with each other:

- One of the two servers might fail due to a hardware or software problem.
- The local network to which one of the servers is attached might fail.
- The network somewhere between the two servers might fail.

In the case of any of these failures, one server can't differentiate between a network failure and a server failure and can't tell if the other server might still be running. So, when the two servers are not in contact, each functions as if the other server is still running, adjusting its operation so that the DHCP service remains reliable, the server databases are not updated with conflicting information, and the two servers don't assign the same IP address to different clients.

Database Synchronization

The failover protocol uses a technique called *lazy updates,* in which each server tries to keep the other up-to-date but neither server is required to be entirely up-to-date in order for the protocol to function reliably. The servers follow a set of rules that prevents either server from behaving incorrectly in cases where updates have not yet been completed. This allows either failover peer to assign an IP address to a DHCP client before it has updated the other peer, which means that there is no performance penalty to the DHCP protocol as a result of using the failover protocol.

Another technique that is often used in distributed databases is a *three-phase commit protocol*, which allows both servers to present the same view of the database and means that both DHCP servers can behave almost identically. The problem with this technique is that it does not work when the servers are unable to communicate with one another, which is precisely the problem the failover protocol must solve. Also, in order for the three-phase commit protocol to work, the commit must be done before the address is offered to the client. This imposes an unacceptable delay between the time that a client requests a lease and the time the server confirms it.

Address Allocation Constraints

Lazy updates work by establishing a set of rules for how the DHCP servers allocate IP addresses. If both DHCP servers follow the same rules, there is no chance that both DHCP servers will ever allocate the same IP address to different DHCP clients, even if the servers are not in communication with one another.

The rules involve three principles. The first is that the primary and secondary failover servers divide the pool of free addresses that they have to serve on any given network segment into free and backup addresses. *Free addresses* are available for the primary server to allocate to clients. *Backup addresses* are available for the secondary server to allocate to clients.

The second principle is that DHCP servers can allocate or extend a lease only to a limited amount of time beyond the lease time known by its peer. This limited time is called the *maximum client lead time* (*MCLT*)—the maximum time that one server's idea of the lease's expiration time can *lead* the other's. The MCLT is typically quite short—certainly no more than an hour. The server can keep extending the lease by MCLT indefinitely, but when this happens, the client has to renew frequently. In order to allocate a longer lease to the client, the allocating server can cooperate with its peer to establish an *acknowledged potential lease expiry time*. When this time has been established, either peer can extend the client's lease for up to that amount of time plus the MCLT. Of course, because the acknowledged potential lease expiry time is a fixed point in time and not a duration, as the MCLT is, whenever a server extends a lease, it has to reestablish the acknowledged potential lease expiry time.

The third principle is that in normal operation, an address that has been assigned to one client cannot be assigned to another client unless both DHCP servers agree that the first client is no longer using it.

Communication Between Failover Peers

Failover peers communicate with each other by using a persistent TCP connection. The failover protocol is asynchronous—that is, either peer can send a message to the other peer at any time, and there is no restriction placed on the order of the responses.

Either failover peer can connect to the other; this allows a failover connection to be established as soon as the second failover peer starts, whether the primary or the secondary peer starts second. When a connection is established, whether the secondary or the primary peer initiated the connection, the primary peer sends a CONNECT message. This message contains identification and authentication information, as well as some information about how the primary peer is configured—in particular, what the MCLT is. If the secondary peer recognizes the primary peer and is able to authenticate it, it sends a CONNECTACK message. This message contains authentication information that is similar to that in the CONNECT message, as well as configuration information from the secondary peer. After these two messages have been successfully exchanged, the peers can communicate normally.

After the failover peers have established a connection, they tell each other what state they are in, and if necessary, the two peers synchronize their IP address databases. This process is described in more detail in the section "Operation in the RECOVER State," later in this chapter. When the servers initially connect, after any synchronization has been done, the two failover peers balance each address allocation pool, making sure that each peer starts out with roughly the same number of IP addresses.

During normal communication, when the DHCP server receives a DHCPREQUEST message from a client, it responds with a DHCPACK and then sends a binding update (BNDUPD) message to its failover peer. When the peer receives the update, it puts the update on a queue to be processed. After the update has been processed, the peer sends a binding acknowledgement (BNDACK) message in response. BNDUPD and acknowledgement messages are also used during the synchronization process.

As each failover peer assigns IP addresses to clients, the pool of free addresses may become unbalanced, with one peer having significantly more free addresses than the other. In this case, the peer that has fewer addresses performs the appropriate pool-rebalancing action, as described later in this chapter, in the section, "Pool Rebalancing."

During periods of inactivity, each peer sends periodic CONTACT messages to the other to probe for network outages. If no message is received from a peer for a certain period of time, the peer assumes that the connection has broken and begins operating independently. The connection between peers can also be terminated because one peer is being shut down; in that case, the server being shut down sends a DISCONNECT message to its peer, and then both peers close the connection.

Lease Handling with Failover

The DHCP has traditionally allowed IP addresses to be in one of two states. Either the lease expiry time for the address is in the past, meaning that the address is available to be allocated to a client, or the lease expiry time is in the future, meaning that the address is not available to be allocated. The duration that is assigned to leases is likewise very simple to calculate. The failover protocol introduces additional complexity both in terms of the number of states an IP address can be in and in terms of how the duration of a lease is calculated.

IP Address Binding States

The failover protocol specifies many possible IP address binding states. These states are used to indicate whether an address is in active use, which failover peer can allocate an address that is not in use, and an address's transition from being in active use to being available for allocation. Table 10.1 lists the complete set of possible binding states for IP addresses. An IP address can also be flagged to indicate that it is reserved for a particular client or that it is assigned to a BOOTP client.

TABLE 10.1 IP Address Binding States

State	Description
ABANDONED	The address has been abandoned as a result of an IP address allocation conflict detected by either server.
ACTIVE	The address is in active use by a client.
BACKUP	The address is available for allocation by the secondary peer.
EXPIRED	The address is no longer known to be in use by the client, but it is still bound to the client.
FREE	The address is available for allocation by the primary peer.
RELEASED	The address has been released by the client, but it is not yet available for allocation.
RESET	The address has been released through administrative action, but it is not yet available for allocation.

When a failover peer makes a change to a client lease, it sends a BNDUPD message to the other peer. The update includes the new state that the lease is in, the time that the change happened, the actual expiry time of the lease, and the potential expiry time of the lease, along with other information that identifies the client and possibly communicates information that the client sent. When the peer processes a BNDUPD, it sends back a BNDACK message.

After the first peer has received the BNDACK message, both peers have the same information about the lease. For example, if an address is in the BACKUP state on the primary peer, it is also in the BACKUP state on the secondary. When the servers are not operating normally, or when a change has been made on one server but the other server hasn't yet processed the update, IP addresses can be in different states on each of the failover peers. The rules controlling how to allocate IP addresses protect both peers from making mistakes when their IP address databases are not in synch.

Two IP address states indicate that an address is available for allocation. Addresses that are available for the primary server to allocate are in the FREE state. Addresses that are available for the secondary to allocate are in the BACKUP state. Addresses are never available for both servers to allocate at the same time.

When an address has been assigned to a client, whether by the primary peer or the secondary failover peer, it enters the ACTIVE state. An address in the ACTIVE state cannot be reallocated to another client until it reaches the FREE state or the BACKUP state. Therefore, either server in a failover pair can extend a lease.

A lease expires at the moment when its expiry time occurs. When the lease on an address expires, the address moves from the ACTIVE state to the EXPIRED state. The server making the change then sends an update to the other server. When the other server receives an update that moves an address from the ACTIVE state to the EXPIRED state, it moves the lease into the FREE state and sends an acknowledgement to the first server. When the first server receives the acknowledgement, it moves the lease into the FREE state. At this point, the lease is available for the primary peer to allocate.

This two-way handshake is required because after an address has been assigned to a client, either the primary or the secondary server can extend the lease. If the server that notices that the lease has expired moved the lease immediately into the FREE state, it might then allocate the lease to a new client while the other server was extending it for the original client.

This brings up an additional complication: After the server that has moved a lease into the EXPIRED state has sent a BNDUPD message to its peer, it can't extend the lease anymore. This is because when the peer receives the update, it immediately moves the lease into the FREE state. Even before the first server has received an acknowledgement, its peer may be able to allocate the lease to a new client.

Two other states are handled similarly to the EXPIRED state: RELEASED and RESET. When a failover peer receives a DHCPRELEASE message from a DHCP client, it places that client's lease into the RELEASED state. The RELEASED state is handled like the EXPIRED state in terms of how binding updates are done. It is also possible for an administrator to release an address. In this case, the address is moved into the RESET state, which is also handled like the EXPIRED state.

Assigning Lease Durations with Failover

In the absence of failover, the DHCP server has a clear process that it follows to compute the client lease duration. The client can request a lease duration in its `DHCPDISCOVER` and `DHCPREQUEST` messages. If it doesn't, the DHCP server assigns a default lease duration, which the network administrator can usually configure. The DHCP server can then check this duration against a minimum specified by the network administrator. If the duration is shorter than the minimum, the lease duration is increased to the minimum. The DHCP server can also check against a maximum specified by the network administrator, and again, if the lease is longer than the maximum, it is reduced to the maximum.

With the failover protocol, this lease duration is referred to as the *desired lease time*. The reason for this name is that this is the time that the server would *like* to give to the client. Whether or not the server *can* give this lease duration to the client depends on the state of the lease. There are three entities that remember a state for the lease: the DHCP client, the primary failover peer, and the secondary failover peer. The lease duration must be chosen so that the DHCP client will not believe that its lease expires later than either of the failover peers believes it expires. The DHCP server assigning the lease to the client can always remember when it assigned the lease to the client, so this is not the problem. The problem is the other server.

Let's say that the primary peer receives a `DHCPDISCOVER` message from a new client—one that has no active lease. The primary peer finds an address that is in the `FREE` state and allocates this address for the client. The secondary peer does not know yet that the primary peer has allocated this address. According to the rules in the section "Address Allocation Constraints," earlier in this chapter, the primary peer can't extend a lease by more than the MCLT. So the primary peer compares the MCLT to the desired lease time; if the MCLT happens sooner than the desired lease time, which is likely because MCLT is chosen to be short, the primary assigns an actual lease expiry time that is the current time plus MCLT.

If the client confirms this lease through the normal four-packet protocol, it will end up with a lease that expires at the actual least expiry time (which is the only lease time a client ever sees)—in this case the MCLT. At this point, the client probably has a lease that is shorter than the desired lease time. The short lease time will still work, and there is no way to avoid giving the client a short initial lease. After the initial lease assignment, the failover protocol tries to make it possible to give the client a lease that is close to its desired lease time.

The server gives the client its desired lease time by estimating when the client will renew its lease and computing a *potential lease expiry time*—the time when the client is expected to renew plus the desired lease time. The server can assume that the client will renew its lease halfway through. When the primary peer updates the secondary peer, it tells the secondary peer the actual lease expiry time and also the potential lease expiry time. When the secondary receives the BNDUPD message, it records the potential lease expiry time in its version of the IP address state. It then sends a BNDACK message. When the primary peer gets the BNDACK message, it knows that the secondary peer has the same potential lease expiry time that it has; this is the acknowledged potential lease expiry time.

Say that a client got a lease with a duration of MCLT. At one half of MCLT, it will try to renew its lease. At this time, the DHCP server will again compute the desired lease time, which will probably be the same as it was in the previous transaction. The acknowledged potential lease expiry time should be exactly that far in the future. So when the client renews, it gets a lease duration that is the desired lease time instead of MCLT. The server does the same computation as before, determining that the client will renew when one half of the desired lease time has expired. So it sets the potential lease expiry time to 1 ½ times the desired lease time, so that when the client renews again, it will again get a lease of the desired length.

The same sequence of events would occur with the secondary responding to the DHCPDISCOVER. The primary and secondary peers allocate IP addresses in exactly the same way, except that the secondary peer allocates IP addresses that are in the BACKUP state rather than the FREE state.

By computing the lease time in this way and maintaining a potential expiry time, a DHCP server that is operating normally in a failover relationship behaves toward the client almost exactly as if it were a DHCP server that is not running failover. The only real difference is that when a client first acquires an IP address or tries to reacquire an address that has expired, the first lease it gets will have a length of only MCLT instead of being the desired length.

Failover Operational States

This chapter has used the phrase *in normal operation*. There are quite a few different operational states for the failover protocol. In normal operation, the failover peers are cooperating with each other. There are two kinds of normal operation: a primary/backup configuration and a load-balancing configuration. When one peer is unable to communicate with another, it goes into the COMMUNICATIONS-INTERRUPTED operational state. If one peer is not operating, the other peer can be placed in the PARTNER-DOWN operational state, either by the administrator or through an automatic process. Several other temporary operational states exist to enable failover peers to make transitions from one major operational state into another or to resolve conflicts in an orderly way.

Normal Operation, Primary/Backup Configuration, and Load Balancing

In normal operation, or, to be more precise, in the `NORMAL` state, only one failover peer will normally respond to a DHCP message from a client. Which server responds depends on whether the client is operating in a primary/backup configuration or a load-balancing configuration. If the servers are operating in a primary/backup configuration, the choice of who responds is very simple: The primary peer always responds.

If the servers are operating in a load-balancing configuration, the decision about who responds is made via the standard load-balancing algorithm, which is a deterministic hash algorithm that operates on the client's identification information. This algorithm, defined in RFC 3074, is specified in enough detail that any DHCP server should obtain the same results as any other DHCP server. This means that two DHCP servers from different vendors should make exactly the same decisions about which clients to respond to. This is very important because if the two failover peers used different algorithms, some DHCP clients might receive service from both DHCP servers. Worse, some clients might get no response at all, even though both DHCP servers received their request.

The load-balancing algorithm allows the server administrator to configure each DHCP server to serve the opposite subset of all clients. When a DHCP server receives a broadcast message from a client, it performs a hash on the identification information the client provided. This hash produces a number between 0 and 255. The server has a 256-bit bitmap, and it uses the result of the hash to look up the corresponding bit in its bitmap. If the bit is nonzero, the server responds to the client's request, and if the bit is zero, the server drops the client's request.

The primary peer is configured with a bitmap, and it sends the exact opposite of this bitmap to the secondary peer in the `CONNECT` message. This ensures that both servers agree on which clients each is serving. Because each server has the exact opposite bitmap, for any given hash value, the value in the bitmap will always be 1 on one server and 0 on the other.

Some DHCP servers are programmed to detect that a client has not received a response from the other failover peer. The server makes this determination by examining the `secs` field of the DHCP client's message. This field indicates how many seconds it has been since the client sent its first message. If the value of the `secs` field is nonzero, it means that the client has sent at least one message for its current transaction. In this case, a DHCP server might choose to respond to the client even if it is in the `NORMAL` state and its hash bitmap tells it not to respond to the client.

During normal operations, each failover peer sends updates to the other whenever it assigns a new IP address to a client or renews a client's lease. Whenever an address is assigned from a pool, the failover peer that assigned it can check to see how many IP addresses it has left to assign and compare that to the number of addresses the peer has left to assign. If the other peer has significantly more addresses, the server that made the assignment can perform the proper pool-rebalancing operation, as described later in this chapter, in the section "Pool Rebalancing."

Operation in the COMMUNICATIONS-INTERRUPTED State

In the COMMUNICATIONS-INTERRUPTED state, neither failover peer can tell if the other is providing DHCP service, and so both peers provide DHCP service for all DHCP clients from which they receive requests. Because the servers are not communicating with one another, there is no way that the two servers can update each other, nor is there any way for them to balance their pools. If either server runs out of addresses to allocate, it must stop assigning new IP addresses to clients. However, either server can renew any client's lease, as long as that server has no record indicating that the lease has been allocated to a different client. Therefore, on a network with a reasonably large number of available addresses, DHCP service can operate indefinitely in the COMMUNICATIONS-INTERRUPTED state.

The COMMUNICATIONS-INTERRUPTED state has an additional disadvantage. Because the DHCP server can only allocate a lease that is as long as MCLT without first updating its peer, and because MCLT is generally chosen to be fairly short, clients have to renew frequently. This means that a server in the COMMUNICATIONS-INTERRUPTED state experiences a heavier-than-usual load. Also, because leases are short, clients' leases expire quickly, so a relatively brief outage on the remaining server could quickly cause some or all clients to lose their leases.

Operation in the PARTNER-DOWN State

For a variety of reasons, it is possible that one member of a DHCP failover pair might stop operating. This could be the result of a planned outage or an unplanned outage. In order to provide the best possible service when one member of a failover pair is down, the other can be placed in the PARTNER-DOWN state. When a server is in the PARTNER-DOWN state, the remaining DHCP server takes control of all DHCP service. When the remaining server has been placed in the PARTNER-DOWN state, it can assume that the other failover peer is definitely not running. This means that it can begin to reclaim the other failover peer's IP addresses, and it can make addresses whose leases have expired available to be allocated to other clients.

In the case of a planned outage, the administrator can place the server that is being shut down into the SHUTDOWN state. This server sends a `state` message to its peer to tell the peer that it is going into the SHUTDOWN state. When the other member of the failover pair sees that its peer has gone into the SHUTDOWN state, it automatically goes into the PARTNER-DOWN state. The peer that is being shut down then completes the shutdown process and exits.

A member of a failover pair could also fail unexpectedly. In that case, its peer quickly goes into the COMMUNICATIONS-INTERRUPTED state. As mentioned in the previous section, this is not very desirable, even though a server can run indefinitely in this state. If one failover peer is actually down and not just out of communication, the server administrator can place the other peer into the PARTNER-DOWN state.

It is also possible to configure a failover pair with a safe period. The *safe period* is the period between the time that a server enters the COMMUNICATIONS-INTERRUPTED state and the time that it concludes that the other server is no longer operating. If a safe period has been set, then when either peer goes into the COMMUNICATIONS-INTERRUPTED state, it sets a safe period timer. When this timer expires, the peer assumes that the other peer is in fact not operating, and it therefore makes the transition into the PARTNER-DOWN state. If the safe period is used, then it is possible that during a communications failure between the failover peers, the same IP address could be allocated to two different clients.

Even when peers are not in communication with one another, they can still extend leases, as long as they follow the rules described earlier in this chapter for determining the length of a lease. Because the failover peer that is running in the PARTNER-DOWN state knows that its peer has followed these rules, and because (as long as the safe period is not used) it cannot enter the PARTNER-DOWN state when the partner is running, it can reliably know when to reclaim IP addresses for which the peer may have extended the lease.

Whether a server got into the PARTNER-DOWN state because its peer went into the SHUTDOWN state while both partners were communicating, or whether it did so because the peer failed and the administrator directed it to enter the PARTNER-DOWN state, the server cannot be sure that it has received a complete set of updates from its peer. Because of this, the remaining server must treat any lease that its peer *could* have extended as if the peer *did* extend it—but the remaining server knows that the lease could not have been extended for more than the MCLT beyond the latest lease time that it has recorded.

When a server changes state, it remembers the *start time of state* (*STOS*). When a server enters the PARTNER-DOWN state, it can reclaim any available IP address (any address that is in the FREE or BACKUP state) that belongs to its peer after MCLT plus STOS has passed. If an address is in the ACTIVE, EXPIRED, RELEASED, or RESET state and the acknowledged potential expiry time is later than STOS, the server can free the IP address after the acknowledged potential expiry time plus MCLT, or after MCLT plus STOS, whichever comes last. This is because the failed peer may have extended the lease to the acknowledged potential expiry time plus MCLT without telling its peer, but it may also have extended it to MCLT plus STOS, and the server in the PARTNER-DOWN state has to account for both possibilities.

The failover protocol specification suggests that failover servers should be able to be told when the partner actually failed. It may be the case that the partner failed in the middle of the night, but nobody was available to place the remaining server into PARTNER-DOWN until morning. In this case, there is no reason to wait for MCLT after STOS to expire before reclaiming addresses; it is sufficient to wait for MCLT after the peer failed. Failover servers are not required to provide this functionality, but it is certainly useful to be able to provide it.

After a failover server in the PARTNER-DOWN state has reclaimed all the leases that belonged to its peer, the MCLT has passed, and all the acknowledged potential expiry times have expired, the remaining peer operates autonomously, as if it were not running the failover protocol at all. It continues to do so until its partner connects to it. Because it is operating autonomously and is not following the lease timing constraints described earlier, it is crucial that when its peer is restarted, the restarted peer must not provide DHCP service until it has resynchronized with the peer that is in the PARTNER-DOWN state, as described in the section “Operation in the RECOVER State,” later in this chapter. If the restarted peer were to begin DHCP service without resynchronizing, it would be operating on the basis of old information, and it might make disastrously incorrect address assignments.

Operation in the STARTUP State

The STARTUP state is a temporary state that all failover servers enter upon startup. When a DHCP server starts, it waits for a certain period of time before serving clients. During this time, it attempts to establish contact with its peer. This allows the server to take note of any change in its peer’s state before it begins to operate. For example, if its peer is in the PARTNER-DOWN state, it will know not to start serving clients again until it has received a complete update from its peer.

There is one special circumstance for the startup state—when the server that is starting has no recollection of ever having communicated with its peer. In that case, the server starts in the RECOVER state, which is discussed in the following section. When a site switches from running two servers that are not cooperating to running two servers that are cooperating, it is likely that *both* servers will start up in the RECOVER state.

Operation in the RECOVER State

The RECOVER state lets a failover server get a complete update from its peer when it starts up and discovers that the peer has been in the PARTNER-DOWN state. It is also used when a failover server starts up and has no record of ever having communicated with its peer. When a failover server enters the RECOVER state, it stops providing service to DHCP clients because it can't consider its state to be correct. It attempts to establish communications with its peer if it is not in communication with its peer.

When the server has established communication, it sends an UPDREQ (update request) message to its peer. The peer goes through its entire list of addresses and sends a BNDUPD message for each address whose state has changed since the last time it communicated with the recovering server in the normal state. If the recovering server has no record of having communicated with its peer, it sends a UPDREQALL (update request all) message, meaning that the partner should send the state of every IP address for which there has been a transaction. This update process reliably synchronizes the recovering server's IP address database to its peer's database. After the peer has sent the last update, it sends an UPDDONE (update done) message to the recovering server, indicating to the recovering server that the update process is complete. If both servers restart in the RECOVER state, each server still behaves in exactly the same way: They both send updates to each other at the same time.

After the server in RECOVER state has received the UPDDONE message, it moves into the RECOVER-WAIT state. The RECOVER-WAIT state exists to let the recovering server wait until MCLT has expired before it begins serving DHCP clients. This allows any leases that the recovering server may have assigned after it lost contact with the other server to expire. This is important when the recovering server is being restored as a result of an incident that caused it to lose its database. In cases where the recovering server has been down longer than MCLT, this waiting period is not required.

When the waiting period indicated by the RECOVER-WAIT state has expired, the recovering server moves to the RECOVER-DONE state. If its peer is in the PARTNER-DOWN state or the RECOVER-DONE state, the peer moves into the NORMAL state. When the recovering server sees that its partner has moved into the NORMAL state, it too moves into the NORMAL state and begins providing normal service to DHCP clients.

Operation in the POTENTIAL-CONFLICT State

The RECOVER state exists to allow servers to make an orderly recovery in the face of a normal outage. However, it is possible for servers to become out of sync due to an administrative error or another incorrect state transition. There are three scenarios that can lead to this, and all three involve one or both of the servers operating in PARTNER-DOWN state while the other is operating in COMMUNICATIONS-INTERRUPTED or PARTNER-DOWN state. This can happen because a communications break between the two peers lasts longer than the configured safe period (see the section "Operation in the PARTNER-DOWN State"). It can also happen if a failover server fails and its peer is placed into PARTNER-DOWN state, but when the failed server comes back up it is unable to communicate with its peer. It is also possible that a server administrator might place one peer in the PARTNER-DOWN state when the other peer is actually operating in the COMMUNICATIONS-INTERRUPTED state, and this would also lead to a conflict.

When a conflict is detected, both servers immediately make a transition into the POTENTIAL-CONFLICT state and stop serving DHCP clients. Because this is an uncontrolled event, it is possible that both servers have assigned the same IP address to different clients. However, both servers should still remember the last time they communicated in the NORMAL state. Therefore, each server must send updates for every IP address whose state has been modified since that time.

When the primary peer enters the POTENTIAL-CONFLICT state, it immediately sends an UPDATE-REQUEST message to the secondary peer. When the secondary peer enters the POTENTIAL-CONFLICT state, it waits for an UPDATE-REQUEST message from the primary peer. When it receives the UPDATE-REQUEST message, it sends the primary BNDUPD messages for all the IP addresses whose state it has modified and for which it has not received a corresponding BNDACK message from the primary peer. After all the updates have been sent, the secondary peer sends an UPDATE-DONE message to the primary peer. When the primary peer receives the UPDATE-DONE message, it makes a transition into the CONFLICT-DONE state, at which point it is responsive to DHCP client requests. When the secondary peer sees the primary peer's state change, it sends an UPDATE-REQUEST message to the primary peer, and the primary peer sends it updates for all the IP addresses whose states the primary peer changed while the peers were out of communication. The primary peer finally sends the secondary peer an UPDATE-DONE message, and the secondary peer makes the transition into the NORMAL state. When the primary peer sees this state transition, it too makes the transition to the NORMAL state.

When either peer sees the other make the transition into the NORMAL state, it can check to see whether its pool of free addresses is low, and if it is, it can perform the proper pool-rebalancing process.

Operation in the CONFLICT-DONE State

As mentioned in the previous section, the conflict-done state is a temporary state that the primary peer enters after it has finished receiving updates from the secondary peer in the POTENTIAL-CONFLICT state. In the POTENTIAL-CONFLICT state, the primary peer should act pretty much as it does in the NORMAL state, except that it should not do load balancing because the secondary peer is not responding. If communication with the secondary peer is lost while the primary peer is in the CONFLICT-DONE state, the primary peer remains in CONFLICT-DONE, and the conflict-resolution process resumes when communication is restored.

Operation in the RESOLUTION-INTERRUPTED State

It is possible that during the conflict-resolution process described in the section "Operation in the POTENTIAL-CONFLICT State," the connection between the two failover peers may be broken. Any server that is still in the POTENTIAL-CONFLICT state when this happens makes a transition into the RESOLUTION-INTERRUPTED state. In this state, servers respond to DHCP clients in a limited way—they may extend existing leases, and they may allocate addresses out of their free pools.

There is a possibility that either server may create an IP address conflict while in this state because the IP address databases are not synchronized. A cautious implementation might choose not to be responsive in the RESOLUTION-INTERRUPTED state. Of course, there is a danger in this kind of caution: If something like this happens when the network administration staff is unavailable for an extended period of time, there might be no DHCP service at all. This is why the failover protocol specification suggests a less cautious strategy.

Binding Update Conflicts

One point that we've glossed over so far is what happens when a failover peer processes a BNDUPD message. Almost by definition, the information in an update is different from the information that the server processing the update has for the IP address being updated. Sometimes the change proposed in the update can be accepted and sometimes it can't. The failover protocol specification provides a scheme for figuring out whether to accept or reject an update. If the server processing the update has to reject it, it sends a BNDACK message to the partner, but it includes a failover option called the REJECT-REASON option. When the peer receives a BNDACK message rejecting an update (that is, a message that contains a REJECT-REASON option), it has the option of trying the update again later. In some cases, the server receiving the update should send its own update to the peer to correct misinformation in the peer's binding database, but in this case, the two servers need to be careful to avoid getting into a loop where they rapidly send each other competing update information.

Pool Rebalancing

As a failover pair operates normally, it is likely that the pool of free addresses and the pool of backup addresses will shrink and grow at different rates. This is particularly true because when leases expire, they always enter the FREE state and never the BACKUP state. This means that the pool of free IP addresses tends to shrink more slowly than the pool of backup addresses. In practice, in a stable environment where few clients are leaving the network, however, the difference in size between the pools is governed by the load-balancing strategy that the administrator chooses.

In order to avoid having both servers constantly rebalancing pools, the responsibilities of each are different. When the secondary peer notices that the pool of available addresses has become significantly unbalanced in favor of the primary peer, it sends the primary peer a POOLREQ (pool request) message. When the primary peer receives the POOLREQ message, it examines all of its pools, and if it finds any that are out of balance, it assigns enough addresses in each pool to the secondary peer to bring things roughly back into balance, by sending the secondary peer a BNDUPD message for each IP address it is making available to the secondary, changing the binding state of that IP address from FREE to BACKUP.

When the primary peer discovers that a pool is significantly out of balance in the secondary peer's favor, the primary peer reclaims backup addresses from the secondary peer in order to bring the pools back into balance. It does this by sending the secondary peer a BNDUPD message for each address it wants to reclaim, moving the address from the BACKUP state to the FREE state. If the secondary peer has assigned the address to a client but not yet told the primary peer about it, the secondary peer will refuse the update.

Complex Failover Configurations

So far in this chapter, we have used the terms *server* and *failover peer* interchangeably. This suggests that one DHCP server can have a failover peering relationship with only one other DHCP server and that all IP address pools on each server must be shared with the failover peer. This is not the case; each address pool can have its own primary and secondary servers. It is also possible to have one DHCP server with more than one address pool; the DHCP server can share some address pools with other failover servers and keep other address pools private.

Figure 10.1 shows a central server that has four separate peering arrangements, each with a different server. Two servers, West and East, have pools that they operate independently, and they also have pools that they share with the central server. Two other servers, North and South, have only one failover relationship each, each with the central server, and they serve no pools independently.

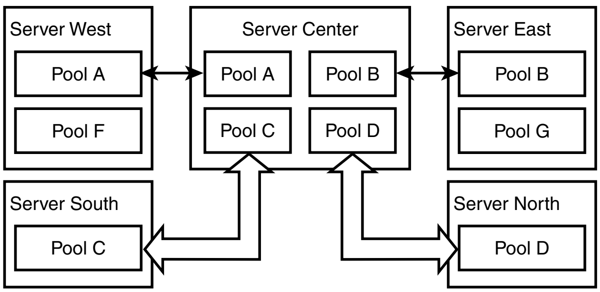

FIGURE 10.1 Failover peering between a central server and four local servers.

Each of the central server's four peering relationships operates independently. It is possible for the relationship between Center and East to be in the `NORMAL` state at the same time that the relationship between Center and South is in `COMMUNICATIONS-INTERRUPTED` state and the relationship between Center and North is in the `PARTNER-DOWN` state. If East is in the `RECOVER` state, it will not serve IP addresses out of the pool it is sharing, Pool B, but it will serve IP addresses out of Pool G, which is not shared.

Summary

The failover protocol allows pairs of DHCP servers to share address pools. It provides a reliable method whereby servers can share pools even when communication between the peers has been interrupted. It provides a mechanism for recovering from the total loss of either failover peer by allowing a newly reconstituted failover peer to request all the lease state database information from the existing failover server.

The failover protocol uses a scheme called lazy updates to keep failover peers loosely synchronized without requiring that failover peers maintain perfectly consistent databases at all times. This minimizes the performance impact that failover has on the DHCP protocol itself.

Failover provides a way for one DHCP server to take over all DHCP service when the other has experienced a long-term failure, and it provides a way for the servers to begin cooperating again after the peer has been restored to service.

11

DHCP–DNS Interaction

IN THIS CHAPTER

- The Domain Name System
- DHCP and DNS
- Dynamic Updates to the DNS Database
- Dynamic Updates and DHCP
- How the DHCP Server Updates the DNS

The Domain Name System (DNS) is a service in which a distributed database maintains mappings between domain names and IP addresses. When you contact `www.dhcp-handbook.com`, your Web browser uses DNS to find the IP address for that Web site.

DHCP manages the assignment of IP addresses, and the DNS manages the translation of IP addresses to names and names to IP addresses. In order for a DHCP client to be identified by name on a network, it must somehow establish a name in DNS, or a name must be established for it. This chapter describes the ways in which DHCP servers and clients interact with DNS and the difficulties and potential pitfalls in these interactions.

NOTE

For a more detailed description of DNS, see *DNS and Bind* by Cricket Liu and Paul Albitz or *Internetworking with TCP/IP: Principles, Protocols, and Architecture* by Douglas Comer.

The Domain Name System

DNS is a database that is stored on a widely distributed collection of servers throughout the Internet. The names in the DNS comprise a tree-structured hierarchy of names—or namespaces—and each DNS server manages its own part of the namespace. DNS servers at the top of the hierarchy provide a means for discovering which DNS servers serve the lower parts of the hierarchy. The administrator for each DNS server establishes policies about DNS name assignment and management for the parts of the namespaces managed by that server.

Resource records (RRs) represent DNS data and are associated with names in the DNS namespace. Many different types of RRs exist, and each is associated with a different type of information. The DNS specification defines these resource record types (RRtypes) and acknowledges that additional RRtypes continue to be proposed as the needs of the community evolve.

The `A` (address) resource record maps a domain name to an IP address. The `PTR` (pointer) record maps a domain name to another domain name and is usually used for reverse mapping of IP addresses to domain names. There can be more than one `A` record attached to a domain name, so a single domain name can map to more than one IP address. It is also permissible to have more than one `PTR` record attached to a domain name, but that is not common.

The `A` record is useful because humans can more easily communicate domain names than IP addresses. It is much easier to tell someone that he or she should connect to `sherchin.fugue.com` than it is to tell the person to connect to 10.0.1.5. This is particularly true with DHCP, with which it is not always possible for a particular computer, particularly one that roams, to have a consistent IP address.

A *reverse mapping* is a mapping from an IP address to a name. The domain `in-addr.arpa` contains a hierarchy of names that represent IP addresses. To convert from an IP address to a name, you convert each octet in the address into a label and arrange the labels in most- to least-specific order, from left to right. For example, the IP address `192.168.11.25` is represented in the `in-addr.arpa` domain as `25.11.168.192.in-addr.arpa`. This is called a reverse mapping because the mapping should point to a name with an `A` record that points back at the IP address being mapped.

Reverse mappings are useful for pretty much the same reason that `A` records are useful: Computer users generally have an easier time recognizing domain names than IP addresses. For example, if I want to see what computers are connected to my computer right now, I will be much more interested to know that, for example, `kwanyin.fugue.com` (my wife's computer) has connected to mine than to know that a computer whose IP address is 10.0.1.4 has connected. This information is really only a convenience—it is not guaranteed to be correct—but it is useful.

DHCP and DNS

Traditionally, names on a DNS server are managed by the network administrator. The network administrator learns that a new machine is to be connected to the network and either comes up with a name for it or negotiates with the owner of the computer about what the name should be. Sometimes the owner of the computer wants a name that has already been taken, but that is an easy problem to resolve over the telephone.

Maintaining a sensible mapping of domain names to IP addresses is complicated by DHCP. When a network administrator's work of keeping track of IP addresses has been automated by a DHCP server, the network administrator won't necessarily know when a new computer is connected to the network. Network administrations who use registration systems and only allow registered hosts to connect may not have this problem, but many networks allow connections by unregistered hosts. On such networks, it is still useful for computers that connect using DHCP to have names and reverse mappings published in the DNS, but in such a case, the process of choosing a name and configuring it in DNS can't easily be done by the administrator.

One solution is to populate the DNS server with preassigned domain names for all the addresses that DHCP manages. Because the names of the clients are not known when DNS is being configured, the network administrator simply makes up a name for each IP address being managed by DHCP. The name could be imaginative, or it could be something mechanical, perhaps based on the IP address. For example, if a DHCP client on the GSI internetwork is assigned 192.168.11.25, its name in DNS could have been previously configured to be `net11-host25.genericstartup.com`. The DNS database is preconfigured with an `A` record that maps `net11-host25` to 192.168.11.25 and a `PTR` record that maps `25.11.168.192.in-addr.arpa` to `net11-host25.genericstartup.com`.

There are a couple problems with this. First, `net11-host25.genericstartup.com` is not much easier to remember than 192.168.11.25. This problem could be solved by simply choosing names that are really names—for example, the names of streets in the local city. Second, the client's name changes whenever its IP address changes. For a desktop computer in an office, that isn't a big problem, but for a roaming laptop or for a customer of a broadband ISP that does dynamic IP address allocation, it isn't very useful.

Dynamic Updates to the DNS Database

RFC 2136, "Dynamic Updates in the Domain Name System," describes a mechanism that allows DNS client programs to automatically make changes to the DNS database, using DNS protocol messages. This means that the DHCP server or the DHCP client can directly change the `A` and `PTR` records for a particular IP address, without the intervention of the network administrator.

The dynamic update mechanism enables clients to supply prerequisites, which are conditions about data in the DNS zone that must be satisfied before the DNS server performs an update. A DNS server performing a dynamic update first checks all the prerequisites in the update request. If all those prerequisites are met—that is, if all the conditions specified as prerequisites are true—the server performs all the requested changes to the DNS data. For example, DHCP clients and servers use prerequisites in DNS update messages to detect duplicate domain names.

Dynamic Updates and DHCP

DHCP clients and servers can use dynamic DNS updates to register domain names and reverse mappings for DHCP clients automatically as they are assigned leases. As a server assigns an address, the server and possibly the client use DNS update messages to add or update the `A` record for the client's domain name and the `PTR` record for the IP address assigned to the client.

You need to consider some questions when you use dynamic DNS updates from DHCP clients and servers:

- What computers do you trust to do dynamic updates?
- Does the DHCP client or the server select the client's name?
- Does the client or the server perform the dynamic update?
- What is the relationship between the duration of a client's lease on an address and the time-to-live (TTL) on the DNS entries for that client?
- What happens if two DHCP clients select the same name?
- What happens if a client moves?
- What happens if a client's name changes?

Together, three protocol specifications describe these problems and how to solve them. All these specifications are currently IETF drafts, meaning that they haven't yet been approved by the IETF. The drafts are titled "A DNS RR for Encoding DHCP Information (`DHCID` RR)," "Resolution of DNS Name Conflicts Among DHCP Clients," and "The DHCP Client FQDN Option." Much of what is explained in this chapter is described in more detail in these protocol specifications.

DHCP Client DNS Name Selection

Three entities can potentially determine the DHCP client's published domain name. We have already talked about one—if the administrator sets up a static IP address-to-domain name mapping in the DNS and does not allow DNS updates, DNS determines the DHCP client's published domain name. With DNS updates, however, it is possible for either a DHCP server or client to determine the domain name.

There are good reasons to allow the client to choose its name, and there are also good reasons for the server to choose—or at least control—the client's choice of name. Which method is used varies from site to site, depending on the needs and preferences of the network administrators and users at each site.

If the DHCP server controls name assignments, it can use the following mechanisms:

- The server can be configured with a name for each client.
- The server can generate a name for each client, based on some algorithm or heuristic.
- The server can generate a name for the client, based on a name supplied to the server by the client when it attempts to get an IP address.

One way for the server to determine the client's domain name is for the network administrators to set up a client registration system. In such a system, the user or the administrator chooses a name for the client and enters it into the registration system. The registration system validates the name (by making sure it is not forbidden and that some other client isn't already using it) and then enters the name into the DHCP server's configuration. Whenever the DHCP server assigns the client an IP address, it updates DNS, using the new IP address and the name assigned to the client. This means that the DHCP client's domain name stays the same even when its IP address changes, but this still allows for central administration of domain name assignments.

In a more typical configuration, the DHCP server forms the name by appending the local domain name to the hostname that the DHCP client sends. Many operating systems now encourage the owner of a computer running that operating system to configure the computer with a hostname. For example, if my laptop sends the name `sherchin` to the DHCP server at work, the DHCP server might form the domain name `sherchin.nominum.com`. This is a very natural way to associate a name with a particular DHCP client because it means the user doesn't need to be told what his or her computer's domain name is—the user knows the domain name because he or she typed it in. This system also doesn't require a special registration application because the user or system administrator chooses the name when setting up the computer the first time.

Some operating systems allow the user to specify a fully qualified domain name (FQDN) such as `sherchin.fugue.com` instead of a simple hostname such as `sherchin`. The reason for doing this is to allow a DHCP client to roam from one administrative domain to another while retaining a consistent, globally accessible domain name. For example, my laptop can always have an `A` record pointing from `sherchin.fugue.com` to its current IP address. This works whether I am at Nominum's corporate headquarters, where the company's domain name is `nominum.com`, or whether I am at home in Chicago, connected through my DSL provider. For this to work, the DHCP server can't be involved in determining the hostname—it has to trust the name supplied by the client.

Responsibility for Performing DNS Updates

Which entity performs the DNS update? Because the DHCP server controls IP address allocation, it is natural for it to update the PTR record associated with each IP address it allocates. The real question is whether the client or the server should update the A record.

The main issue to consider when deciding whether clients or servers should update DNS is a key-management issue. A DHCP client or server needs to have credentials in order to update DNS. These credentials need to be unique to a particular updater—if they are not, then it's impossible to restrict the set of records that any given updater is allowed to change.

Typically there are fewer DHCP servers than DHCP clients, so a site that has the DHCP server do the DNS updates needs to keep track of fewer keys than a site that allows DHCP clients to do the updates. At the time of this writing, the authors are not aware of any large sites where DHCP clients are allowed to update DNS directly. However, at smaller sites, where trust issues can be handled on the basis of personal relationships between network administrators and users, it is practical to allow DHCP clients to update DNS on a case-by-case basis.

If the DHCP client is to update its own A record, it chooses the name for that A record. After it has received an IP address from the DHCP server, it sends the update to the DNS server that manages the namespace containing that name. However, the DHCP server still has to update the PTR record. The DHCP server and the DNS server may be in different administrative domains. The Internet Draft titled "The DHCP Client FQDN Option" defines a DHCP option that the DHCP client and server use to communicate about the FQDN that the client wants, who should do the update, and what the DHCP server should store in the PTR record.

When a client is configured to do its own update, it sends an FQDN option in each message it sends to the DHCP server. The FQDN option sent to the server contains the fully qualified domain name the client has chosen. The option also carries a flag that indicates whether the client wants the server to do the update of the A record; if the flag is not set, the client asserts that it will be doing the A record update.

When a DHCP server receives an FQDN option in which the update flag indicates that the client will update its own A record, the DHCP server can choose to honor the client's request, or it can follow its own local policy and assign the client a name of its own choosing. This can be independent of the site's policy on allowing clients to update DNS. If the client indicates that it can update the DNS, the server can just assume that the client knows what it's doing.

The DHCP server sends an FQDN option in its response to the client. If the server chose not to honor the client's intention to update its own A record, it sends an FQDN option telling the client what name the server assigned it, and it also sets a flag in the option that tells the client not to update its own A record. The client is, of course, free to ignore this flag, but its reverse mapping points to the name of the server assigned it, not to the name it chooses. If the server chooses to honor the client's intention, it sends to the client an FQDN option that contains the same domain name the client sent as well as a flag that tells the client that the server is cooperating with it.

The FQDN option is intended to be the standard way that a DHCP client tells the DHCP server the name that it wants to use, whether the client or server updates the DNS. If a client only has a hostname, rather than an FQDN, it should send that hostname to the DHCP server, using the FQDN option. Older DHCP clients use the hostname option to tell the DHCP server their names.

NOTE

Some versions of the Microsoft DHCP client that implements the FQDN option attempt to update their own A records by using GSS-TSIG authentication, a special variant of transaction signatures (TSIG) that uses the Kerberos protocol to manage keys. They do this by default, even if they don't share credentials with the DNS server for the namespace that would contain their A records. This is mostly harmless because the updates can't succeed, but it can generate a lot of noise in the DNS server's log. The easy workaround to this problem is to configure the DHCP server not to cooperate with clients that want to update DNS. When the Microsoft client receives the information from the server that the server doesn't want it to do the DNS update, it doesn't attempt an update. This creates a problem at sites that would like to support client updates on a case-by-case basis, though, because updates are disabled for all clients—not just for Microsoft clients that haven't been configured correctly. The ISC DHCP server works around this problem by noticing that the FQDN option from these Microsoft clients contains just a hostname, not a FQDN, and tells the client not to do the update in this case.

DHCP Client Name Collision

If the users of DHCP clients are permitted by local policy to select their own hostnames, one problem that can arise is that two users may choose the same name for their computers. Currently no provision exists in the DHCP protocol to resolve such conflicts, although (as discussed in Chapter 21, "DHCP Clients") the Microsoft DHCP client tries to use the Windows Internet Name Service (WINS) protocol to resolve naming conflicts.

The Internet Draft titled "Resolution of DNS Name Conflicts Among DHCP Clients" refers to the situation in which two DHCP clients have the same name as a name collision, or a name conflict, and it provides a method that clients and servers can use to detect this. When a DHCP server or client updates an A record, it attaches a `DHCID` record to the same name to which it is attaching the A record. The `DHCID` record contains data that identifies the client: the value of the DHCP `client-identifier` option that the client sent in its DHCP request or, if it did not send a client identifier, its link-layer address. Because the `DHCID` record is publicly accessible in DNS, the client's identification information is masked by hashing it, along with the client's FQDN, using the MD5 algorithm. This produces an identifier that does not reveal any information about the client but can be reliably regenerated from the client's identification information.

Whether the A record update is performed by the DHCP client or by the server, the updater can use the prerequisites section of the update query to ensure that no other client is currently using the name in question. The updater first sends an update request for the `A` and `DHCID` records to the DNS server. The update has a prerequisite that prevents the update from being done if the name exists in DNS.

If the name being added already exists, the updater attempts to delete all `A` records currently attached to the name and add the new one. This time the update has a prerequisite that if the name exists and has attached to it a `DHCID` record that does not match the `DHCID` record that the updater generated based on the client's identification information, the update will fail. If the prerequisite test in this second request succeeds, the DNS server updates the `A` record for the client to reflect the client's new IP address and leaves the `DHCID` record in place.

NOTE

This second DNS update can fail for one of two reasons. If a different client's `DHCID` record exists, then some other client is using the domain name in question. If there is no `DHCID` record, then the name was entered by something or someone other than a DHCP server—probably the network administrator. In either of these cases, the new client doesn't get the name it asked for. DHCP does not provide a mechanism for the client to choose a new name; if the client can't have the name it wants, the DHCP server must either assign it a different name or not give it a name at all.

Lease Expiration

When a lease expires, or is released, the updaters of the `A` and `PTR` records try to remove these records from the DNS database. This prevents the DNS database from becoming overloaded with stale names, and it frees disused names for reuse. It also allows a client with two network interfaces to associate its name with the interface it is using at any given time because the client identifier is usually associated with an interface and the updater is using the client identifier to determine the owner of the name.

DHCP servers and clients keep records of names they have added so that they can remove them later. However, the mere fact that a DHCP agent remembers adding a name does not mean that the records currently associated with that name in DNS are still the records that the agent added. A name to which an A record has been assigned by a DHCP agent should have a DHCID record as well, and the A record associated with the name should be the same one that the DHCP agent added. So when a DHCP agent removes an A record and the name associated with it, it places a prerequisite on the DNS update that it sends. The prerequisite is that the A record that the agent added must still be attached to the name, and the DHCID record that is attached to the name must match the DHCID record the server used to add the name. If both of these conditions are true, the DNS server removes the name.

Because the DHCP server controls the IP address, the server is responsible for updating and deleting the reverse mapping that corresponds to the address. To delete the reverse mapping, the DHCP server simply sends an update that deletes the in-addr.arpa name.

If the DHCP server adds the A RR, it is also responsible for deleting the A and DHCID RRs when the client's lease expires. If the DHCP client adds its own A record, it is responsible for removing the records before its DHCP lease expires (that is, before it releases its lease).

NOTE

Removal of A records by clients is another of the protocol issues that touches on sites' administrative policies. If sites decide to allow DNS updates from their DHCP clients, their clients should be reliably able to maintain the DNS information that they add, or additional manual administrative effort may be required in order to police their DNS updates. Clients may be shut down and removed from the network without warning as individuals move about and as organizations replace hardware. This means that it is unlikely that the client will remove its A record beforehand. When the client gets a new lease, it corrects this problem, but during the period that it is not connected to the network, its A record is incorrect.

Client Relocation

Many organizations that use dynamic DNS also use multiple DHCP servers. Suppose a site has two buildings and uses a separate DHCP server in each building. What happens when a laptop is shut down at a user's desk in Building A and is restarted on the network in a conference room in Building B?

The DHCP server in Building A (Server A) doesn't know that this happened. It knows only that the client has an active lease with it. When the DHCP server in Building B (Server B) leases an IP address to the client, it expects to add a PTR record that corresponds to the IP address and an A record that corresponds to the client's domain name. However, Server A already had an A record (one that contains the IP address that Server A leased to the client). The client's A record should represent the most recent active lease, so Server B attempts to add an A record and a DHCID record for the client's domain name.

Server B generates a DHCID record, either from the client's client-identifier or from its link-layer address. Server B sends to the DNS server a DNS update message. The message includes the prerequisite that the client's FQDN does not exist. The update message adds an A record and a DHCID record. The data in the A record is the new IP address, and the data in the DHCID record is the client identifier, as described in the section "DHCP Client Name Collision," earlier in this chapter.

This update fails because the FQDN is already present, reflecting the existing lease from Server A. When Server B is notified of this failure, it then forms another update message. This message includes a prerequisite that the DHCID record for the client's domain name match the DHCID that Server B generates. The message deletes any existing A records and then adds a new A record for the IP address that Server B assigned to the client. The update in the message does not try to modify the DHCID record. The DHCID prerequisite should match in this example, so the update should succeed. After the update has been completed, the client's name should have just one A record, pointing to the client's IP address in Building B.

Server A sees none of this activity. As far as it is concerned, the client has a valid lease on an IP address in Building A. The client's lease on its IP address in Building A may expire while the client is in Building B. When that lease expires, Server A will try to delete the A record that it added when the client was in Building A. The deletion will fail because Server B already deleted the A record that Server A added. Following the rules specified in the section "Lease Expiration," earlier in this chapter, maintains the consistency of the name database.

There is one problem here, though. In order to avoid constantly sending DNS updates for names that already exist in the DNS database, the network administrator may configure the DHCP server not to update DNS if it has a record that says it already updated DNS. So what happens if the client goes from Building B back to Building A *before* the original lease in Building A expires? In this case, the client will renew its lease, but the DHCP server may not update DNS because it thinks it has already done so. So the client's A record will remain in DNS, pointing to the IP address assigned by Server B, but the client will actually be using the IP address assigned by Server A. A DHCP server that implements this optimization must therefore detect that the client has changed networks and not do the update optimization in that case.

Client Name Change

If a client changes its name while it has a valid lease, when it renews its lease, the DHCP server detects that the client's name has changed and tries to remove the A and DHCID records from the old name, just as it would if the client's lease expired.

It then adds the name to DNS again, using the same rules described in the section "DHCP Client Name Collision."

DNS Dynamic Update Security Issues

Dynamic updates to the information in DNS represent a significant potential for security problems. Without some restrictions on the acceptance of dynamic updates, anyone can send a dynamic update request to a server to change the IP address associated with a domain name. So, if an adversary wanted to intercept all the traffic aimed at `www.genericstartup.com`, he or she could simply send a dynamic update to `dns.genericstartup.com` that changes the IP address for `www.genericstartup.com` to his or her computer's IP address.

The least secure form of update identification is to restrict the set of IP addresses from which DNS updates can be received. Many DNS servers provide this capability. The problem with this technique is that DNS is a UDP-based protocol, and it's very easy to send a forged datagram with an IP source address that the attacker knows is permitted to do updates. There is one case in which address-based authentication might provide sufficient security: when the DHCP server and the DNS server are running on the same computer. There is still some risk of a spoofed update; the network administrator should be sure that the host on which the DHCP and DNS servers are running will reject forged datagrams with a source and destination address of 127.0.0.1 if they arrive on a network interface other than the loopback interface.

DNS provides a standard mechanism, called TSIG, that can be used to authenticate DNS updates. TSIG signatures are generated by using a secret that the updater and the DNS server share. By checking the signature in the update packet, using this shared secret, the DNS server can be sure that the update came from an updater that possesses the key. Therefore, as long as the key is kept secret, TSIG updates can be trusted.

TSIG works very nicely for DHCP servers because the DHCP server is under the control of the network administrator, so the server can be trusted to follow the rules with respect to updating the zone. TSIG works less well for DHCP clients than it does for DHCP servers. The problem is that in order to make sure that clients update only their own records, every client has to have its own secret key. Setting up a special secret key for each individual DHCP client that needs to update a zone is a lot more work than setting up one key for each DHCP server. Sites with only a few clients may find this worthwhile. Sites with more clients may have more difficulty.

How the DHCP Server Updates the DNS

It is important to know something about how the DHCP server actually figures out how to update the DNS. Without that knowledge, it can be difficult to debug problems. First, you must understand that the DNS namespace is broken up into zones on the basis of authority records (usually referred to as SOA records). SOA records define which DNS server is responsible for the information in a given namespace. The definitive version of the information in a namespace has to reside on a single server, known as the *primary server*. There are two other kinds of name servers—secondary and caching servers. *Secondary servers* mirror data that is on primaries, and some secondary servers can accept DNS updates, but only the primary server for a zone can make changes to it. Name servers that act as primary or secondary servers for certain zones are said to be *authoritative name servers* for those zones. *Caching servers* remember names that they have looked up in the past and keep information about those names in their cache. When the same name is looked up repeatedly, the caching name server answers each subsequent query itself rather than consulting an authoritative name server.

In order for the DHCP server to update the DNS, it has to identify the zone it needs to update, and then it has to send the update to the primary server for that zone. A *zone* is a portion of the DNS hierarchy. DNS starts with a root zone, which is at the top of the tree. Within the root zone are all the top-level domains, which are all separate zones—for example, `.com.` is a top-level domain. Underneath the top-level domains are organizational domains, such as `fugue.com.`, which is my home domain. Organizational domains are run by the organizations that own them, or by DNS service providers. It is a safe bet that the top three levels of the domain hierarchy are all separate zones. However, it is not required that each label be in its own zone. It is possible for `manhattan.fugue.com.` and `bisbee.fugue.com.` to both be part of the `fugue.com.` zone. It is also possible for them to be separate zones.

The DHCP server figures out the primary server for a given name by asking DNS, using the DNS resolver, which is an operating system service for looking up records in the DNS. For example, if the server needs to update `samten.bisbee.fugue.com.`, it first checks to see if `samten.bisbee.fugue.com.` is a zone, by asking DNS for an SOA record for the domain. If there is an SOA record for `samten.bisbee.fugue.com.`, then the DHCP server sends the update to the DNS server named in that SOA record. If there is not, the DHCP server tries to find an SOA record for `bisbee.fugue.com.`, and then `fugue.com.`, and then `com.`, and then `.`. Of course, it won't have to traverse all the way up to `.`, but in an organization with a deep hierarchy, it might have to try a few times before it gets the SOA record it needs.

One consequence of this is that it is difficult to short-circuit the process for testing purposes. Because the structure of the DNS is contained within the DNS, if you want to set up a test server but don't want to publish it, you have to somehow tell the DHCP server to send updates to the test server and not to the IP address that it would find if it queried the DNS. Because the DHCP server searches up the hierarchy rather than starting from the top and moving down, this is possible; if you configure the DNS resolver on the DHCP server machine to use the test DNS server for name resolution, then when the DHCP server asks which server is authoritative for the zone it wants to update, the test server claims that it is authoritative, even though it really isn't.

There is an additional complication. TSIG keys are generally defined on a per-zone basis or for some set of zones. Therefore, the TSIG key `mykey.fugue.com.` might work for updating `manhattan.fugue.com` but not for `bisbee.fugue.com`. The DHCP server or client has to provide a way for the administrator to tell it which key to use with which zone. Some DHCP servers may be able to accomplish two goals at once with this configuration process. If the DHCP server can be configured with the IP address of the IP address of the primary server for the zone as well as the TSIG key to use when updating the zone, the DHCP server doesn't have to search for the zone to update.

One other consideration when using TSIG is that TSIG updates do not work if the updater and the server do not have clocks that are synchronized to within five minutes of each other. This is a result of the way TSIG prevents replay attacks. In a replay attack, an attacker captures DNS update messages on the network and stores them. Later, it replays one of these messages, to make the DNS server redo the update. This puts invalid data into the DNS server. To prevent this, DNS agents that generate the messages put timestamps on each message that is signed with TSIG. The DNS server checks each timestamp, and if that timestamp differs from the current time by more than five minutes, the server discards the message.

Summary

DNS provides a mapping between mnemonic names for networked devices and the IP addresses assigned to those devices. The mapping information is stored as a database whose contents are distributed among DNS servers throughout the Internet.

As DHCP servers assign IP addresses to clients, either the DHCP server or client must add database entries to the DNS database or the DNS database must already contain forward and reverse mappings for the client.

The dynamic DNS update mechanism described in RFC 2136 defines a mechanism through which DNS messages can update—not just query—the DNS database. Three protocol specifications describe how RFC 2136 can be used in combination with DHCP services: “A DNS RR for Encoding DHCP Information (`DHCID` RR),” “Resolution of DNS Name Conflicts Among DHCP Clients,” and “The DHCP Client FQDN Option.” These specifications define a set of techniques that can be used to reliably keep the DNS database synchronized to the DHCP lease database.

PART III

DHCP Servers and Clients

IN THIS PART

12

Theory of the Operation of a DHCP Server

IN THIS CHAPTER

- Address Allocation Strategy
- Allocation and Renewal in Response to a DHCPREQUEST Message
- DHCP Message Handling
- Abandoned Lease Address Reclamation

This chapter describes the actual operation of a DHCP server. It uses as an example the operation of version 3.0 of the ISC DHCP server because the source code for this server is readily available; interested readers can follow along and see how it is implemented. In addition, one of the authors is intimately familiar with version 3.0's operation. Although the operation of some of the features of the DHCP server is ISC-specific, this discussion should be meaningful to users of other DHCP servers as well.

Address Allocation Strategy

A DHCP server allocates IP addresses to clients according to the configuration set up for the server by the DHCP administrator. IP addresses are allocated either dynamically or statically. In *dynamic allocation,* a client can be allocated any address out of an address pool. (An *address pool* is a list of IP addresses that are available for allocation on a particular network segment.) When the server receives a request for an address from a client, the server looks through its configuration and identifies an address from an appropriate address pool to be allocated to the client.

Each address pool is associated with the network segment for the subnet declaration in which it is defined. If a network segment is configured with more than one IP subnet, then any address pool for that network segment can contain IP addresses from any IP subnet on that network segment. Any address pool can have an *access control list*—that is, a list of tests to run on a client to see whether the server can allocate an address out of the pool for that client. There can be more than one pool per network segment, so that different clients on that network segment can get different IP addresses.

In *static allocation,* you define an address for a specific host in the DHCP server configuration. The ISC server uses a host declaration to identify a host. The host declaration can contain a fixed address, specifying one or more IP addresses for static allocation. These addresses are allocated only to the client that matches the host declaration.

NOTE

The ISC DHCP server does not check for conflicts between IP addresses declared in `fixed-address` statements and IP addresses declared in address pools. This means that if you declare an address range that includes an IP address that is also in a `fixed-address` declaration, the DHCP server might assign the same IP address to two different clients. This check is not done because a DHCP server with a large configuration file that contains many host declarations would take too long to start.

NOTE

The reason that it is possible to provide more than one statically allocated IP address to a single client is that the client might have network interfaces on more than one network or it might need to be able to receive statically allocated IP addresses as it roams among different network segments.

Allocation in Response to `DHCPDISCOVER` or `BOOTREQUEST` Messages

When a server receives a `DHCPDISCOVER` or `BOOTREQUEST` message, it attempts to allocate an IP address for the client that is sending the message. It first looks for host declarations that match the client and that contain fixed-address declarations. It checks each IP address in these declarations to see if it is valid on the network segment to which the client is connected. If it finds an IP address that is valid, that IP address is always assigned to the client. Otherwise, it looks to see whether the client has an existing lease on the network segment to which it is connected that is either still valid or hasn't been reused since it expired. If it finds such a lease, it checks to see whether the client is permitted to use the address. If the client is permitted to use the address, the DHCP server assigns the client that address. (Reasons the client might not be permitted to use the address are described in a later section called "Address Use Denied.")

If a client has specified an address by using the `requested-address` option and that address is on the network segment to which the client is connected, the server checks to see whether that address is available and whether the client is allowed to use it. If the address is available and the client is allowed to use it, the server allocates that address for the client.

If the client still doesn't have an IP address, the server goes through the list of pools for the network segment to which the client is connected and tries to find one that is free and that the client is permitted to have. If it finds one, it allocates it to the client. If there are no free addresses that the client is allowed to have, the server logs a `no free leases` message and doesn't respond to the client.

In searching each pool, the server first looks for an address that has never been associated with a client, and then, if it doesn't find one, it looks for a previously assigned address that is now available. The server continues to search through all the address pools for the network segment to which the client is attached, looking for an address that has never been associated with a client, until it finds one or runs out of pools.

Address Assignment

After the server identifies an address that can be allocated to the client, it sends an ICMP echo request to that address, to see whether the address is already in use. It waits for about one second for a response, and if it doesn't receive one, it sends a message to the client with the address that was allocated.

If the client is a DHCP client, the DHCP server sends a `DHCPOFFER` message to the client that contains the address that the DHCP server has allocated, as well as all the parameters the server intends to send to the client. The lease is not yet final, so the DHCP server does not write the lease to disk, but it does reserve the lease in its in-memory database for two minutes, to give the DHCP client time to confirm the lease with a `DHCPREQUEST` message.

If the client is a BOOTP client, the server sends a `BOOTREPLY` message to the client with the client's new address. This is the end of the transaction for a BOOTP client. If the client's address is a statically assigned address, the server writes no information into the lease database. If the client's address is dynamically allocated, before the server sends the `BOOTREPLY` message, it writes the address to the lease database so that when it is restarted, it will remember that it has dynamically allocated the address to the BOOTP client. Because BOOTP has no concept of a "lease", and a BOOTP client considers its address to be permanently allocated, the DHCP server records the dynamically allocated address for a BOOTP client as a permanent assignment with an infinite lease.

If the server receives an ICMP echo reply message in response to its ICMP echo request, it marks the address as abandoned and logs a message indicating that this has occurred so that an administrator can take action. A permanent lease is assigned to that IP address, and it is marked as abandoned. The abandoned lease is immediately written to the lease database so that the server does not attempt to allocate the address again later. The server does not send any response to the client in this case.

Allocation and Renewal in Response to a DHCPREQUEST Message

After a DHCP client receives a DHCPOFFER message, if it chooses to select the address the server offered, it responds with a DHCPREQUEST message, requesting the offered address. Clients with leases also send DHCPREQUEST messages from time to time, to renew their leases. When the server receives a DHCPREQUEST message, it has three choices:

- It can ignore the message.
- It can respond with a DHCPNAK message.
- It can respond with a DHCPACK message.

If the server recognizes the address the client is requesting as its own—that is, if it is in an address pool declaration or a static address assignment—and the address is valid for the network segment to which the client is attached, the server checks whether the client has permission to continue using the address. If the client does not have permission to continue using the address, the server sends the client a DHCPNAK message.

Sometimes a server sends a client a DHCPNAK message when it is not strictly allowed by the protocol. For example, if the client is in the RENEWING state, a DHCPNAK message is unexpected, but the server can't reliably tell that a client is in the RENEWING state, so it might send a DHCPNAK message to the client anyway. Sending a DHCPNAK message in this situation does not cause problems.

If the server recognizes the IP address being requested as its own, or if it is declared to be authoritative, the server checks the address to see whether it is valid for the subnet to which the client is attached. If it is not valid, the server responds with a DHCPNAK message.

NOTE

The authoritative keyword indicates whether the network administrator believes that the DHCP server has been configured with complete information about the network segments it is serving. One of the things this affects is whether the server sends DHCPNAK messages aggressively. Many DHCP servers do not send DHCPNAK messages aggressively, which causes problems when clients roam to new networks. A DHCP server should allow the user to control this parameter so that users who are setting up DHCP servers that are not authoritative can avoid causing problems, while network administrators can get the more efficient DHCPNAK behavior that an authoritative DHCP server can provide.

If the address is valid for the network segment to which the client is connected but the server does not recognize the address, the server does not respond, even if the client specifies the server's address in the `dhcp-server-identifier` option.

Lease Extensions

If the address the client is requesting is valid for the network segment to which the client is attached and is an address the server can assign, and if the client is allowed to use the address, the server computes the new expiration time for the lease. If the client requests a particular lease duration, the server makes sure the requested lease time is within a range specified by the `min-lease-time` and `max-lease-time` parameters. If the requested lease time is not within the specified range, it is set to the value of `min-lease-time` if it is too short or to the value of `max-lease-time` if it is too long.

If the client does not request a specific lease duration, the lease duration specified in the `default-lease-time` is used, and the same limits are applied.

After the server computes the duration of the lease, it records the lease on disk and waits for the operating system to confirm that the data was written to disk. It then sends a `DHCPACK` message with the computed lease duration to the client and writes a log message. This sequence of events ensures that the DHCP server will never forget, because of a power failure or system crash, that it has assigned a lease to a client.

Address Use Denied

There are several reasons a client could be unable to continue using its old address. If the client is connected to a different network segment than it was when the address it is requesting was allocated, it cannot continue using its old IP address because that address simply wouldn't work.

Another reason a client might not be able to continue using an address is that a server administrator changed some configuration information, such as the pool permit list. A *pool permit list* is a list of permits that apply to a particular pool. The permits for a pool are checked against the client before an address from that pool is allocated to the client. If the client matches all the permits, it is allowed to receive an address from the pool; otherwise, it is not. Permits control access on the basis of whether a client is a BOOTP or DHCP client, for example, or whether a client is a member of a class. Classes are discussed in more detail in Chapter 20, "Conditional Behavior."

If the client received its address because it matched a pool's permit list, but the administrator changed the permit list and the client no longer matches it, the server refuses to give the client the same IP address. It is also possible for the information that the client sends to change so that the client no longer matches the permit list.

When this happens, the server also refuses to allocate the same IP address. For example, suppose a subnet has two address pool declarations, as shown in Example 12.1.

Example 12.1

```
pool {
  range 10.0.0.10 10.0.0.99;
}

pool {
  range 10.0.0.100 10.0.0.199;
}
```

Suppose that when a particular client first acquires its address, it is assigned the IP address 10.0.0.10. After the client acquires this address, the administrator decides to move all clients out of the 10.0.0.10–10.0.0.99 pool. The administrator modifies the pool configuration as shown in Example 12.2 and then restarts the DHCP server.

Example 12.2

```
pool {
  deny all clients;
  range 10.0.0.10 10.0.0.99;
}

pool {
  range 10.0.0.100 10.0.0.199;
}
```

When the client tries to renew its lease on IP address 10.0.0.10, the server checks the permit list for the pool from which the address is allocated and finds that the client can no longer use the address.

DNS Updates

If the DHCP server is doing DNS updates (see Chapter 11, "DHCP–DNS Interaction"), it sends the DNS update before it writes the lease entry or sends the `DHCPACK` message. If the DHCP client supports the `FQDN` option, the DHCP server reports the result of the update to the DHCP client in the `DHCPACK` message.

If the server revokes a client's lease by sending it a `DHCPNAK` message and it had previously done a DNS update for that client, the server removes the DNS information for the client.

If the client fails to renew its lease, when the lease expires, the DHCP server removes the lease from the lease database. When the DHCP server is stopped and then restarted, it checks to see whether any leases expired while it was stopped, and if so, it removes any DNS entries that had been entered for those leases.

DHCP Message Handling

DHCP clients can send a variety of other messages during operation, including `DHCPDECLINE`, `DHCPRELEASE`, `DHCPINFORM`, and `DHCPLEASEQUERY` messages. The following sections discuss the handling of these messages.

`DHCPDECLINE` Message Handling

A DHCP client can send `DHCPDECLINE` messages if it is assigned an IP address that another client is using. The client detects this by sending an ARP request for the IP address it is assigned when it receives a `DHCPACK` message while in the `REQUESTING` state. If it receives a response to this ARP message, the client knows that the address it received is in use, so it sends a `DHCPDECLINE` message, indicating to the DHCP server that the client cannot use the address.

The ISC DHCP server normally honors `DHCPDECLINE` messages from clients, creating permanent leases for such addresses, marking them as abandoned (see the section "Address Assignment"), and logging a message indicating that this happened. When an address is marked as abandoned, it is not normally allocated to any client. If a client is sending `DHCPDECLINE` messages as a denial of service attack, the server reclaims abandoned leases according to a strategy described later in this chapter. This prevents such denial of service attacks from causing great harm, although they still place an additional load on the DHCP server.

`DHCPRELEASE` Message Handling

When a server receives a `DHCPRELEASE` message from a client, it changes the expiration time of the lease to the current time and records the lease. A message is logged, indicating that this happened.

`DHCPINFORM` Message Handling

When the server receives a `DHCPINFORM` message, it collects the parameters that are appropriate for the network segment that the client claims to be connected to and sends them to the client in a `DHCPACK` message. No address allocation is performed. The server prints log messages, indicating that the `DHCPINFORM` message was received and that the `DHCPACK` was sent. If the server has not been configured to be authoritative, the server ignores and does not respond to `DHCPINFORM` messages because the information it has may not be correct.

DHCPLEASEQUERY Message Handling

The DHCPLEASEQUERY message is very new, and it is not supported by many DHCP servers, including version 3.0 of the ISC DHCP server. However, this section talks about how it might be handled by a DHCP server.

Unlike other DHCP messages that are sent to the DHCP server, the DHCPLEASEQUERY message comes from a DHCP relay agent, not from a DHCP client. Relay agents, particularly in broadband ISP environments, frequently append routing information to DHCP client messages before forwarding them to DHCP servers, using the Relay Agent Information option. These routers generally cache the DHCP server's response, which gives the router a mapping between the client's IP address and information about how to forward datagrams to the client. When such a router is rebooted, it loses this cached information.

When the router then receives a datagram for a particular IP address that is no longer in its cache, it sends a DHCPLEASEQUERY message to the DHCP server that contains the IP address in the ciaddr field. The DHCP server then looks up that IP address in the lease database and returns whatever information it has to the relay agent, by using the DHCPACK message. If the IP address hasn't been assigned to a client, the DHCP server returns a DHCPNAK message instead.

In order for the DHCPLEASEQUERY message to be useful, the DHCP server must store the information that the relay agent sent in the Relay Agent Information option. DHCP servers frequently do this anyway because the information is useful in other situations such as identifying the client or associating the client with its location on the network.

Abandoned Lease Address Reclamation

The DHCP server attempts to reclaim abandoned leases in situations where it appears that the lease is abandoned due to an implementation error in the DHCP client. If a DHCP client sends a DHCPREQUEST message, requesting an abandoned IP address, the server removes the flag which indicates that the lease is abandoned and enables the client to renew the lease.

If there are no IP addresses available for allocation but there are abandoned addresses, the DHCP server reclaims one of the abandoned IP addresses and attempts to assign it to a client. The server uses an ICMP echo check, as described in early in this chapter, to check on the availability of the address and it abandons the lease again if the address still appears to be in use. If the address does not appear to be in use, the server assigns it to the client.

NOTE

The *lease database* is a simple log-structured file, and each lease update is written to the end of it. It is common for there to be several entries in the lease database for the same IP address. Without some kind of compaction algorithm, the file simply grows without bound until the disk on which it is stored becomes full. To prevent this from happening, the server rewrites the lease database from its in-memory database once every hour. After the file is rewritten, the old version is saved as a backup and the new one is moved into place.

Summary

The DHCP server runs continuously as a background process, processing client requests as they arrive. When a client request for a new IP address arrives, the server must attempt to allocate an address, either from the dynamic pool or by using a prearranged static assignment. If an address is found for the client, the server ensures that it is not in use and then offers it to the client.

When a client tries to renew a lease, the server must decide whether the IP address the client is renewing is appropriate for the client. It must tell the client to stop using the address, tell the client it can continue using the address, or remain silent.

If the server finds an IP address that should be available but is nonetheless in use, it does not assign the address to a client but marks it as abandoned. The server does not attempt to use addresses marked as abandoned unless a shortage of IP addresses occurs during some subsequent address allocation attempt. If such a shortage occurs, the server again attempts to allocate abandoned addresses.

The server maintains a persistent database of leases that are confirmed to DHCP clients and does not send a message to the client to confirm a lease until the lease is written to the persistent database.

13

The Microsoft DHCP Server

IN THIS CHAPTER

- Installing the Microsoft DHCP Server
- Managing DHCP Servers
- Configuring DHCP Servers
- Controlling the Windows DHCP Server

The Microsoft DHCP service is distributed as part of Windows 2000 Server and it includes two components: the DHCP server and the DHCP manager. The DHCP server is responsible for all interactions with DHCP clients, and the DHCP manager provides a user interface for controlling and configuring the server.

This chapter describes how to install, configure, and manage the Microsoft server. Appendix A, "Microsoft DHCP Server Examples," includes additional examples and screen shots of Microsoft server use.

Installing the Microsoft DHCP Service

You install the Microsoft DHCP service on Windows 2000 with the Add/Remove Programs option in the Control Panel. To bring up this option, select the Start menu, and then select Settings, Control Panel, as shown in Figure 13.1. Double-click the `Add/Remove Programs` icon, and then select the `Add/Remove Windows Components` icon.

Next, select Networking Services and click the Details button. Scroll down and select the Dynamic Host Configuration Protocol (DHCP) entry, as shown in Figure 13.2, and click the OK button. This process loads both the DHCP server and the DHCP manager.

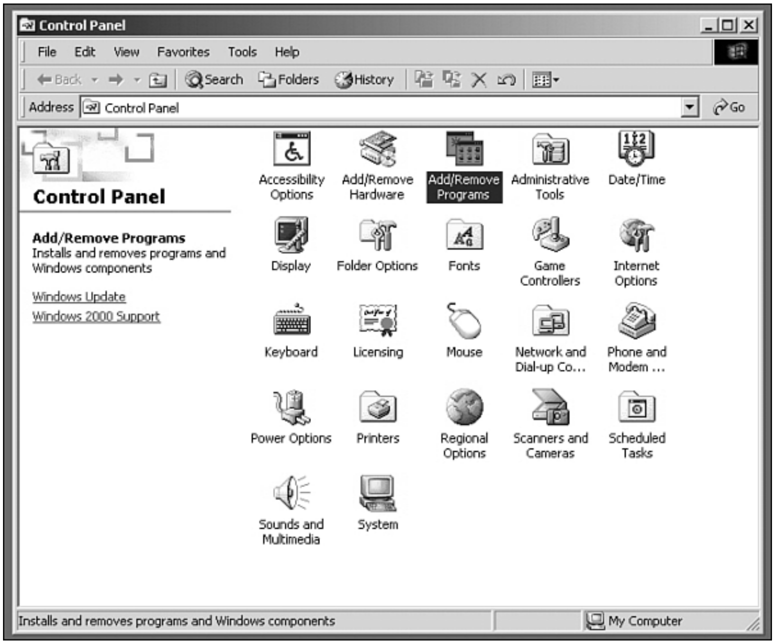

FIGURE 13.1 Starting the Add/Remove Programs icon from the Control Panel.

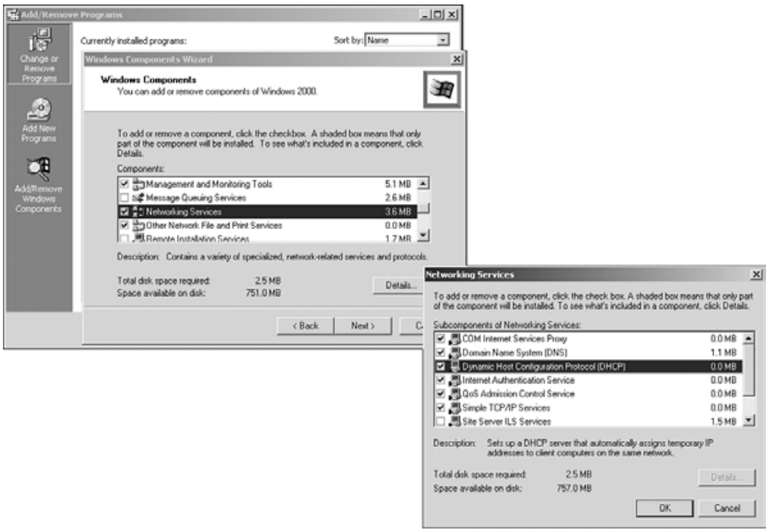

FIGURE 13.2 Installing the Microsoft DHCP service from the Add/Remove Programs dialog box.

When the Microsoft DHCP service is installed, DHCP server is configured to start automatically (along with other services) when you start Windows 2000. However, the server doesn't respond to incoming client requests until you configure and activate the scopes, as described in the next section of this chapter. You can control the operation of the server by using the Service option from the Control Panel, which is described in more detail later in this chapter.

You must fully configure the Microsoft DHCP server before activating any scopes; otherwise, the server might respond to client requests with incomplete or incorrect configuration information. It might also interfere with the operation of clients that are managed by other servers, by inappropriately sending `DHCPNAK` messages for requests from network segments that are not yet included in the server's configuration.

Managing DHCP Servers

The Microsoft DHCP service can include multiple DHCP servers, which are controlled through the DHCP manager. The installation process configures the DHCP manager for a DHCP server that is running on the same computer as the DHCP manager. If you plan to use other DHCP servers, you must configure the DHCP manager for the additional servers.

You start the DHCP manager by clicking the Start menu and then selecting Programs, Administrative Tools, DHCP, as shown in Figure 13.3.

To add a new server, select Action, Add Server. The DHCP manager displays the window shown in Figure 13.4. Enter the name or IP address of the new server and click OK. The DHCP manager contacts the new server and adds it to its list of available servers.

Configuring DHCP Servers

The Microsoft server configuration is defined around *scopes*, which are similar to the *subnet* declarations for the ISC server. Each scope defines a network segment, the addresses available for assignment within that network segment, and the subnet mask for the segment. The definition of a scope in the Microsoft server performs the same function as the `subnet` and `range` settings in the ISC configuration file and is unrelated to the concept of scopes and scoping in the ISC DHCP server.

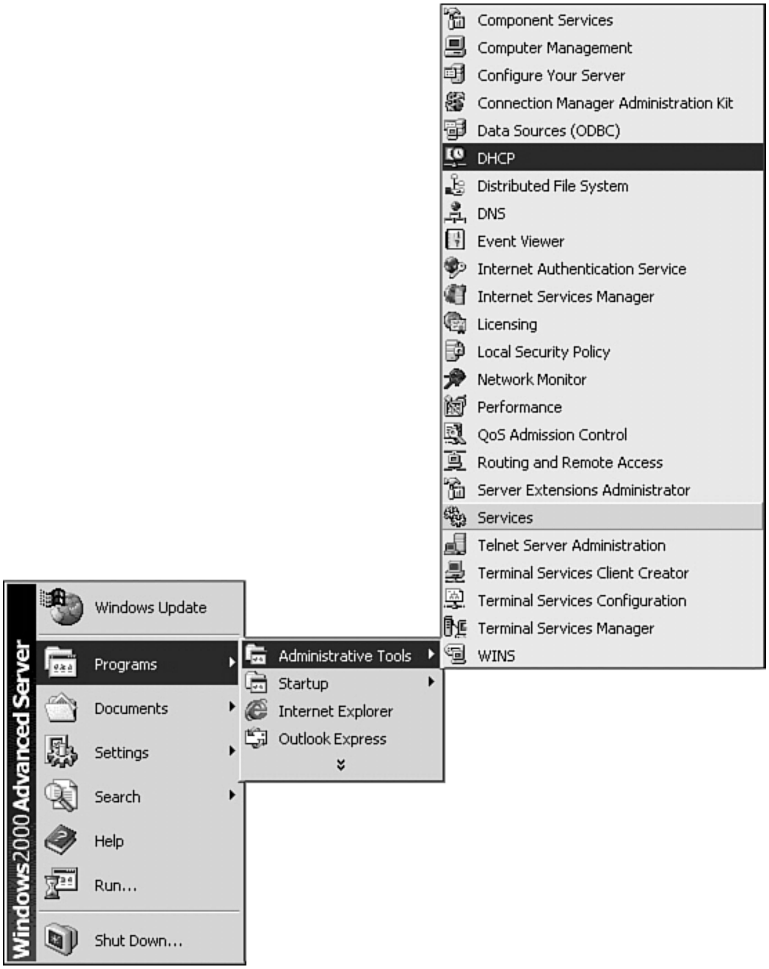

FIGURE 13.3 Starting the DHCP manager.

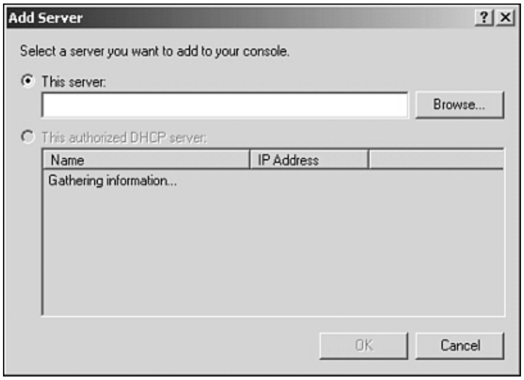

FIGURE 13.4 Adding a new DHCP server.

Configuring Scopes

To configure the Microsoft server, you need to start by defining and configuring the scopes the server will manage through the DHCP manager. Start the DHCP manager and select the entry for the DHCP server you want to configure. This section uses the `dhcp-example` entry, which identifies the DHCP server on the computer used in this chapter, as an example. Select `New Scope` from the Action menu. The DHCP manager begins the New Scope Wizard; through this wizard you are prompted to enter the name and an optional comment for the scope, the available addresses in the scope, and the subnet mask. Figure 13.5 shows the dialog box you use to enter the name and the description of a new scope.

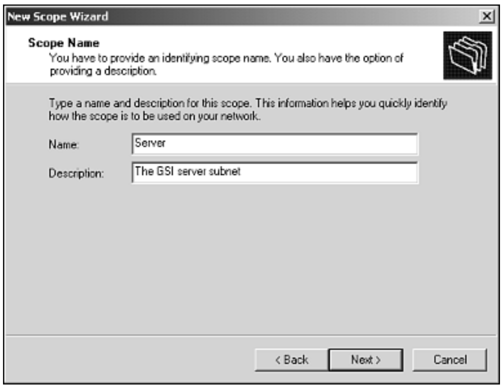

FIGURE 13.5 Creating a new scope in the DHCP manager.

Naming Scopes

Each scope defined for the Microsoft DHCP server can have an associated name. You use this name to identify the scope in the list of available scopes displayed by the DHCP manager. Because the DHCP manager displays the scopes in alphabetical order, you might want to define a naming convention that groups related scopes together by assigning names with the same prefix to those scopes.

Available and Excluded Addresses in a Scope

Within a scope, you define the set of addresses that are available for assignment by the DHCP server. You can also designate a list of excluded addresses that the DHCP server should not assign. You can define the list of available addresses so that it contains just the addresses in the scope that are available for the server to assign. However, each scope definition can contain only one list of available addresses; each scope definition must be contiguous and specified by the first and last addresses in the list. Therefore, if you want the list of available addresses to include noncontiguous addresses, you must declare the list of available addresses to span all the available addresses and then specifically exclude some addresses.

To illustrate how to define available addresses to the Microsoft DHCP server, suppose a network segment is defined to have IP address 192.168.11.0 and subnet mask 255.255.255.0 and that the DHCP server is to assign addresses 192.168.11.10–192.168.11.19 and 192.168.11.30–192.168.11.39. In this example the corresponding declaration in the configuration file for the ISC server is shown in Example 13.1.

Example 13.1

```
subnet 192.168.11.0 subnet-mask 255.255.255.0 {
  range 192.168.11.10-192.168.11.19;
  range 192.168.11.30-192.168.11.39;
}
```

Figure 13.6 illustrates the definition of the list of available addresses through the New Scope Wizard. Note that the subnet mask for the scope is also defined in this dialog box.

You can configure the Microsoft DHCP server for the available addresses shown in Figure 13.6 by setting the range to 192.168.11.10–192.168.11.39 and then putting the range 192.168.11.20–192.168.11.29 in the list of excluded addresses. Figure 13.7 shows the next dialog box in the New Scope Wizard, in which the addresses 192.168.11.20 through 192.168.11.29 have been excluded from assignment.

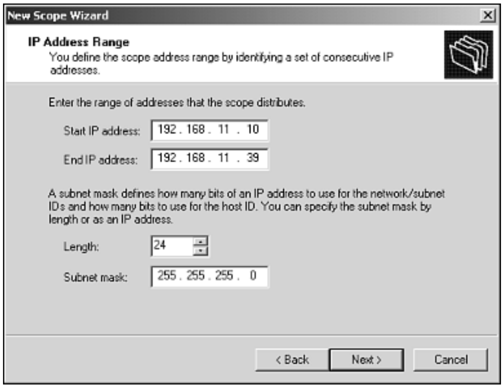

FIGURE 13.6 Defining a scope and available addresses with the DHCP manager.

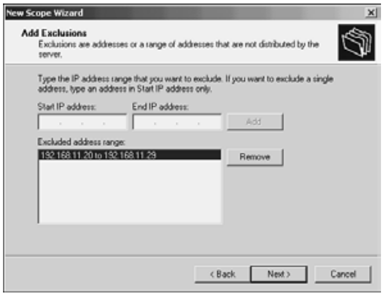

FIGURE 13.7 Excluding addresses from assignment.

A more general configuration of the Microsoft DHCP server in this case is to declare the entire range of addresses in the subnet, 192.168.11.1–192.168.11.254, as available addresses, and then to explicitly add all the addresses that are not available for DHCP to the list of excluded addresses. With this configuration for the scope, you can change the list of available addresses by modifying the list of excluded addresses, without changing the range of available addresses in the scope. The list of excluded addresses in this example is as follows:

192.168.11.1–192.168.11.9

192.168.11.20–192.168.11.29

192.168.11.40–192.168.11.254

Setting Lease Duration

The next dialog box in the New Scope Wizard asks for the duration of leases associated with addresses assigned from the scope (see Figure 13.8). The default value is eight days. The appropriate duration for leases depends on your specific network requirements and design, so you will probably want to review this default duration to see if it is appropriate. Chapter 19, "Tuning a DHCP Service," gives detailed advice on choosing lease durations. In Figure 13.8, the lease duration is set to 90 days, illustrating the configuration for the `genericstartup.com` network discussed in Chapter 3, "Configuring the DHCP Server."

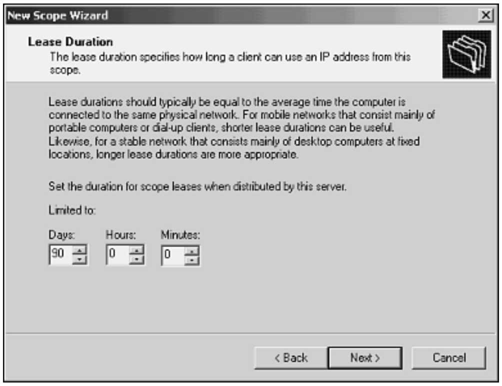

FIGURE 13.8 Selecting lease duration.

Configuring Other Options

The New Scope Wizard allows you to configure other configuration parameters, or options, to be returned to hosts that are assigned addresses in the scope. The first of these options is the list of default gateways (or routers) for the scope, as shown in Figure 13.9.

The next configuration parameters to set are the domain name and a list of DNS servers for hosts in the scope to use. The dialog box you use to set these parameters is illustrated in Figure 13.10.

Finally, the New Scope Wizard prompts you to enter any WINS servers for the scope, as shown in Figure 13.11.

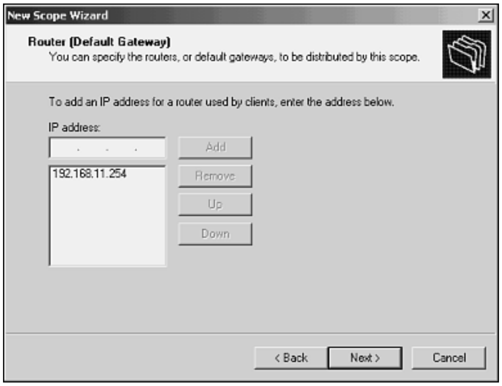

FIGURE 13.9 Selecting default gateways.

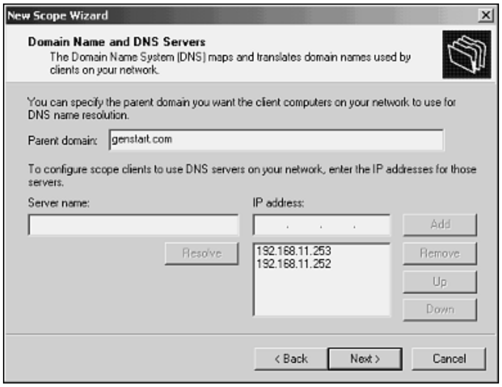

FIGURE 13.10 Setting the domain name and the list of DNS servers.

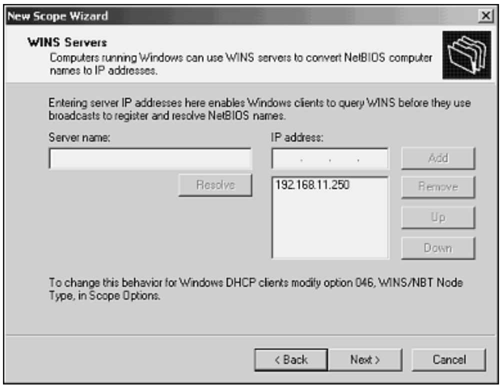

FIGURE 13.11 Identifying WINS servers for the scope.

The final New Scope Wizard dialog box asks if you want to activate the scope (see Figure 13.12). As soon as you activate the scope, the DHCP server begins assigning addresses from that scope, so be sure the scope is completely configured and there are no other DHCP servers assigning the addresses in the scope before you activate it.

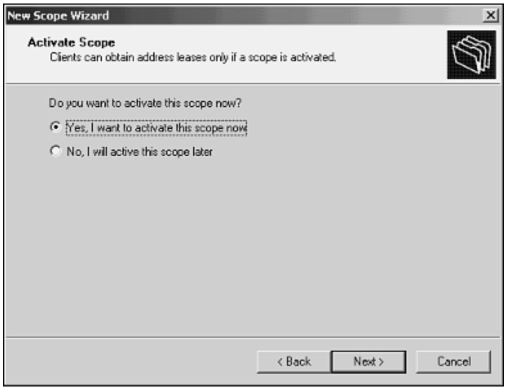

FIGURE 13.12 Activating a new scope.

Configuring Superscopes

The Microsoft Windows 2000 DHCP server includes the ability to define *superscopes*, through which you can associate multiple scopes with a single network segment. Superscopes are equivalent to the `shared-network` declaration in the ISC server configuration file. You use a superscope when you've assigned multiple IP subnets to a single network segment.

The primary function of superscopes is address assignment. The DHCP server groups together all the available addresses from the scopes in a superscope into a single pool of available addresses for the network segment. It finds an unassigned address from one of the scopes in the superscope to assign to a client that needs an address from the superscope.

Client options are managed through the separate scopes in a superscope. The DHCP manager does not include a mechanism for defining options that are common to all scopes in a superscope, nor does it give you a way to examine the list of available addresses across the superscope.

NOTE

The DHCP administrator must make sure that options are defined consistently across the scopes within a superscope. Options such as the `DNS server` or `NTP server` options should usually be defined uniformly across the superscope. Some options, such as the `default routers` option, should be configured differently for each scope.

You manage superscopes through the DHCP manager. To create a superscope, select a DHCP server and then select Action, the New Superscope. The DHCP manager starts the New Superscope Wizard for creating superscopes, as shown in Figure 13.13.

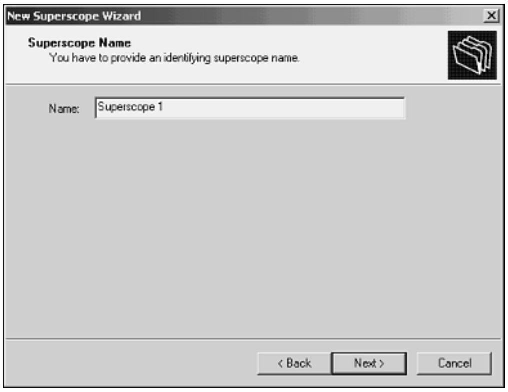

FIGURE 13.13 The New Superscope Wizard in the DHCP manager.

In the first dialog box of the New Superscope Wizard, you should enter the name of the new superscope. The next dialog box, shown in Figure 13.14, displays a list of scopes that you can add to the new superscope. Add the desired scopes to the superscope and click Next.

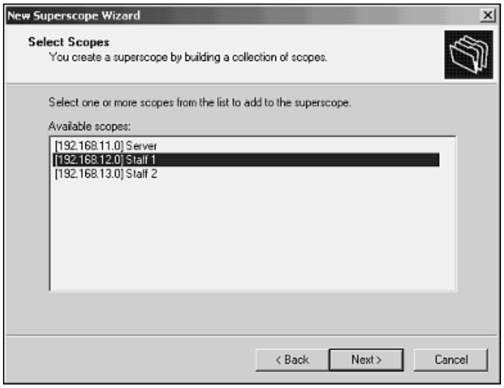

FIGURE 13.14 Adding scopes to a superscope.

Adding a Scope to an Existing Superscope

To add a scope to an existing superscope, first select the scope in the DHCP manager, and then choose Add to Superscope from the Action menu. Then, select the superscope to which the scope should be added and click OK. To remove a scope from a superscope, first select the superscope and then the scope to be removed. Choose Remove from Superscope from the Action menu and click Yes.

Configuring Reservations

You can permanently assign a specific address to a client by establishing a *reservation*, which is equivalent to a host declaration in the configuration file for the ISC DHCP server. A reservation is appropriate for a client that should always use the same IP address, such as a computer that provides a service such as DNS. A reservation is established as part of the scope to which the reserved address belongs. The address assigned through a reservation must be in the range of available addresses for the scope.

You use the DHCP manager to create, change, and delete reservations. To create a reservation, select the scope to which the reservation is to be added, and then select Reservations under that scope. Select Action, New Reservation, and the Server manager displays a dialog box that requests the following information (see Figure 13.15):

Information Requested	Description
`Reservation name`	A name used to identify the reservation by the DHCP manager.
`IP address`	The address to be reserved; it must be an unassigned address within the range of addresses for a scope.
`MAC address`	The Media Access Control (MAC) address (usually the Ethernet hardware address) of the client for which this reservation is defined.
`Description`	Optional text to describe the client and reservation.

You can also select whether this reservation should apply to DHCP requests, BOOTP requests, or both. After filling in the values for the requested fields, click the Add button to add the reservation to the DHCP configuration database.

FIGURE 13.15 Defining a reservation for a DHCP client.

Configuring Options

As with the ISC server, you can configure the Microsoft server with rules that govern the values for options that the server passes to DHCP clients. However, you can configure the Microsoft server with only three types of rules:

- Values for options that are defined to apply to specific clients that are preassigned specific addresses
- Values for options that are defined to apply to clients in a specific scope
- Values that are to be applied to every client the server manages

To select a specific option for configuration, the DHCP manager displays a pull-down menu with a list of all available DHCP options. In this list, the options are identified by both option number and name. Table D.1 in Appendix D, "DHCP Options Summary," lists all the DHCP options and the Microsoft DHCP server's name for each option.

Scope Options

Scope options specify values for options to be returned to any client within the scope for which the option is defined. That is, a client receives the option values from the scope to which the client belongs.

To configure a scope option, select a scope in the DHCP manager and click Scope Options. Select Action, Configure Options. The DHCP manager displays a window in which you can choose an option and the value to be returned to clients in that scope. After you've selected an option for a scope, it appears in the DHCP manager's main window when the scope is selected. Figure 13.16 shows the configuration of the Routers option for a single scope.

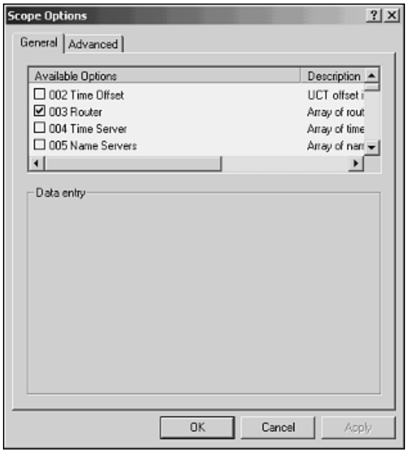

FIGURE 13.16 Configuring the Routers scope option.

Server Options

You can also define *server options*, which are returned to clients from any scope that is managed by the server. To modify the server options, click on Server Options and select Actions, Configure Options. Figure 13.17 shows the configuration of the DNS Servers option as a server option.

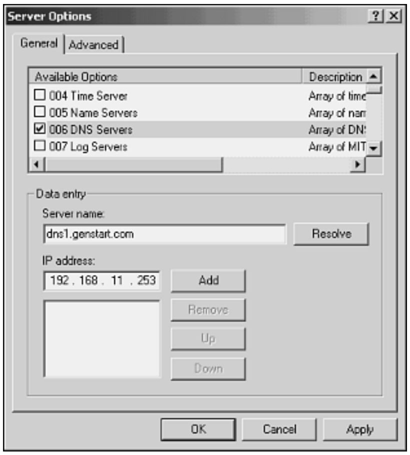

FIGURE 13.17 Configuring the DNS Servers server option.

Client Options

The Microsoft DHCP manager allows you to specify options for a client that has a reserved address. By selecting the specific reservation under Reservations in the scope that contains the reservation, you can add options and values, and you can override scope and server options. Click on the reservation you want to change and then select Action, Configure Options to specify the options that are associated with the reservation.

Vendor-Specific Options

You can configure the Microsoft DHCP server to send vendor-specific options to clients that identify themselves with a vendor-class identifier. When a client includes a `vendor-class-identifier` option in a message to the server, the Microsoft DHCP server matches the value of the option against its list of vendor classes. If the server finds a match, it includes the specified options in a `vendor-specific-information` option in the message the server returns to the client.

The Microsoft DHCP server is configured with four vendor classes:

- The default DHCP Standard Options class, which is used for any client that does not send a vendor-class-identifier option
- The Microsoft Windows 2000 Options class, which is used for Windows 2000 clients
- The Microsoft Windows 98 Options class, which is used for Windows 98 clients
- The Microsoft Option class, which is used for other Microsoft clients

You configure a `vendor-class-identifier` option through the same Scope Options window you use for other options. When you click on the Advanced tab, you should see a window in which you can select the vendor class and option, as shown in Figure 13.18.

You can define a new vendor class and options for that class by selecting a server and choosing Action, Define Vendor Classes. To add a new vendor class, click the Add button and fill in the requested information, as shown in Figure 13.19. The display name is the identifier used in the vendor class list when you're configuring options, and the ID is the value used to match with the `vendor-class-identifier` option sent by the client. You can enter the identifier value in either ASCII or hexadecimal.

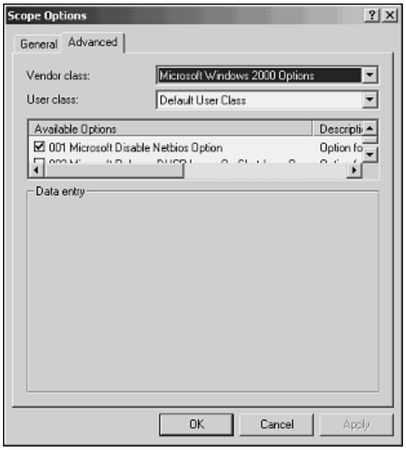

FIGURE 13.18 Configuring vendor-specific options.

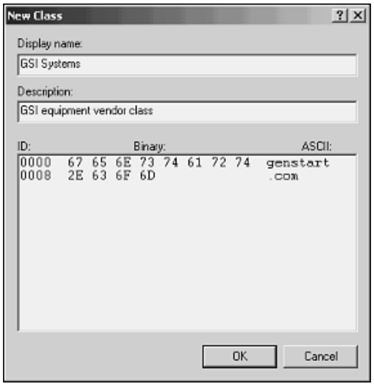

FIGURE 13.19 Adding a new vendor class.

User Class Options

You can also configure the Microsoft DHCP server to send options to a client based on its user class. When you do so, the server examines the contents of the `user-class` option (RFC 3004) and sends to the client any options that have been selected for that class. To configure the server to send options based on the client's user class, you need to configure the server with an option and select the User Class in the Advanced tab of the Scope Options dialog box.

You can define a new user class by selecting Action, Define User Classes and filling in the dialog box that appears, as shown in Figure 13.20.

FIGURE 13.20 Adding a new user class.

Defining New Options

The Microsoft DHCP server is configured with all the options defined in RFC 2132. You can define new options in a server by selecting the server and choosing Action, Set Predefined Options. You need to click the Add button in order to display a dialog box for the new option (see Figure 13.21). The Name field defines the name, and the Code field defines the option code for the new option.

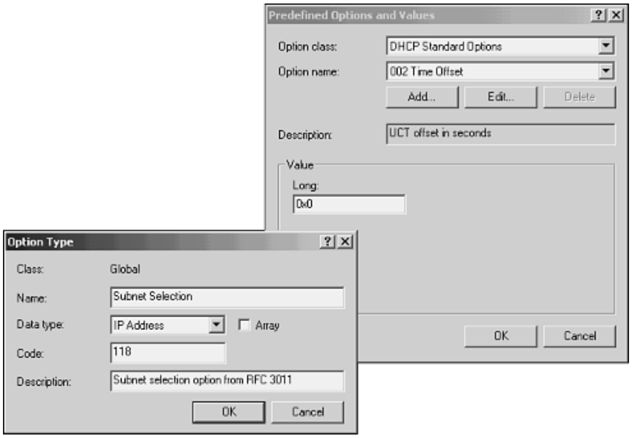

FIGURE 13.21 Defining a new option.

You can use the Data Type drop-down list box to select the type of data that can be carried in the newly defined option. The following data types are allowed:

Data Type	Description
`Byte`	8-bit integer
`Word`	16-bit integer
`Long`	32-bit integer
`Long Integer`	32-bit integer
`IP Address`	32-bit IP address
`String`	String of ASCII characters
`Binary`	Arbitrary binary value
`Encapsulated`	32-bit integer

Selecting the Array checkbox defines the option to carry a list of data values in the way that the Routers option carries a list of IP addresses.

When a new option is defined, it appears in the Available Options dialog box for defining scopes or server options. Selecting the option in this dialog box displays the current value of the option and enables you to change the value. If the option carries a list of data values, you can add, delete, or reorder the list of values through the Data Entry dialog box.

To define a new vendor-specific option for a vendor class, select the option class and then click the Add button. Then, you can enter the information for the new option. New options you define for an option class appear in the Advanced dialog box when you configure new options. Figure 13.22 shows a new option for the Generic Startup vendor class.

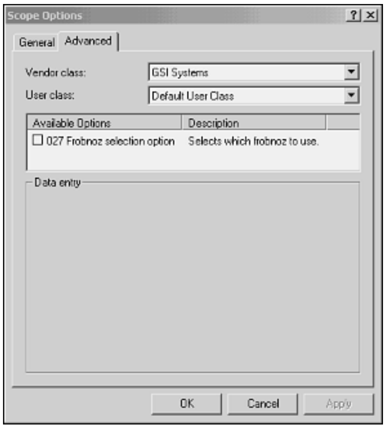

FIGURE 13.22 A new option in the Generic Startup vendor class.

Controlling the Windows DHCP Server

You can control the DHCP service through three different methods: DHCP manager, Services control panel, and Network control panel. Each control function is covered in the next sections of this chapter.

Activating Scope

The DHCP manager can activate or deactivate individual scopes. You should completely configure each scope before activating it. When you first create a new scope, the DHCP manager asks whether it should be activated; click No and configure any options and reserved addresses before activating the scope.

After you configure the scope, highlight the scope in the main server manager window and select Action, Activate.

Starting the DHCP Server Automatically and Manually

You can start and stop the DHCP server and arrange for the server to start automatically when Windows 2000 is started. To control these server functions, click Start and then select Settings, Control Panel. Double-click the Services icon in the Control Panel and then select DHCP Server from the list of services. You can click the Stop or Start button in the menu bar to stop or start the DHCP service manually. Select Action, Properties to display a window through which you can disable or enable automatic initiation of the DHCP service at system startup.

Uninstalling the DHCP Service

To uninstall the DHCP service, click Start and then select Settings, Control Panel. Double-click the Network icon in the Control Panel. Select Services, select Microsoft DHCP Server from the list of services, and click the Delete button. The DHCP server and DHCP server manager are removed from your system.

Summary

The Microsoft Windows 2000 Server distribution includes a DHCP server. The Microsoft DHCP software includes two components: a DHCP server and a DHCP manager. The server runs as an independent process and is responsible for all interactions with DHCP clients. The DHCP manager is a GUI for configuration and management of DHCP servers. You specify the configuration of a network to the DHCP server through the DHCP manager.

Each range of available addresses is defined as a *scope*. The definition for a scope can include option values for that scope; any client assigned an address from that scope also receives the option values that are defined for the scope. Options can also be defined to apply to all scopes managed by a server. The Microsoft DHCP manager can assign values to most of the options defined in Appendix D, and you can define your own options to accommodate newly standardized DHCP options or your own locally defined options.

The Microsoft DHCP manager uses superscopes to define the configuration of network segments that are assigned to more than one IP subnet. Each superscope consists of a set of scopes, which are then assumed to be assigned to a common network segment. The Microsoft DHCP server can choose from the available addresses of any of the scopes in a superscope when assigning a new address to a DHCP client.

The Microsoft DHCP server can differentiate clients based on vendor class or user class. The server can be configured to send specific options and parameter values based on the client's vendor and user class. The server is distributed with several predefined vendor and user classes, and you can also define your own classes and options through the DHCP manager.

14

The ISC DHCP Server

IN THIS CHAPTER

This chapter discusses how the Internet Software Consortium (ISC) DHCP server operates. Specifically, it describes the following:

- How to get the ISC DHCP server
- How to configure the ISC DHCP server
- What you need to do before you can start the ISC DHCP server
- How you can configure the ISC DHCP server to run in a production environment

The complete set of ISC DHCP server configuration commands is documented in Appendix B, "ISC DHCP Server Configuration File Reference." Although this chapter briefly describes what you must do to configure the ISC DHCP server, you should also read the other chapters in this book that describe in detail the theory behind setting up a successful DHCP service on a production network.

Obtaining the ISC DHCP Server

The ISC DHCP server is one of the three components of the ISC DHCP distribution. The other two components are a DHCP client and a DHCP relay agent. When you install the ISC DHCP distribution, you install all three components. Chapter 15, "Configuring a DHCP Server," provides an example of how to use the ISC DHCP relay agent, and Chapter 21, "DHCP Clients," describes in detail how to use the ISC DHCP client.

The ISC DHCP distribution is an open-source product; that is, the source is provided free of charge, and you can modify it as needed. The ISC does not charge to use the software. Because the software is normally provided in source form, that is the form described in this chapter.

The ISC DHCP distribution is available from the ISC Web site, at `www.isc.org/dhcp.html`. You can also find it at `ftp://ftp.isc.org/isc/dhcp`. The DHCP distribution is updated from time to time, and the version number is encoded in the name of the file you download. The format of the filename depends on whether a released version is available or is in alpha or beta testing. Released versions have filenames similar to `dhcp-3.0pl2.tar.gz`, in which `3.0` is the version number, `pl2` is Patchlevel 2, and `.tar.gz` is appended because the distribution is a Unix Tape Archive (tar) file that is compressed with the GNU zip utility (`gzip`).

Beta releases have filenames similar to `dhcp-2.0b1pl27.tar.gz`, in which `2.0` is the version to be used when the software is released from beta, `b1` indicates Beta 1, and `pl27` stands for Patchlevel 27. Alpha releases have filenames similar to `dhcp-3.0.1-20020724`, in which `3.0` is the version number under which the software will eventually be released and `20020724` is the date of the snapshot. The first four digits represent the year, the next two digits represent the month, and the final two digits represent the day of the month.

When a new version of the ISC DHCP distribution is being released, the ISC issues release candidates. A release candidate should not contain any bugs that would prevent a release from happening. When a release candidate is issued, users who want the new features or bug fixes in the candidate try it out to see if it works for them. If bugs are found that the ISC isn't willing to allow into the release (we call these bugs "showstoppers"), the release is delayed until the bugs can be fixed, and then the ISC issues a new release candidate. If no showstopper bugs are found in the release candidate, that release candidate becomes the new release. Release candidate versions consist of the intended release version, followed by rc, followed by a number (for example, 3.0.1rc8). When the release is made, it is released with the same version number, but no rc (in this case, 3.0.1).

Support for the ISC DHCP Server

Support for the ISC DHCP server is provided free of charge (as time allows, of course) on the ISC DHCP server mailing list; information about the mailing list is available at `www.isc.org/dhcp.html`. To get help, just send an email to the mailing list, describing your problem in detail and asking for help with it.

NOTE

Although the e-mail address of the author and maintainers of the ISC DHCP distribution is readily available, he would appreciate it if you would not contact him directly for support, or even to ask about the appropriateness of asking for support. If you need help with the ISC DHCP server, ask for help on the ISC DHCP server mailing list. If you don't get a reply when you ask for help, it's not because the folks at the ISC didn't see your message—it's because they don't have time to answer your question or because it wasn't a good question. If you send mail to the author or maintainers of the ISC DHCP server, they will simply tell you to ask the question again on the mailing list.

Information about an archive of the mailing list and a FAQ are also available at `www.isc.org/dhcp.html`.

Requests for help

When asking for help with the ISC DHCP server, be sure to provide complete information to the ISC. Following is a list of the information to include in your request:

- The name and exact version number of the operating system that is running on the computer on which the DHCP server or client is running (for example, Solaris 2.8).
- The name and exact version number of the operating system that is running on the computer on which the client is running, if you have trouble getting a client working with the server (for example, Windows 98 SE). If you are running on an open-source operating system such as Linux, you need to also specify *which* DHCP client you are running because at least three different DHCP clients run on Linux.
- If you're running Linux, the version of the kernel, C library, and distribution (for example, Linux 2.2.35, libc 5, Red Hat 7.2).
- The specific version of the DHCP distribution you're running (for example, 3.0.1b1pl19, not 3.0).
- A detailed explanation of the problem that is written so that someone who has absolutely no knowledge of your situation will have enough information to understand the problem.
- Your `dhcpd.conf` and `dhcpd.leases` files, if they're not huge (if they are huge, the ISC might need them anyway, but don't send them until you're asked).
- A log of your server or client running until it encounters the problem (for example, if you have trouble getting a client to acquire an address, restart the server with the `-d` flag and then restart the client, and send the resulting output to the support team).

- If the server is dumping core, a stack trace the debugger. For example, if your debugger is `gdb`, perform the following:

```
gdb dhcpd dhcpd.core
(gdb) where
      [...]
(gdb) quit
```

  This assumes that you're debugging the DHCP server and that the core file is in `dhcpd.core`.

- If you can get one, a trace file. Trace files do not work with failover, but they do work without it. A trace file contains a complete replay log of the DHCP server's transactions, and often, the trace file can be used to precisely reproduce a problem. You should play the trace file back yourself before you send it in, to make sure it does reproduce your problem, and if the file is large, ask before sending it. Also, bear in mind that the file contains your complete server configuration, so if there is information in your configuration that you do not want to share with other participants on the ISC mailing lists, you should wait until someone from the ISC asks you for the trace file, and send it just to that person.
- Finally, when you ask for help with the ISC DHCP server, remember that you are asking for a favor from people who may not have time to help you. It is somewhat discouraging to have to say this, but frequently people are very upset when nobody will help them with their ISC DHCP problems. This is free software, and at least at the time of this writing, there is nobody who is being paid to work on it. If you don't get an answer, you have to figure out the problem for yourself. Don't spam the mailing list with repeated copies of the same message, and don't send flaming diatribes about the poor service. If you want good service, you can purchase one of a number of DHCP servers.

Installing the ISC DHCP Distribution

To use the ISC DHCP distribution, you must unpack the distribution, configure it for your system, build it, and install it.

Unpacking the Distribution

To unpack the ISC DHCP distribution, you must connect to a directory on a filesystem that has about 12MB of free space. Then you must use the `gunzip` command to uncompress the tar file and use the `tar` command to extract its contents, as shown in Example 14.1.

Example 14.1

```
% gunzip <dhcp-3.0.1.tar.gz |tar xf -
%
```

The `tar` command extracts the distribution into a directory with the name of the distribution you are building. In the preceding example, this is `dhcp-3.0.1`.

Configuring the Distribution

To configure the distribution, you need to connect to the directory that the `tar` command extracted. Use the `configure` command to configure the distribution, as shown in Example 14.2.

Example 14.2

```
% cd dhcp-3.0.1
% ./configure
System Type: netbsd
%
```

If the `configure` command indicates that it is unfamiliar with the system you are using, you must configure the DHCP distribution yourself or ask for help on the ISC DHCP server mailing list.

Building the Distribution

After you unpack the distribution, you must build it. To build it, simply type **`make`**, as shown in Example 14.3. In this example, many lines of output were deleted and replaced with ellipses to save space. However, the example does show part of the output of the `make` command, to give you an idea of what to expect.

Example 14.3

```
% make
Making all in common
cc -g  -I.. -I../includes -Wall -Wstrict-prototypes -Wno-unused -Wno-implicit -
Wno-comment  -Wno-uninitialized -Werror -Wno-switch -pipe  -c raw.c
[...]
rm -f libdhcp.a
ar cruv libdhcp.a raw.o parse.o nit.o icmp.o dispatch.o conflex.o upf.o bpf.o s
ocket.o  lpf.o dlpi.o packet.o memory.o print.o options.o inet.o convert.o tre
e.o tables.o hash.o alloc.o errwarn.o inet_addr.o dns.o  resolv.o execute.o dis
cover.o auth.o
```

Example 14.3 Continued

```
[...]
ranlib libdhcp.a
Making all in server
cc -g  -I.. -I../includes -Wall -Wstrict-prototypes -Wno-unused -Wno-implicit -
Wno-comment  -Wno-uninitialized -Werror -Wno-switch -pipe  -c dhcpd.c
[...]
cc  -o dhcpd dhcpd.o dhcp.o bootp.o confpars.o db.o class.o failover.o ../commo
n/libdhcp.a
Making all in client
cc -g  -I.. -I../includes -Wall -Wstrict-prototypes -Wno-unused -Wno-implicit -
Wno-comment  -Wno-uninitialized -Werror -Wno-switch -pipe  -c dhclient.c
cc  -o dhclient dhclient.o clparse.o ../common/libdhcp.a
Making all in relay
cc -g  -I.. -I../includes -Wall -Wstrict-prototypes -Wno-unused -Wno-implicit -
Wno-comment  -Wno-uninitialized -Werror -Wno-switch -pipe  -c dhcrelay.c
cc  -o dhcrelay dhcrelay.o ../common/libdhcp.a
%
```

NOTE

The `make` command should complete without errors, but some operating systems might give warning messages. This might occur because of differences between the vendor's C library and the POSIX and ANSI standards, or, more likely, because of differences in areas not covered by the standards. For operating systems that the ISC DHCP distribution currently supports, warnings you might see generally do not represent actual problems about which you should be concerned. If you have concerns, you can raise them on the ISC mailing list. The ISC's goal is for warnings to be eliminated in all compiles on all architectures.

If the `make` command fails with an error, it could be because the version of the DHCP distribution you are using wasn't tested on the architecture on which you are compiling, or it could be because you have a nonstandard compiler setup. Make sure the problem is not with your installation before you ask for help on the mailing list. You should also make sure you are using the most recent distribution of the release cycle you're trying to build. For example, don't ask for help compiling 2.0b1pl27 (a beta test snapshot) when 2.0pl2 (that is, two patchlevels after the end of the beta test) is available. However, if you are sure you are using the most recent distribution and you can't get it to compile, you are encouraged to ask for help.

Installing the Distribution

After you build the distribution, you can install it on your system by using the `make install` command. This installs the entire distribution: the client, the server, and the relay agent, as well as all the documentation. If you want to install only the server, or if you want to install in a special location, you can locate the executables within the build tree and copy them to the correct places yourself. Example 14.4 shows a sample installation.

Example 14.4

```
% su
Password:
# make install
Installing in common
for dir in /usr/share/man/cat5; do  foo="";  for bar in `echo ${dir} |tr / ' '`;
 do  foo=${foo}/$bar;  if [ ! -d $foo ]; then  mkdir $foo;  chmod 755 $foo;  fi
;  done;  done
install -c  dhcp-options.cat5   /usr/share/man/cat5/dhcp-options.0
install -c  dhcp-eval.cat5   /usr/share/man/cat5/dhcp-eval.0
install -c  dhcp-contrib.cat5   /usr/share/man/cat5/dhcp-contrib.0
Installing in server
for dir in /usr/sbin /usr/share/man/cat8 /usr/share/man/cat5 /var/db; do foo="
";  for bar in `echo ${dir} |tr / ' '`; do  foo=${foo}/$bar;  if [ ! -d $foo ];
 then  mkdir $foo;  chmod 755 $foo;  fi;  done;  done
install -c -m 444 dhcpd /usr/sbin
chmod 755 /usr/sbin/dhcpd
install -c  dhcpd.cat8   /usr/share/man/cat8/dhcpd.0
install -c  dhcpd.conf.cat5   /usr/share/man/cat5/dhcpd.conf.0
install -c  dhcpd.leases.cat5   /usr/share/man/cat5/dhcpd.leases.0
Installing in client
nroff -man dhclient.man8 >dhclient.man8
for dir in /sbin /etc /usr/share/man/cat5 /usr/share/man/cat8 /var/db;  do foo
="";  for bar in `echo ${dir} |tr / ' '`; do  foo=${foo}/$bar;  if [ ! -d $foo
]; then  mkdir $foo;  chmod 755 $foo;  fi;  done;  done
install -c -m 444 dhclient /sbin
chmod 755 /sbin/dhclient
if [ xnetbsd = xnone ]; then  echo "No client script available.";  else instal
l -c -m 444 scripts/netbsd /etc/dhclient-script;  chmod 700 /etc/dhclient-scri
pt; fi
install -c  dhclient.cat8   /usr/share/man/cat8/dhclient.0
install -c  dhclient-script.cat8   /usr/share/man/cat8/dhclient-script.0
install -c  dhclient.conf.cat5   /usr/share/man/cat5/dhclient.conf.0
install -c  dhclient.leases.cat5   /usr/share/man/cat5/dhclient.leases.0
```

Example 14.4 Continued

```
Installing in relay
for dir in /usr/sbin /usr/share/man/cat8; do  foo="";  for bar in `echo ${dir}
| tr / ' '`; do  foo=${foo}/$bar;  if [ ! -d $foo ]; then  mkdir $foo; chmod 7
55 $foo;  fi;  done;  done
install -c -m 444 dhcrelay /usr/sbin
chmod 755 /usr/sbin/dhcrelay
install -c  dhcrelay.cat8   /usr/share/man/cat8/dhcrelay.0
# exit
%
```

Vendor-Supplied Versions of the ISC DHCP Distribution

Your operating system vendor might supply a version of the ISC DHCP server in its distribution. It would be nice if you could use this version because then you would not have to go through the steps described previously. Unfortunately, even vendors that do include the ISC DHCP distribution with their operating systems typically do not include the most recent version, for the simple reason that the most recent stable version that was available when they released their operating system distribution has since been superseded by a new version. Before you use a vendor's version of the DHCP server, it's best to go straight to the ISC and see that the version the vendor is shipping doesn't have known bugs. The ISC Web page for any given version of the DHCP distribution includes a complete history of the changes that were made. You can follow that history back to the version you have from your operating system vendor and see whether any significant changes were made. The ISC also provides pointers to RPMs that are available for popular versions of Linux; these RPMs help you easily install the most up-to-date software.

Example 14.5 shows how you can tell what version of the DHCP server you currently have installed. The version number is printed at the end of the first line of output. (If you don't have a configuration file yet, this command prints an error message after it gives you the release information, but you can ignore the error message for now.)

Example 14.5

```
% dhcpd -t
Internet Software Consortium DHCP Server V3.0-alpha 980424
Copyright 1995, 1996, 1997, 1998, 1999 The Internet Software Consortium.
All rights reserved.
%
```

Configuring System Logging for the ISC DHCP Distribution

The DHCP server logs status information and errors by using the *syslog* application programming interface (API). The syslog API provides a way for Unix server programs running as daemons to log events to a centralized event logging daemon, `syslogd`. You configure this daemon through the `/etc/syslog.conf` file. The syslog API enables each information provider to specify a facility code, indicating a rough category into which it falls. The ISC DHCP server normally classifies itself in the daemon category, as do such other daemons as the name server and the Network Time Protocol (NTP) server.

Some sites prefer to keep the DHCP logs separate. If you want to do this, you can change the facility that the DHCP server reports using the `log-facility` statement in the `dhcpd.conf` file. For example, to log on to the `local1` log facility, you would add the following to your `dhcpd.conf` file:

```
log-facility local1;
```

In addition, you would need to add the following to your `/etc/syslog.conf` file:

```
local1.info /var/log/dhcp.log;
```

In this case, `local1` is the log facility, and `info` is the log level. A log level of `info` indicates that the server should log details of each DHCP transaction. This might be more information than you need. To limit logging to only messages that describe errors that have occurred, use the `error` log level instead. The `syslog` daemon does not create a log file for you; you must create the file yourself and then restart `syslogd`. `syslogd` then begins to write log messages to the log file.

Prerequisites to Operation of the ISC DHCP Server

The DHCP server is implemented so that if it detects something inconsistent about its configuration, it refuses to operate. This is because a misconfigured DHCP server can potentially cause more harm to a network than simply having no DHCP server at all. Version 3 of the server requires that you explicitly enable any features that can potentially cause problems. This prevents these features from being enabled accidentally.

The Lease Database

Before invoking the DHCP server, you must create an empty lease database file for it. On a Unix or Unix-like machine, you can do this by typing `touch` ***`file`***, in which ***`file`*** is the full path and name of the lease database file.

The DHCP server stores its lease database as an ASCII text file, much like its configuration file. The lease database file contains the DHCP server's entire knowledge of what it promised to its clients, and thus is *very* important. The DHCP server does not operate if it detects conditions that suggest that by doing so it will lose information from the lease file.

When the DHCP server rewrites the lease database to remove obsolete entries, it does so by creating a temporary file called `dhcpd.leases.`*`pid`* (where *pid* is the process ID of the DHCP server when it creates the new lease file), writing a complete lease database into that file, renaming the old lease file with the backup filename `dhcpd.leases~`, and renaming the new database with the old name.

If the system crashes just as this renaming operation occurs, some of the filesystem directory information might update prior to the crash, but some might not. This can result in there being a new copy of the lease database in `dhcpd.leases.`*`pid`* and an old database in `dhcpd.leases~`, but no lease file called `dhcpd.leases`.

This scenario should not ever happen on a POSIX-compliant Unix system because the DHCP server uses the `rename` system call to perform an atomic operation, renaming the lease file, but because the lease file is so important, the DHCP server doesn't make the assumption that the rename system call is POSIX compliant (it is better to confuse someone who is trying to get the DHCP server working for the first time than it is to accidentally delete the contents of the lease database, even though this is an extremely unlikely occurrence).

To the DHCP server, this condition can't be safely distinguished from the situation in which you have just installed the DHCP server and do not yet have a lease. As a result, the DHCP server *never* creates a lease database file on its own. It always insists that you do it. If you try to start the server without first creating a lease database file, it displays an error message similar to the one shown in Example 14.6.

Example 14.6

```
Can't open lease database /tmp/dhcpd.leases: No such file or directory
Check for failed database rewrite attempt!
Please read the dhcpd.leases man page if you don't know what to do about this.
```

Only leases for clients whose IP addresses have been dynamically allocated are recorded in the `dhcpd.leases` file. A client whose IP address comes from a `fixed-address` declaration will never appear in the lease database. So if your DHCP server does only static allocation, the lease file will always be empty.

The Configuration File

The DHCP configuration file is described in this chapter and previous chapters. You must include the following in the DHCP server's configuration file, or the server will not operate:

- Complete subnet declarations for all network segments on which the server provides service
- IP addresses for all clients for which the server provides service
- Complete subnet declarations for all network segments to which the server computer's network interfaces are connected
- Statements about whether the server is authoritative for any given subnet
- A `ddns-update-style` statement

You must declare every network segment to which clients served by the DHCP server are attached. Even if every client you are serving has a `fixed-address` statement, the server still needs to know the configuration of the network segments: the subnet number of each segment and each segment's subnet mask. If more than one subnet exists on a given network segment, every subnet on that network segment must be explicitly declared—not just the subnet or subnets the DHCP server serves. All subnets on a given network segment should be enclosed within a `shared-network` statement. This is explained in greater detail in Chapter 15.

IP Addresses

You must provide IP addresses for the server to assign for all clients you want to serve. If you are providing static IP address assignments, you must have a `host` declaration for every client on every network segment to which the client might ever be connected. Static assignments are described in detail in Chapter 16, "Client Identification and Fixed-Address Allocation."

If you are providing pools of addresses from which the server dynamically allocates addresses to clients, you must declare these pools by using `pool` declarations. These `pool` declarations must appear within `subnet` or `shared-network` declarations. Each `pool` declaration must contain at least one `range` statement, listing a range of one or more addresses that the server can allocate from the pool. Dynamic allocation is covered in Chapter 15. Address pools are covered in Chapter 20, "Conditional Behavior."

The Network Segment Declaration

DHCP servers are commonly deployed in such a way that the network segment or segments to which they are connected do not require DHCP service or receive DHCP service from other servers. The configurations of these network segments must still be declared; otherwise, the DHCP server can't behave correctly.

If the network to which the DHCP server is connected is a machine room network segment on which no DHCP service is needed, it is sufficient to provide complete declarations for the network segment. If there is only one subnet on this network segment, a `subnet` declaration is sufficient. If there is more than one subnet on this network segment, you must write a `shared-network` declaration that contains empty `subnet` declarations for all the subnets that are configured on the network segment.

Statements of Authority

The server configuration file needs to have a statement of authority for each `subnet` or `shared-network` declaration for which the default authority is incorrect. If the server's authority is the same on all configured network segments, it can be stated in the global scope.

Version 3 of the ISC DHCP server assumes that it does not have complete information about the configurations of the subnets it is serving or about other subnets of which it is aware but on which it is not providing service. This means that if a client is moved from one network segment to another and sends a `DHCPREQUEST` message for an address on its former subnet, the server does not send a `DHCPNAK` message, so the client continues to use the incorrect address until it can no longer renew its lease. DHCP servers deployed in production environments should simply state that they are authoritative in the global scope so that proper `DHCPNAK` behavior occurs. DHCP servers that are authoritative for some network segments and not for others should state that they are authoritative in those scopes.

If the computer on which the DHCP server is running is connected to a network segment that is not controlled by the network administrator who operates the DHCP server, the DHCP server should be configured not to be authoritative for that network segment. To do this, use `not authoritative;` in the `subnet` or `shared-network` declaration for that network segment.

Configuring the ISC DHCP Server

The DHCP server configuration is stored in the `/etc/dhcpd.conf` file. This file is an ASCII text file that contains a sequence of declarations and statements. The syntax is similar to a C or Perl program. A complete reference to the DHCP server's configuration statements is provided in Appendix B. This section introduces the concepts involved in writing configuration files and shows some examples.

The configuration file performs six basic functions:

- It sets parameters that control how the DHCP server behaves.
- It defines options and parameters that are sent to clients.
- It determines how the DHCP server will interact with other network servers and peers (for example DNS servers, OMAPI clients, DHCP failover peers).

- It describes the layout of the network being served.
- It defines ways of sending different parameters to different clients, based on what they send or to what part of the network they are connected.
- It provides IP addresses for DHCP clients.

Server Control Parameters

Server parameters control a number of things:

- The duration of leases the server provides
- The DHCP server's address assignment policies
- Configurable aspects of the DHCP protocol

Example 14.7 shows a typical set of server parameters.

Example 14.7

```
default-lease-time 86400;
max-lease-time 86400;
min-lease-time 300;
use-rfc1048 true;
authoritative;
```

In this example, the default and maximum lease times are set to 86,400 seconds (that is, one day). The minimum lease time is set to 300 seconds (that is, 5 minutes). The server is declared to be authoritative for the networks it serves, and it is instructed to reply with RFC 1048–style options even if it receives a BOOTP request that does not conform to RFC 1048. The complete meaning of all these parameters is described in Appendix B.

Client Options and Parameters

The DHCP server sends *options* to clients. Options are described in greater detail in other chapters, but essentially they contain information about the network to which the client is connected or about network services that are available to the client. The DHCP server can also send three parameters that are not technically options but that are treated in the same way as options: the *`filename`*, *`server-name`*, and *`next-server`* parameters.

Options are specified with the `option` keyword, followed by the option name, followed by the option data. Example 14.8 shows a typical option that might be sent to a client.

Example 14.8

```
option domain-name "example.org";
```

Client parameters are specified in the same way as server parameters, with the name of the parameter followed by its value, as shown in Example 14.9.

Example 14.9

```
filename "netbsd-alpha";
```

All the options defined by the DHCP server are documented in Appendix B.

User-Defined Options

In addition to the predefined options listed in Appendix B, the ISC DHCP server provides the ability to define new options. These include new options standardized after the version of the server you are running was shipped, experimental options, site-local options, and vendor-specific options.

You define a new option by specifying an option name, a code, and a format. Example 14.10 shows an option called `acme-config-file`, with a code of `129`, whose format is a simple string of ASCII text.

Example 14.10

```
option acme-config-file code 129 = text;
```

Options can be more complex than this. You can define an option that encodes a set of values of different types, an array of values, or an array of sets of values. Example 14.11 shows an option definition that encodes one or more IP addresses, an option definition that encodes one or more sets of IP addresses and 16-bit numbers, and an option that encodes an IP address, a 32-bit number, and an ASCII text string.

Example 14.11

```
option acme-font-servers code 130 = array of ip-address;
option acme-registrars code 131 = array of { ip-address, integer 16 };
option acme-license-info code 132  = { ip-address, integer 32, text };
```

NOTE

The DHCP server currently supports only data types defined in RFC 2132. You cannot encode some things that might be desirable; for example, you cannot send more than one text string.

Appendix B gives a complete list of data types that you can use in defining new options.

Option Spaces

In addition to defining new options that are used just like regular DHCP options, you can declare new option spaces. An *option space* is a collection of options. The standard set of DHCP options is numbered from 0 to 255 and all option numbers greater than 128 are reserved for site-local use. If you define a new option space, it has its own numbering system, probably also from 0 to 255, with different options corresponding to each number. For example, the `host-name` option has an option code of 1. If you define a new option space, you can define a different option in that option space that also has an option code of 1. This new option does not replace the `host-name` option because it is in a different option space.

Five predefined option spaces exist: the `dhcp` option space, the `server` option space, the `agent` option space, the `nwip` option space, and the `fqdn` option space. You cannot define new option spaces with these names. The `dhcp` option space contains all the DHCP options that are not encapsulated. The `server` option space represents all the server control parameters—for example, the `authoritative` statement is represented internally as the `authoritative` option in the `server` option space. The `fqdn` option space allows the fields of the `fqdn` option to be examined and modified individually. The `agent` and `nwip` option spaces contain the suboptions for the `relay agent information` option and the `netware/IP` option.

You can define new option spaces for three reasons: to support vendor-specific options, to support site-specific options, and to define a new option in the `dhcp` option space that encapsulates its own set of suboptions, like the `agent` and `nwip` option spaces.

Some DHCP clients use vendor-specific options. These options are encapsulated in the `vendor-specific information` option. This option is handled differently than the other encapsulation options because you might need to configure the DHCP server to handle more than one vendor's set of vendor-specific options. The `vendor-specific information` option is described in Chapter 20.

The second reason to define a new option space is so that overlapping site-specific option codes can be defined. This might be necessary because some vendors of DHCP and BOOTP clients (for example, NCD and Hewlett-Packard) used the site-specific code range (128–254) to define their own vendor-specific option codes before it was reserved for site-specific options and before the `vendor-specific information` option was defined.

The `site-option-space` Parameter

Although some of the option codes for these vendors might overlap, you may want to provide different global values for different vendors. You can do this by defining option spaces and then using the `site-option-space` parameter to determine which option code is actually sent to the client.

Example 14.12 shows a configuration that provides different global values for a site-specific option to two different devices that are identified by using `host` declarations. This example makes practical sense only if many `host` declarations exist, not just two.

Example 14.12

```
option space acme;
option acme.toast-color code 137  = integer 16;
option acme.toast-color 4096;

option space ypl;
option ypl.ot-modulation code 137  = array of integer 32;
option ypl.ot-modulation 10, 188277, 423119, 9188271;

host acme1 {
  hardware ethernet 99:aa:bb:0c:ad:e2;
  site-option-space acme;
}

host ypl1 {
  hardware ethernet aa:ee:ee:1c:a2:29;
  site-option-space ypl;
}
```

The `site-option-space` parameter tells the DHCP server that for all options whose codes are between 128 and 254, the server should use the values declared in the specified option space instead of the values from the default option space.

Configuring Connections to Other Services

An ISC DHCP server can communicate with three other types of entities on the network—DHCP failover peers, DNS servers, and OMAPI clients. Whether the DHCP server interacts with these other services, and how it does so, is configured in the `dhcpd.conf` file.

DHCP Failover Configuration

DHCP failover configuration information specifies which DHCP failover servers, if any, the DHCP server will be peering with, and what pools will be shared in which peering arrangements. The DHCP server can be configured with any number of peering relationships, and any IP address allocation pool can be associated with any failover peering relationship that has been declared.

DNS Update Configuration

The DHCP server can be configured to set up DNS mappings for DHCP clients when it assigns IP addresses to them and to remove those mappings when clients stop using their leases. The information that can be configured includes information about what names to assign to DHCP clients, what DNS servers to contact for various sets of names, and authentication keys to use when updating those DNS servers.

The DHCP server may also use DNS servers to look up hostnames that are specified in the configuration file. This depends on how the particular operating system on which the DHCP server is running performs hostname-to-IP address translation. If this can be configured at all, it must be configured in the operating system's configuration files, not in the DHCP configuration file.

OMAPI Configuration

The DHCP server can be configured to accept connections from OMAPI clients. OMAPI is a protocol that allows a client to examine and modify objects on the DHCP server over a network connection. OMAPI is not enabled by default. To enable OMAPI, you must write an `omapi-port` statement in the DHCP configuration file. OMAPI clients can make significant changes to the DHCP server's internal state, so it is important to authenticate the OMAPI protocol. To do this, you must use the `omapi-key` statement. These statements are described in Appendix B. The `omapi-key` statement requires you to define a key. Keys are used primarily for DNS updates, and the procedure for defining keys is described in Chapter 23, "Configuring Dynamic DNS Updates."

OMAPI is a control protocol for the ISC DHCP server and client. The use of OMAPI in controlling the DHCP server is beyond the scope of this book. For information about how to use OMAPI, consult the documentation that comes in the ISC DHCP distribution. The `dhcpd` manual page documents DHCP server OMAPI objects. The `dhcpctl` and `omapi` manual pages describe how to write C programs that use OMAPI. The easiest way to learn how to use OMAPI is to read the `omshell` man page and try to use the `omshell` program to examine and modify the state of the DHCP server.

Network Configuration Information

Network configuration information describes to the DHCP server the physical layout of the network. The DHCP server actually doesn't need to know how network segments are interconnected; the network infrastructure—routers, bridges, ethernet switches, and so on—takes care of this. It does need to know what subnets are assigned to each network segment and what network segments are present in the network.

Even network segments that don't get DHCP service may need to be described to the server, either so it can behave appropriately in response to requests from those networks or so it can be told not to respond to requests on those networks. The ISC DHCP server is quite insistent about this. If you do not describe the networks to which the DHCP server machine is attached, the ISC DHCP server refuses to operate.

Configuration File Declarations

You must use the `shared-network` and `subnet` declarations to describe the network configuration. The `shared-network` declaration describes a network segment: a single network on which any connected node receives all link-layer broadcasts sent by any other connected node. The `subnet` declaration declares an IP subnet: a single network on which any connected node receives all IP-layer broadcasts sent by any other connected node. You can configure more than one subnet on a network segment; if you do so, you must enclose the `subnet` declarations for those subnets in a `shared-network` declaration. If only one IP subnet is configured on a particular network segment, you don't have to write an explicit `shared-network` declaration.

Chapter 15 describes `subnet` and `shared-network` declarations in detail. Example 14.13 shows an example of a `subnet` declaration for the subnet whose network number is 10.117.22.0 and whose subnet mask is 255.255.255.0.

Example 14.13

```
subnet 10.117.22.0 netmask 255.255.255.0 {
}
```

Scopes

The ISC DHCP server derives the parameters used to control its behavior with respect to a certain client, and the parameters and options it sends the client, from the *scopes* in which the client appears. Scopes do not control the choice of address assigned to a client; this is controlled by the network segment to which the client is connected and the address allocation rules that apply for the addresses that are available on that network segment.

An option that appears in the global scope (that is, outside any declaration with its own scope) applies to all clients. Options in scopes other than the global scope apply only to the clients that appear within those scopes and the scopes outside them. For example, if a client matches a `host` declaration, the client appears in the `host` declaration scope and all the scopes outside it.

Example 14.14 shows a configuration that makes heavy use of scoping. Every declaration in the DHCP server configuration file that has enclosing braces automatically comes with its own scope. In addition, you can explicitly declare a scope by using the `group` statement. Example 14.14 uses the `group` statement to provide the same parameters to a group of `host` declarations for NCD X terminals that otherwise have no common scope.

Example 14.14

```
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;

subnet 10.0.17.0 netmask 255.255.255.0 {
   option broadcast-address 10.0.17.255;
   option routers 10.0.17.254;
   range 10.0.17.10 10.0.17.253;
}

subnet 10.0.16.0 netmask 255.255.255.0 {
   option broadcast-address 10.0.16.255;
   option routers 10.0.16.254;
   range 10.0.16.10 10.0.16.253;
}

# NCD X terminals
group {
   filename "/tftpboot/Xncd19c";
   host ncd1 { hardware ethernet 00:00:a7:09:21:ac; }
   host ncd2 { hardware ethernet 00:00:a7:1c:33:90; }
   host ncd3 { hardware ethernet 00:00:a7:2a:04:17; }
}
```

If a client connected to the 10.0.16.0 subnet sends a `DHCPDISCOVER` message, the DHCP server considers that client to be in the scope of the 10.0.16.0 subnet. It finds the `broadcast-address` and `routers` options in that scope, and then it moves to the scope that is outside the subnet scope—in this case, the global scope—and finds the `domain-name` and `domain-name-servers` options. The explicitly specified options it finds for the client, then, are the `broadcast-address`, `routers`, `domain-name`, and `domain-name-servers` options.

A client can appear in more than one scope at the same time. Figure 14.1 shows a diagram of the scopes declared in Example 14.14. As you can see, each host declaration has a scope. When a client matches a `host` declaration, the DHCP server considers that client to be in the scope of the `host` declaration. It *also* considers the client to be in the scope of the `subnet` declaration for the subnet from which the client is assigned an IP address. If, for example, the client matching the `ncd1 host` declaration is assigned an address on the 10.0.16.0 subnet, the client appears in that subnet's scope and in the `host` declaration scope, as well as in all the scopes containing them.

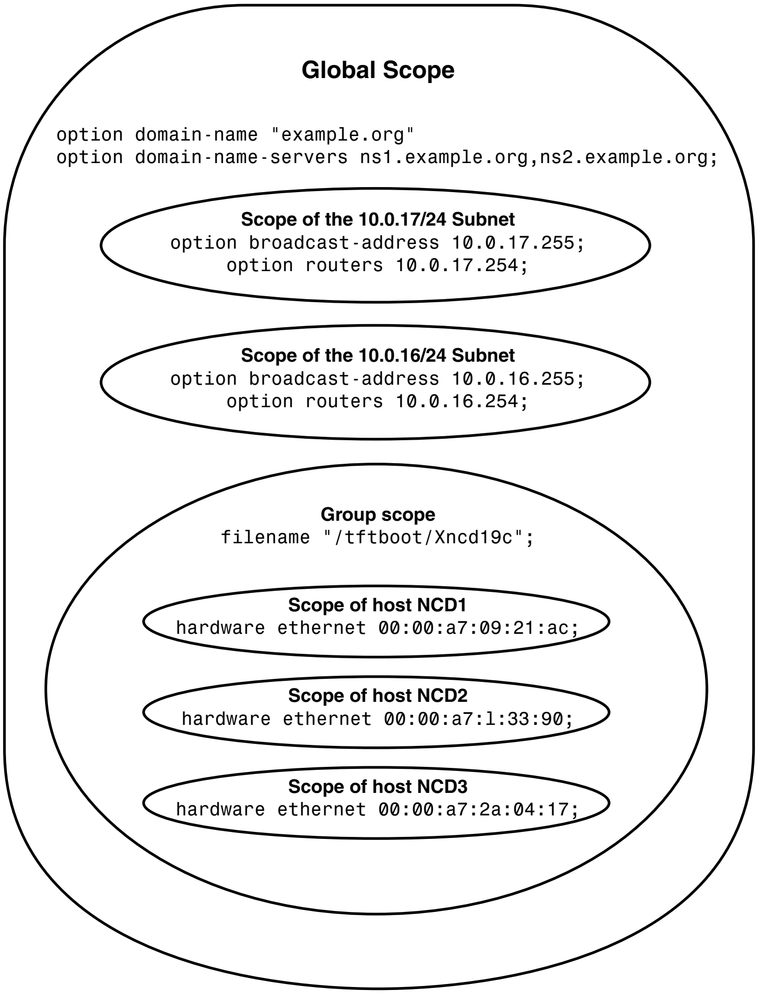

FIGURE 14.1 A diagram of the scopes in Example 14.14.

The DHCP server processes scopes by cycling through each specific scope in which the client appears, in reverse order of specificity—in this case, the subnet scope and then the host scope.

Specific Scopes

A *specific scope* is a scope the client specifically matches, not the scopes that contain such a scope. For example, a client never matches the global scope; it gets parameters from the global scope because the global scope contains some scope that the client does match.

The specificity of a scope that a client matches is dependent on the type of declaration with which the scope is associated. The fewer clients a particular type of declaration can match, the more specific its scope is considered to be. This ordering is somewhat arbitrary; classes are considered more specific than subnets, for example, so a `subnet` declaration that can match only 10 clients can be considered less specific than a `class` declaration that happens to match hundreds of clients throughout the network.

The exact ordering of specificity of declarations starts with the most specific declaration, the `host` declaration, and continues through the subclass, class, subnet, and pool scopes, to the least specific scope, which is the shared-network scope. Scopes that don't match clients (for example, group scopes and the global scope) do not have an inherent specificity. For each specific scope that the server considers, it also considers the scopes outside that scope, all the way to the global scope.

The server never considers the same scope twice. If the same outer scope contains two specific scopes that the client matched, that scope is considered in the process of evaluating the more specific of the two specific scopes and is skipped when the other specific scope is processed.

The scopes considered for the client `ncd1` in Example 14.14 are, in order, the global scope, the scope of subnet 10.0.16.0, the scope for the group declaration that surrounds the `ncd1` host declaration, and the scope for the `ncd1` host declaration.

An additional complication is that different values for the same parameter or option can be specified in different scopes. The scoping rules described previously are intended to ensure that the option or parameter that applies to a particular client is predictable. The result of these rules is that the parameter or option that appears in the most specific scope is always the one that is used. If a parameter within a `host` declaration is different from the same parameter within a `subnet` declaration, the parameter in the `host` declaration is used. If an option declared within a `subnet` declaration is different from the value declared for that option in the global scope, the value in the `subnet` declaration applies to all clients connected to that subnet, and the value in the global scope applies to all other clients (assuming that it isn't overridden in other inner scopes).

IP Address Assignments

You can provide IP addresses for clients in two ways: through static IP address assignments and through *pool* declarations that contain `range` statements.

Static IP address assignments are performed through `host` declarations. Chapter 16 describes static IP address assignment in detail. Briefly, for each client that receives a static assignment, there must be a `host` declaration with information that the server can use to match the declaration to the client, and that declaration must also contain a fixed-address declaration that contains the IP address to assign. When a fixed mapping exists for a client, and the client is connected to the network segment on which that address is valid, the DHCP server will insist on assigning the client the IP address in the `fixed-address` declaration. If the client is connected to a network segment for which it has no valid `fixed-address` declaration, the DHCP server assigns it a dynamically-allocated IP address, or no address at all, depending on how the server is configured.

Addresses are assigned dynamically out of address pools. Address pools are declared with `pool` declarations that contain `range` statements. Chapter 20 describes the `pool` declaration. Chapter 15 describes simple `range` statements.

If you are configuring a pool that will be shared between two DHCP servers in a failover peering relationship, the pool declaration must contain a reference to the failover peer declaration. If this reference is not present, you will experience major problems with your DHCP service.

Invoking the ISC DHCP Server

After you have produced a configuration for a DHCP server, you must know how to invoke it. Invoking the DHCP server involves two aspects: command-line arguments and getting the server to work right for a particular operating system.

Command-Line Arguments

The server takes command-line switches that determine the following:

- The UDP port on which it listens
- Whether it runs in the background as a daemon
- Whether debugging output is logged to `stdout`
- What filename to use for the configuration file, lease database file, and process ID file

- Whether to merely test the configuration
- Whether to be verbose on startup
- Whether to save a trace file
- Whether to play back a trace file
- Whether to check the lease database for correctness
- The interface or interfaces on which the server should listen

You can specify interfaces directly on the command line. You must specify the other parameters by using command-line switches. A switch consists of the `-` character followed by a letter or keyword. Some switches are followed by arguments, in which case the switch and the argument are separated by a space character.

The `-p` switch indicates the UDP port on which the server should listen. If this switch is specified, the server uses the next higher port number as its source port. DHCP specifies that the server should always listen on port 67 and transmit on port 68. If the `-p 67` switch is given (or if no `-p` switch is given), the server uses ports 67 and 68. In general use, it is never appropriate to use the `-p` switch, but it can be very useful for debugging or benchmarking.

The `-f` switch indicates that the server should operate normally but should not fork a subprocess and exit to the invoking process. Normally, if you just type `dhcpd`, the DHCP server prints a startup message and then appears to exit, at which point you receive a shell prompt. However, what actually happens is that the server started a child process that then detaches itself from the terminal and runs in the background, while the parent process (the one you started when you typed the command) exits. The `-f` switch causes the server to remain attached to the terminal and run in the foreground, and it is useful in situations in which you are invoking the DHCP server from `/etc/inittab` (this is a UNIX System V concept and may or may not even be possible on the version of the operating system you are running).

The `-d` switch tells the server to run in the foreground, as with the `-f` switch, but in addition to logging all its output to the syslog daemon, the server logs all its output to the terminal. This can be very useful when you are debugging the server; you invoke it with the `-d` switch the command line and watch as it configures (or fails to configure) a client. If you are reporting a bug on the `dhcp-server` mailing list, you are always asked for the server output, and using the `-d` switch is the easiest way to get it.

The DHCP server normally looks for its configuration file in `/etc/dhcpd.conf`. If you are testing, or simply don't want your configuration file to be stored in `/etc`, you can override this by using the `-cf` switch, followed by the filename that you prefer.

The server normally stores its lease database in a specific directory that varies from system to system. You can use the `-lf` switch to specify a different filename and location for the lease database. All temporary filenames are also based on the specified name and directory.

When the server runs in the background as a daemon, it creates a file into which it saves its process ID. This file is normally stored in the `/var/run` or `/etc` directory, again depending on the operating system. You can use the `-pf` switch to specify a different filename and directory.

The DHCP server does not continue to operate if it finds errors in the configuration file because any errors could result in it badly misconfiguring clients on the network. Therefore, after making changes to the configuration file, it's a very good idea to make sure that it's correct before installing it. You can do this with the `-t` switch. The server tests the specified configuration file and prints error messages if it finds errors. The exit status is zero if no errors exist, and nonzero if errors exist. Thus, shell scripts and human users can use the `-t` switch.

The DHCP server might operate incorrectly if you make changes to the lease database while it is stopped and those changes are not correct. After making changes to the configuration file, you should check that it is correct before installing it. You can do this with the `-T` switch; the server tests the specified configuration file and lease file and prints error messages if it finds errors. The exit status is zero if no errors exist and nonzero if errors exist.

The server normally prints a startup message with the version number, the copyright information, and some information on how to contact the ISC. After you have seen this information once, you probably don't need to see it again (besides, it can make the system startup messages look messy). To prevent this message from printing, you use the `-q` switch.

If you are having a problem with DHCP and want to be able to provide a tracefile so that the problem can be reproduced in a controlled way, you can do this by starting the DHCP server with the `-tf` switch, which takes a filename as an argument. The file must not exist when you start the server. The server logs to the tracefile as long as it is running, and if the error you are trying to find occurs while the server is logging to the tracefile, you have captured the error. In current versions of the ISC DHCP server, tracefile logging does not work properly with failover, although a trace file from a failover server can be useful for debugging problems not related to failover. To play back a tracefile, use the `-play` switch, followed by the name of the file. When you are playing back a tracefile, you must specify the name of a lease file to write to, and this lease file should not be your usual lease file because the lease file will contain the results of the server trace playback, not your current lease database.

Specifying Interfaces

On some operating systems, you can specify on which network interface or interfaces the DHCP server should respond to requests. This works correctly only on operating systems on which the DHCP distribution works with computers that have more than one network interface. If the operating system doesn't support a mechanism by which the server can tell on which interface a packet arrived or determine on which interface one will be sent, it is not possible to specify on which network interfaces it should listen for requests.

The usual reason for specifying interfaces on the command line is that you are installing a DHCP server on a machine that is directly connected to networks in two different administrative domains. For example, in a home office configuration where you have a DSL connection to the Internet, you might set up a Linux or NetBSD system as a router between your ISP's network and your home office network. Both of these networks appear to be broadcast networks, but the DSL network is in your ISP's administrative domain, and your ISP will probably become very upset if you begin providing DHCP service on its network. By specifying on the command line that the interface attached to your home office network, you can prevent the DHCP server from operating on the other network.

Another reason to specify interface names on the command line is that the DHCP server might sometimes incorrectly identify a network interface as a broadcast interface when it is not one. It might then attempt to use this interface, and it might be unable to operate as a result. If this happens, you can explicitly list all your computers' broadcast interfaces on the command line, and the server should ignore the misidentified interface.

Server Operation

After you have installed the server, you will need to do a variety of things. The first task is to make sure the server is started automatically whenever the server computer is started so that DHCP service isn't stopped by system restarts. You might subsequently need to restart the server to install a new configuration, or you might need to modify the lease database.

Starting the Server Automatically

After you configure the DHCP server to your satisfaction, you must arrange for it to start automatically. On Unix and Unix-like operating systems, servers are normally started by the `init` daemon. The `init` daemon starts in one of three ways, depending on the operating system: from an `/etc/inittab` file, from an `/etc/rc` shell script, or from a separate script in the `/etc/rc.d` directory.

You might be able to determine which of these three methods is used on your operating system by reading the man page for the `init` program. If that doesn't help, look for an `/etc/inittab` file; if one exists, that may be how servers are started. There should be a man page for the `/etc/inittab` file. If you invoke the server from `inittab`, make sure to give it the `-f` switch because versions of `init` that use `inittab` automatically start a new instance of a daemon when the old one exits. Without the `-f` flag, the DHCP server always appears to have exited from the perspective of the process that started it.

If no `/etc/inittab` exists, look for an `/etc/rc` file. If such a file exists, it is probably a shell script that invokes various daemons on startup. Most operating systems that provide `/etc/rc` scripts expect you to make local modifications in a local file, called `/etc/rc.local`. Your entry in `/etc/rc.local` should look something like the one in Example 14.15.

Example 14.15

```
if [ -f /etc/dhcpd.conf ]; then
  echo -n ' dhcpd'; dhcpd >/dev/console 2>&1
fi
```

Shell Scripts

If no `/etc/rc` exists, or if `/etc/rc.d` exists in addition to `/etc/rc`, you should add a shell script to the `/etc/rc.d` directory. These scripts operate differently on different operating systems, so you should try to find documentation on how they work on your operating system, or look at an existing script to see how it works. Example 14.16 provides a very simple example of a startup script.

Example 14.16

```
#!/bin/sh

if [ x$1 = xstart ]; then
  dhcpd >/dev/console 2>&1
fi

if [ x$1 = xstop ]; then
  kill `cat /var/run/dhcpd.pid`
fi

exit $?
```

Startup Files

After you add a shell script to /etc/rc.d, you must configure it to run. Systems that use /etc/rc.d generally order their startup with a set of /etc/rc*n*.d directories, in which *n* is a number representing a runlevel. Runlevel numbers aren't standardized. If you don't know your runlevel and your documentation doesn't say, you must look through these directories to find the one where network daemons are started. When you find that directory, you must create a symbolic link from the startup file you just created into this directory.

The system startup script sorts the names of the files in this directory and executes them in the sort order. Therefore, the filename you choose must appear later in the sorted list of files than any scripts that must be executed before the server can start (for example, the script that configures the network). Filenames in these directories usually start with a letter, followed by a two-digit number. You should use the same letter, a number that's higher than the number of any network startup scripts your script depends on (for example, the one that starts the name server), and then the name of the real script file. So, for example, if you name your script dhcpstart, and your rc*n*.d directory for network startup is rc3.d, you might type **ln -s /etc/rc.d/dhcpstart /etc/rc3.d/S57dhcpstart**.

Updating the Server Configuration

When the server is running, you can freely update the configuration file without disturbing it. After you update the file, you need to stop the server by using the Unix kill command. If you are running the server out of /etc/inittab, it should restart automatically. If you are not running the server out of /etc/inittab, you must restart it yourself by invoking it with the same arguments specified in the system startup script.

You do not risk data loss when sending a SIGTERM signal to the server; the server never sends updates to the client for information that isn't confirmed as written to disk by the operating system.

Modifying the Lease Database

If, for some reason, you must make changes to the lease database, you *must* stop the server before any such changes are made; otherwise, data loss is not only possible but also likely. After you make the changes, you can restart the server, and it immediately picks up any changes that you made to the file.

RPM Packages for DHCP

If you are using a distribution of Linux that is based on the Red Hat Package Manager (RPM), you might want to install the RPM packages contributed by Charles R. Anderson, which are available via a link from the ISC Web site. The RPM packages were built on Red Hat Linux, and hence they integrate best with that distribution. There are binaries built specifically for several recent Red Hat Linux distributions on several architectures, as well as a source RPM (SRPM) so that you can build your own binary packages for your particular system. The packages are distributed in subdirectories that reflect the distribution name and version on which they were built.

Five binary RPM packages are provided, and you can install just the parts you need: `dhcp`, `dhcp-client`, `dhcp-server`, `dhcp-relay`, and `dhcp-devel`. All packages require the base `dhcp` package. The others contain the DHCP client, DHCP server, DHCP relay agent, and software development components, respectively. The package files are named after the version of the ISC source from which they were built, plus an RPM package release version and the architecture for which they were built. For example, RPM package release 1cra of version 3.0 of the DHCP server built for the i386 architecture is named `dhcp-server-3.0-1cra.i386.rpm`. The SRPM for this release is named `dhcp-3.0-1cra.src.rpm`.

After you have chosen and downloaded the components you need for your particular distribution and architecture, you need to install them, by using the `rpm` command as follows:

```
rpm -i dhcp*-3.0-1cra.i386.rpm
```

If you are upgrading from a previous RPM package release, you should use the RPM upgrade switch instead of the install switch:

```
rpm -U dhcp*-3.0-1cra.i386.rpm
```

If instead you choose to rebuild the binary packages from the SRPM, the basic command is as follows:

```
rpm --rebuild dhcp-3.0-1cra.src.rpm
```

You have to set up your RPM build environment properly to build the packages; you can do so as a normal user or as the superuser. In the latter case, the binary packages are written to the `/usr/src/redhat/RPMS/architecture` directory and should be installed as explained previously in this section. You can use a couple options to control how the packages are built:

Option	Description
`build_debug`	Builds the software with debugging information included
`build_scd`	Creates an `/etc/dhcpd` directory for the configuration file

The options are specified as arguments to the RPM `--define` switch, with `1` as the value. You can specify multiple `--define` switches to combine the options. For example, to build the packages with debugging information and an `/etc/dhcpd` directory, you would use this:

```
rpm --rebuild --define 'build_scd 1' --define 'build_debug 1' \
dhcp-3.0-1cra.src.rpm
```

After you have installed the RPM packages, you must configure them. Documentation and sample configuration files are included in each package's subdirectory, under the system documentation directory (for example, `/usr/share/doc/dhcp-server-3.0` or `/usr/doc/dhcp-server-3.0`, depending on the distribution). `init` scripts are installed in `/etc/rc.d/init.d`, and they cause the server, client, and/or relay components to start automatically upon bootup.

You can configure the `init` scripts to pass different options to the actual programs by editing `/etc/sysconfig/dhcpd` for the server and `/etc/sysconfig/dhcrelay` for the relay agent. The options for `dhcpd` are shown in Example 14.17, and the options for dhcrelay are shown in Example 14.18.

Example 14.17

```
# Where to read the configuration file from.
CONFIGFILE="/etc/dhcpd.conf"
# Where to store the lease state information.
LEASEFILE="/var/lib/dhcp/dhcpd.leases"
# Define INTERFACES to limit which network interfaces dhcpd listens on.
# The default null value causes dhcpd to listen on all interfaces.
#INTERFACES="eth0"
INTERFACES=""

# Define OPTIONS with any other options to pass to the dhcpd server.
# See dhcpd(8) for available options and syntax.
OPTIONS="-q"
```

Example 14.18

```
# Define SERVERS with a list of one or more DHCP servers where
# DHCP packets are to be relayed to and from.  This is mandatory.
#SERVERS="10.11.12.13 10.9.8.7"
SERVERS=""

# Define OPTIONS with any other options to pass to the dhcrelay server.
# See dhcrelay(8) for available options and syntax.
#OPTIONS="-q -i eth0 -i eth1"
OPTIONS="-q"
```

All the options except for the `dhcrelay SERVERS` option are optional. Examples 14.17 and 14.17a show the values that the init scripts use by default.

Before starting the DHCP server, you must first manually create the lease file as documented earlier:

```
touch /var/lib/dhcp/dhcpd.leases
```

Finally, you can start and stop the server manually with these commands:

```
/sbin/service dhcpd start
/sbin/service dhcpd stop
```

You can restart the server as follows:

```
/sbin/service dhcpd restart
```

The init script for dhcpd automatically tests the configuration file and the lease database before it restarts the server, by invoking dhcpd with the `-t` and `-T` switches. This prevents a currently working server from being restarted if there is an error in the new configuration file.

Summary

The ISC DHCP server is an open-source software product from the ISC. You can download the source code for the server from the ISC Web site. You can also get binaries for Red Hat Linux by using a link through the ISC Web site, and many operating systems include versions of the ISC DHCP server.

Before you can use the DHCP server, you must create a lease database and a configuration file. The configuration file should define IP addresses to assign to clients, set server operating parameters and provide options that are correct for the clients on all network segments on which IP addresses may be assigned.

The server runs continuously as a background daemon. When the configuration file is updated, the server must be restarted for the changes to take effect. If the lease database is to be modified other than by the DHCP server, the DHCP server must be stopped prior to modification and restarted afterward. If the server is being used in production, it should be started automatically when the system on which it is configured to run is powered on.

15

Configuring a DHCP Server

IN THIS CHAPTER

- Configuring a DHCP Server to Be Authoritative
- Configuring an Individual Subnet
- Supporting Multiple Network Segments
- Configuring Multiple IP Subnets on Each Network Segment

This chapter demonstrates how to configure a DHCP server to provide basic DHCP service, using the ISC DHCP server as an example. Configuring a DHCP server involves the following processes:

- Describing the network topology
- Setting up pools of addresses from which the server can dynamically assign addresses to new clients
- Setting up basic services, some of which may vary according to the topology

Chapter 16, "Client Identification and Fixed-Address Allocation," covers static address assignment and client identification.

To work through this chapter, you need an understanding of the basic server configuration file format the ISC DHCP server uses, as described in Chapter 3, "Configuring the DHCP Server." You also need an understanding of IP subnetting, as described in Chapter 4, "Configuring TCP/IP Stacks," and an understanding of DHCP message passing, as described in Chapter 7, "Transmitting DHCP Messages."

Configuring a DHCP Server to Be Authoritative

The first thing you need to do with the ISC DHCP server is to tell it whether it is authoritative. A DHCP server is authoritative if it is administered by the person or persons who run the networks it serves. A DHCP server is not authoritative if it is administered by someone else.

Most DHCP servers are authoritative. The reason that you have to configure the ISC DHCP server to be authoritative is that this makes it unlikely that someone would accidentally configure the ISC DHCP server to be authoritative. A poorly configured DHCP server that is configured as authoritative can wreak havoc on the network to which it is attached.

It is important that you configure every official DHCP server as authoritative. If the DHCP server is not authoritative, it will not tell clients when they have changed networks, so clients that change networks will not get new IP addresses. To configure the ISC DHCP server to be authoritative, you need to write an authoritative statement, as shown in Example 15.1.

Example 15.1

```
authoritative;
```

Configuring an Individual Subnet

Even if a network is small, it always includes at least one subnet. This section uses an example based on Ted Lemon's home network. Ted's network consists of a single Ethernet segment behind a router that connects it to the Internet. There is one subnet on the Ethernet segment, and it has a network number of 10.0.0.0 and a subnet mask of 255.255.255.0. To define this subnet for the server, Ted writes a *subnet declaration,* as shown in Example 15.2.

Example 15.2

```
subnet 10.0.0.0 netmask 255.255.255.0 {
}
```

This declaration simply informs the server that the 10.0.0.0 subnet exists; it does not configure the server to answer requests for addresses on the 10.0.0.0 subnet. With this information, the server has enough knowledge about the network that it can tell when an address that a client requests is invalid. For example, if a DHCP client broadcasts a `DHCPREQUEST` message on Ted's home network, asking for an IP address of 10.20.0.15, which is a different subnet, the DHCP server responds with a `DHCPNAK` message, informing the client that the requested address is not valid.

SUBNET DECLARATIONS

Some DHCP servers do not require written subnet declarations in all cases. These servers derive network topology information by looking at the IP addresses and subnet masks of the interfaces connected to the DHCP server machine and then requiring the network administrator to specify more information for networks to which they are not directly connected.

This might seem like a convenience at first. However, consider what is really happening: It is not that the network administrator has not configured the DHCP server with topology information, but rather that he or she has done so implicitly, rather than explicitly. As the network grows—and the DHCP server configuration file grows with it—the subnet configuration information the network administrator included implicitly when creating the configuration file might change. If the administrator tries to move the DHCP server with such a configuration file to a different subnet, the implicit part of the configuration changes, and the configuration file no longer works..

For this reason, the ISC DHCP server requires that all subnets supported in the configuration file be explicitly declared. The file is, thus, completely portable, and all assumptions about the network topology are visible to other network administrators who might need to adjust the configuration. Even if the DHCP server you are using does not require that all assumptions about network topology be made explicit, you should make the topology explicit if at all possible.

Address Allocation

To allow the server to assign addresses to clients, the network administrator must define which addresses in the subnet are available for dynamic allocation. The subnet in this example contains the addresses 10.0.0.0 through 10.0.0.255—a total of 256 addresses. Of these, 10.0.0.0 and 10.0.0.255 are reserved as broadcast addresses, so 254 IP addresses are available to be allocated.

The first step in setting up a subnet for dynamic allocation is to determine which of these addresses the DHCP server should make available. It is also important to decide whether to reserve some addresses for computers whose addresses must not change. These computers either should not be configured using DHCP or should be configured using static IP address assignments, which are described in Chapter 16. You may also want to reserve some addresses for later use.

Reserved Addresses

On the subnet in this example, two fixed addresses are not allocated by DHCP: the router's address, 10.0.0.1, and the DHCP server's address, 10.0.0.2. The DHCP server does not necessarily need to know about these addresses, but it is important to remember to exclude these addresses from the list of addresses that may be allocated by the server.

The Dynamic Address Pool

If you are confident that no servers with permanently assigned IP addresses will be added to the network, you can allocate all the remaining 252 IP addresses to the DHCP server in a dynamic pool. However, it can be helpful to save some addresses for new servers that might be added later. For example, you can reserve addresses 10.0.0.1 through 10.0.0.9 for servers and give the remaining addresses to the DHCP server to allocate to DHCP clients. To provide addresses to the DHCP server for assignment, you use the `range` statement, as shown in Example 15.3.

Example 15.3

```
range 10.0.0.10 10.0.0.254;
```

Now that the DHCP server has some IP addresses to allocate, it can respond when a client requests an address. When a DHCP client broadcasts a `DHCPDISCOVER` message on the network in this example, the DHCP server chooses an address from the range shown in Example 15.3. It's not really possible to guess which address the server will allocate because it depends on the server implementation.

Client Configuration Information

At this point, the DHCP server in our example has only enough information to assign the client an IP address. As mentioned in Chapter 1, "An Introduction to DHCP," and Chapter 9, "DHCP Options," this is not enough information to enable the client to do anything useful. At a minimum, the client must know the IP address of a router that can serve as its default route and the IP address of at least one name server. This is shown in Example 15.4.

Example 15.4

```
option domain-name-servers ns.fugue.com;
option routers gw.nyc.fugue.com;
```

NOTE

In Example 15.4, the values of the `domain name servers` and `routers` options are declared by using the domain names that correspond to those servers (for example, `ns. fugue.com`). The DHCP server translates these domain names into IP addresses and sends the IP addresses to the DHCP client. Because the network administrator used domain names, there is no need to edit the configuration file when the IP addresses assigned to the routers and servers change.

The only drawback to this is that it takes time to look up a domain name. If a client sends a request and the DHCP server finds that it has to look up one or more names to satisfy that request, it has to send its own request to the DNS server for each name, wait for the responses, and then send its reply to the client.

To take advantage of this feature, it is a good idea to have a DNS server running either on the same machine as your DHCP server or locally on the same network segment that is either a primary or secondary server for all domain names mentioned in the DHCP server configuration file. DNS and DHCP tend to go together, so this is a very common configuration anyway.

Some clients also require the `broadcast address` option, although many compute it themselves from the IP address and subnet mask. Example 15.5 shows how to send this option explicitly.

Example 15.5

```
option broadcast-address 10.0.0.255;
```

If the DHCP network administrator does not specify the `subnet-mask` option, the DHCP server uses the subnet mask value from the subnet declaration. To simplify maintenance, you should not write an explicit `subnet-mask` option declaration. Example 15.6 shows an example of a `subnet-mask` option declaration.

Example 15.6

```
option subnet-mask 255.255.255.0;
```

A Complete Subnet Configuration

In the previous DHCP examples in this chapter we have described the minimal set of options that most DHCP clients need to use a simple network, such as the one at Ted Lemon's home. In practice, you must usually specify more options. Chapter 9 gives a complete list of DHCP options.

Example 15.7 shows the complete DHCP server configuration file, with the option declarations shown in the previous examples.

Example 15.7

```
Authoritative;

subnet 10.0.0.0 netmask 255.255.255.0 {
        range 10.0.0.10 10.0.0.254;
        option domain-name-servers ns.fugue.com;
        option routers gw.nyc.fugue.com;
        option broadcast-address 10.0.0.255;
}
```

Supporting Multiple Network Segments

Many sites require DHCP service on more than one subnet. For example, at the DEC Palo Alto campus, five separate business units were located in four buildings, and each business unit had between one and four network segments of its own.

When servicing more than one network segment, the network administrator must do the following:

- Write a subnet declaration for each network segment
- Define a range of addresses that the DHCP server can allocate on each subnet
- Write option declarations for global options and also for options that vary on different subnets
- Make sure DHCP packets from all network segments can get to the DHCP server

Each additional subnet declaration is written as described in the section "Configuring an Individual Subnet," earlier in this chapter.

There are two ways to configure the DHCP server and the network routers so that the DHCP server receives DHCP broadcasts from each of the network segments that it supports:

- Connect the DHCP server directly to each network segment.
- Relay messages to the DHCP server from network segments to which it is not directly connected, using a relay agent.

Multiple Network Interfaces

One way to support more than one network segment from a single DHCP server is to connect the DHCP server directly to every network segment you want to support. To make this work, the server must be installed in a location where all network segments are available, and a network interface must be installed in the server computer for each network segment. Figure 15.1 shows an example of such a configuration.

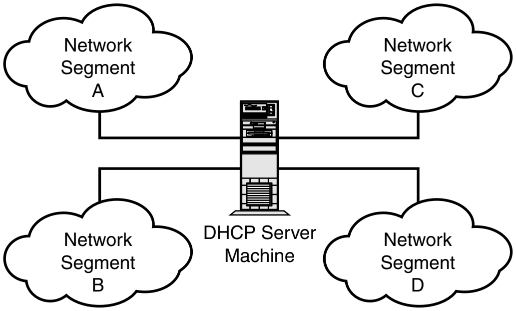

FIGURE 15.1 Connecting the DHCP server directly to all network segments.

Using DHCP Relay Agents

Rather than directly connect the DHCP server to every network segment it serves, most network administrators configure a DHCP relay agent on each network segment. Relay agents are configured with a list of one or more DHCP servers. When a relay agent receives a message from a DHCP client on a particular network segment, it records the IP address of the interface on which it received the request in the `giaddr` field of the message, and then it forwards the message to the DHCP server.

The server uses the `giaddr` field to determine what network segment the client is attached to. When it has a reply for the client, it sends the reply back to the relay agent, using the IP address the relay agent stored in the `giaddr` field. The DHCP relay agent then sends the message back to the DHCP client. Chapter 7 describes this process in greater detail.

Relay Agent Types

There are two common kinds of DHCP relay agents: those that run in IP routers and those that run on general-purpose computers. Relay agents are not routers; they are network server programs that relay packets from clients to servers and from servers to clients.

One possible configuration is to have a server machine on every network segment on which DHCP services will be provided and to have one of the services on that computer be a relay agent that relays DHCP requests back to a central DHCP server. This configuration is illustrated in Figure 15.2. The drawback to this setup is that a server is needed on every network segment.

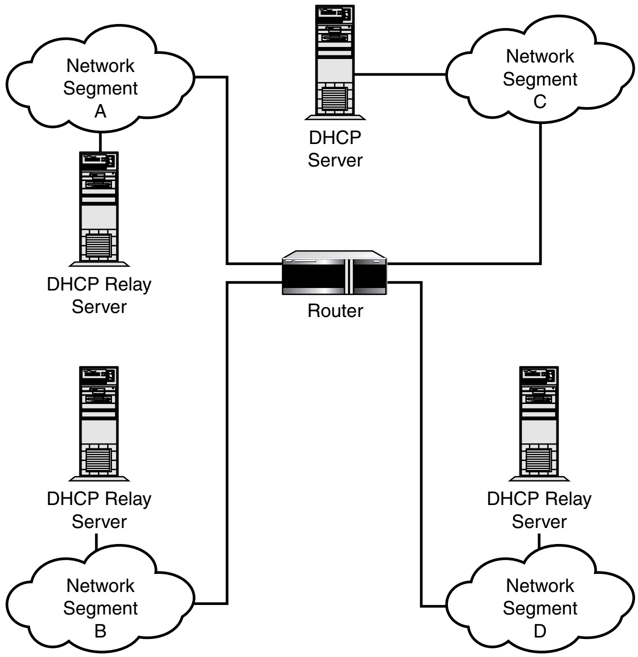

FIGURE 15.2 Using general-purpose computers as relay agents.

If you use dedicated routers to route on the network, and if the router software includes a DHCP relay agent, it is very convenient to use the embedded relay agent. A router must be connected to every network segment anyway, so in this case you do not incur an additional equipment overhead for providing DHCP service. Figure 15.3 shows an example of this sort of configuration.

If a general-purpose computer does the routing, you can likewise run a DHCP relay agent program on each router. It is becoming popular at some cost-sensitive sites to set up inexpensive Linux or NetBSD-based computers that start from a floppy disk or a very small hard drive to act as dedicated routers. Starting up a DHCP relay agent on such a machine takes little additional effort and serves the same purpose as a built-in DHCP relay agent on a dedicated router. Figure 15.4 shows such a configuration.

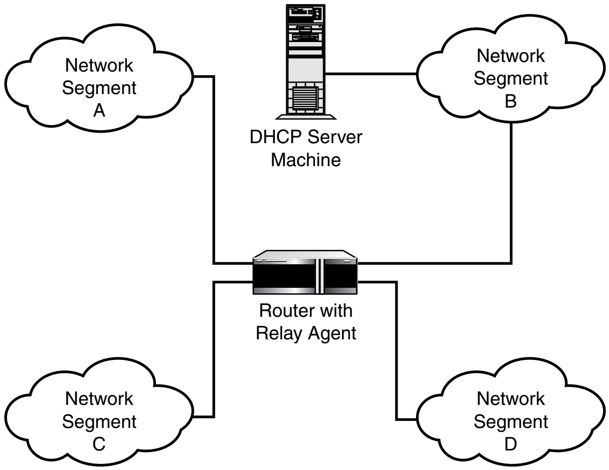

FIGURE 15.3 Using a dedicated router with its own DHCP relay agent.

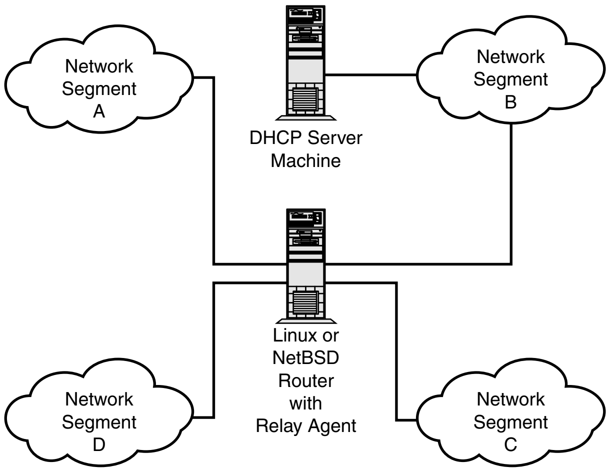

FIGURE 15.4 Using an inexpensive PC running Linux or NetBSD for routing and DHCP relay.

The ISC DHCP Relay Agent

The *ISC DHCP relay agent* is a widely available relay agent that works on general-purpose computers. This program typically runs automatically at system startup. The IP addresses or domain names of the DHCP server or DHCP servers to which the relay agent should forward DHCP messages are specified on the command line as shown in Example 15.8.

Example 15.8

```
dhcrelay dhcp.rc.isc.org
```

The Cisco DHCP Relay Agent

Most dedicated routers contain built-in DHCP relay agents, and each router must be configured differently. It is difficult to document every possible embedded router; for this book, Cisco routers are used as examples.

Cisco routers running Cisco IOS versions 10.0 and higher support DHCP relaying as part of a general mechanism for forwarding UDP broadcasts. Every interface on a Cisco router is declared by using an `interface` declaration. After the interface declaration for the interface connected to each network segment for which the network administrator wants to provide DHCP relaying, one or more `IP helper-address` statements must be written—one for each DHCP server to which DHCP packets will be relayed.

The Cisco `IP helper-address` statement enables relaying of a number of UDP-based protocols by default. These include the Time Server (port 37), TACACS (port 49), DNS (port 53), DHCP/BOOTP (port 67 and 68), TFTP (port 69), NetBIOS Name Service (port 137), and NetBIOS Datagram Service (Port 138) protocols. It is a good idea to disable relaying of these protocols unless you want them relayed. The Cisco `IP forward-protocol` statement controls the relaying of UDP broadcast packets.

Example 15.9 shows how relaying is enabled on two interfaces to a single DHCP server at IP address 10.20.11.7 and how all UDP ports except for the DHCP ports are disabled.

Example 15.9

```
no ip forward-protocol udp 37
no ip forward-protocol udp 49
no ip forward-protocol udp 53
no ip forward-protocol udp 69
no ip forward-protocol udp 137
no ip forward-protocol udp 138
ip forward-protocol udp 67
ip forward-protocol udp 68
interface ethernet 0
    ip address 10.20.11.1 255.255.255.0
interface ethernet 1
    ip address 10.20.12.1 255.255.255.0
    ip helper-address 10.20.11.7
interface ethernet 2
    ip address 10.20.13.1 255.255.255.0
    ip helper-address 10.20.11.7
```

In Example 15.9, three Ethernet interfaces are defined. The first is connected to subnet 10.20.11/24, which is the subnet that the DHCP server is also connected to. No `IP helper-address` statement exists for this interface because the DHCP server does not need relaying for its own subnet. The other two interfaces are connected to subnets 10.20.12/24 and 10.20.13/24, and both of these interface definitions include `IP helper-address` statements that point to the DHCP server. The `IP forward-protocol` statements are outside the interface definitions because they apply globally.

Other Embedded Relay Agents

Most dedicated routers include DHCP relay agents. The documentation that comes with a router should state what is needed to configure the router. If the documentation is not sufficient, you can try to contact the router vendor for help. If the vendor can't help you, you can also contact the support group for the DHCP server. In all likelihood, someone else who is using that server is also using the same router and knows how to make it work. It is beyond the scope of this book to provide a complete list of such support groups; information on obtaining support can be found in the documentation accompanying most routers and DHCP servers.

A DHCP CONFIGURATION FOR THE ISC OFFICE

The ISC office is a modest affair—a small building with perhaps 20 offices, a machine room, and a testing lab hosted by Vixie Enterprises. It has two network segments: the office network and the lab network. The DHCP server runs on a server on the office network, and a small, dedicated router with a built-in relay agent relays DHCP traffic from the lab network.

The `isc.org` domain is actually used only for external servers; all the internal machines are in the `rc.vix.com` domain for the Redwood City, California, offices of Vixie Enterprises. Figure 15.5 is a diagram of the network.

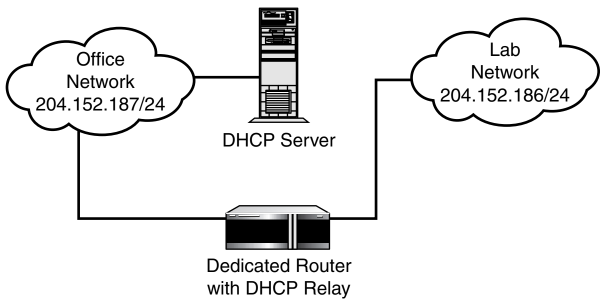

FIGURE 15.5 The ISC office network.

The ISC office has a pair of DNS servers: one running on `bb.rc.vix.com` and one running on `ib.rc.vix.com`. These servers provide name service for the entire office—every DHCP client uses them, regardless of the network segment to which it is connected. The domain name for the entire office is `rc.vix.com`. Network Time Protocol (NTP) is also used quite heavily. The ISC's standard NTP server is `clock.isc.org`.

Because these options are the same, regardless of the network segment to which a DHCP client is connected, they can be defined globally for all network segments. The global options are specified at the beginning of the file, before any subnet declarations, as shown in Example 15.10.

Example 15.10

```
option domain-name-servers bb.rc.vix.com, ib.rc.vix.com;
option domain-name "rc.vix.com";
option ntp-servers clock.isc.org;
```

The office network has a network number of 204.152.187.0 and a subnet mask of 255.255.255.0. The Cisco router, which routes to both the lab network and the Internet, is connected at IP address 204.152.187.254. Example 15.11 shows the subnet declaration for the office network.

Example 15.11

```
subnet 204.152.187.0 netmask 255.255.255.0 {
        range 204.152.187.200 204.152.187.239;
        option routers 204.152.187.254;
        option broadcast-address 204.152.187.255;
}
```

You might notice that for a subnet with 254 possible IP addresses, only 40 addresses are available to the DHCP server for dynamic allocation. This configuration has been specified because there are many computers with fixed addresses on this network, and only a few computers—mostly notebooks and desktop computers—are configured with DHCP.

The lab network has a network number of 204.152.186.0 and a subnet mask of 255.255.255.0. It is routed through the same Cisco router as the office network. The Cisco router's IP address on the lab network is 204.152.186.254. Example 15.12 shows the subnet declaration, which looks very much like the one for the office network.

Example 15.12

```
subnet 204.152.186.0 netmask 255.255.255.0 {
        range 204.152.186.20 204.152.186.30;
        option routers 204.152.186.254;
        option broadcast-address 204.152.186.255;
}
```

The difference in the address ranges of the lab and office networks reflects the fact that the lab network contains many computers with fixed IP addresses running a variety of different operating systems. Unfortunately, none of them provide a DHCP client, and all of them must have permanently assigned IP addresses. It is quite rare for somebody to plug a DHCP-aware computer into the network, but a few addresses are made available just in case.

A Cisco router connects the two networks. The router is configured as a relay agent by using the Cisco-specific `IP helper-address` and `IP forward-protocol` statements. Example 15.13 shows the relevant sections of the Cisco configuration file.

Example 15.13

```
no ip forward-protocol udp 37
no ip forward-protocol udp 49
no ip forward-protocol udp 53
no ip forward-protocol udp 69
no ip forward-protocol udp 137
no ip forward-protocol udp 138
ip forward-protocol udp 67
ip forward-protocol udp 68
interface ethernet 0
    ip address 204.152.187.0 255.255.255.0
interface ethernet 1
    ip address 204.152.186.0 255.255.255.0
    ip helper-address 204.152.187.11
```

Example 15.14 shows the complete DHCP configuration file for the ISC Redwood City office.

Example 15.14

```
option domain-name-servers bb.rc.vix.com, ib.rc.vix.com;
option domain-name "rc.vix.com";
option ntp-servers clock.isc.org;

subnet 204.152.187.0 netmask 255.255.255.0 {
        range 204.152.187.200 204.152.187.239;
        option routers 204.152.187.254;
        option broadcast-address 204.152.187.255;
}

subnet 204.152.186.0 netmask 255.255.255.0 {
        range 204.152.186.20 204.152.186.30;
        option routers 204.152.186.254;
        option broadcast-address 204.152.186.255;
}
```

Configuring Multiple IP Subnets on Each Network Segment

It is very common to use the term *subnet* interchangeably with the term *network segment*. Both terms are defined as any set of network connections such that a computer plugged into any one connection can send packets to a computer plugged into any other connection, without requiring the help of a router.

In fact, subnets are an artifact of the way IP addressing works. In order for IP routing to work, all machines on a given subnet must be on a single network segment. However, it is not required that only one IP subnet be configured on a single network segment.

Some sites take advantage of this as a way of allocating address space; when a network segment is first deployed, a single IP subnet is allocated for it. When most of the IP addresses on that network segment have been consumed, the site administrator allocates a second IP subnet to run on the same network segment.

The ISC DHCP server documentation refers to this kind of configuration as a *shared network* because one network segment is shared by two IP subnets. Another common term for this configuration is an *overlay network*. In the Microsoft DHCP server, a subnet is referred to as a *scope*, and a network segment with more than one subnet is referred to as a *superscope*.

NOTE

Except when referring specifically to Microsoft DHCP server menu options, the word *scope* is never used in this book to refer to subnets because it conflicts with the usual meaning of the word *scope*.

Such configurations exhibit some unexpected behaviors. For example, two computers connected to the same network segment but with IP addresses on different subnets cannot normally send packets to one another without the help of a router. In theory, the two machines should be able to transmit packets directly to each other at the physical network layer, but the IP protocol does not allow it

Address Allocation on Shared Networks

Because the network administrator defines IP subnets and assigns them to network segments, a client's subnet cannot be determined just by its network connection. The client is connected to a network segment on which any number of IP subnets may be configured.

Thus, when a request arrives from a client, the DHCP server must first determine from which network segment the message was sent. If the client is requesting an existing address, the DHCP server can check the requested address to determine whether it is from any of the IP subnets assigned to the client's network segment. If it is, and if the address is available for the client, the server can assign the client that address.

If a client is asking for a new address on its network segment, however, and there is more than one subnet from which to choose, nothing tells the DHCP server which of the subnets on that shared network to use. The ISC DHCP server normally picks the first available IP address from the list of available IP addresses for that network segment. That list could contain addresses for any number of IP subnets. So, the choice of the subnet within a shared network on which the client gets its IP address is essentially arbitrary.

The arbitrary assignment of the subnet can be inconvenient. If clients on different subnets must communicate through routers, it is good to ensure that clients that might often communicate with one another wind up on the same subnet. For instance, suppose that three departments—accounting, sales, and marketing—are using one network segment. If you have allocated three IP subnets to that network segment, you might prefer that all the clients from sales be located on the first subnet, all the clients from marketing be on the second subnet, and all the clients from accounting be on the third subnet. Chapter 20, "Conditional Behavior," discusses ways this can be done.

You can declare a shared network by enclosing a set of `subnet` statements within a `shared-network` statement, as shown in Example 15.15.

Example 15.15

```
shared-network FLOOR1 {
        subnet 204.152.188.0 netmask 255.255.255.240 {
        }
        subnet 10.0.0.0 netmask 255.255.255.0 {
        }
}
```

This declaration defines two IP subnets, 204.152.188.0 and 10.0.0.0, which share a single network segment.

Option Scoping with Shared Networks

Although the process of selecting the subnet on a shared network on which a client will be assigned an address is somewhat arbitrary, after that subnet is selected, the client can be assigned parameters that are specific to the subnet. In many cases, this is required. For example, each subnet must have its own default route and may have a different broadcast address. Options that are not globally defined for all subnets but that are common to all clients on a shared network can be declared within the `shared-network` statement, before the `subnet` statements, as illustrated in Example 15.16.

Example 15.16

```
shared-network FLOOR1 {
        option domain-name-servers 204.152.188.10;

        subnet 204.152.188.0 netmask 255.255.255.240 {
               option routers 204.152.188.14;
        }
        subnet 10.0.0.0 netmask 255.255.255.0 {
               option routers 10.0.0.254;
        }
}
```

Avoiding Routing on a Shared-Network Segment

When a machine on one IP subnet wants to send a packet to a machine that it has determined is on the same subnet, it acquires the link-layer address of the second machine and then sends the packet directly to that address. It does this by using *Address Resolution Protocol* (ARP).

Suppose a machine with address 10.0.0.1 wants to communicate with a machine with address 10.0.0.2. It broadcasts an ARP packet that asks , "Who has IP address 10.0.0.2?" The machine with that IP address replies: "I have 10.0.0.2, and my link-layer address is 08:00:2b:4c:29:3d." The machine at 10.0.0.1 then sends the packet directly to the machine at 10.0.0.2.

When a machine on one IP subnet wants to send packets to a machine on a different IP subnet, it chooses a router from its routing table. It uses ARP to get the link-layer address of that router (routers in a computer's routing tables must always be on a subnet to which that computer is directly connected), and it then sends the packet to the router. That router then sends the packet on toward its final destination. On a shared network, this means the packet is sent on the local network segment to the router, which then sends it back on the same network segment to the destination machine.

Proxy ARP with Microsoft Windows

The Microsoft IP implementation has a special mode, which is not specified by the IP protocol standard, that enables machines on one subnet on a network segment to send packets directly to machines on a different subnet of that same segment. It does this by sending the client's own IP address in the `routers` option. When Windows sees that its default route points at its own IP address, it treats *all* IP addresses as if they are local; it never tries to send IP packets through a router. Example 15.17 shows how to make the ISC DHCP server send the client's IP address for its default route.

Example 15.17

```
use-lease-addr-for-default-route on;
```

In Example 15.17, imagine that you have a Windows computer whose IP address is 204.152.188.10 and whose default route points to its own IP address. If Windows needs to send a packet to a computer at 10.0.0.19, it broadcasts an ARP packet asking for the link-layer address corresponding to the IP address 10.0.0.19. Because 10.0.0.19 is on the same network segment as the Windows computer, Windows receives a response from the computer at 10.0.0.19 and then sends its packet directly to that computer without going through a router.

However, what if that same machine wants to send a packet to another computer at 204.152.186.112? That computer is *not* on the same network segment. When the first computer sends out an ARP request asking for the link-layer address of 204.152.186.112, it receives no reply because ARP broadcasts are never routed off a network segment.

This can be solved by using a feature of many dedicated routers called *proxy ARP*. If a computer on the network to which the router is connected sends an ARP request, asking for the link-layer address of a machine whose IP address is not on the same network segment, the router replies with its own link-layer address. This fools the machine that is using ARP into sending the off-network packet to the router, which then correctly forwards it to its final destination.

PROBLEMS WITH PROXY ARP

Using proxy ARP is not a good way to do IP routing, of course. No standard exists for describing how to accomplish proxy ARP; it is just a collection of ad hoc solutions that various vendors have provided. In a very constricted, homogeneous environment, proxy ARP might work well, but it will fail if machines from vendors that do not support it are installed.

In particular, if the DHCP server behavior shown in Example 15.17 is enabled, clients that do not support proxy ARP will receive an incorrect default route and therefore cannot route packets off their local network segments.

Proxy ARP also generates extra ARP traffic. Every time a machine must communicate with a new machine that is not on the local network segment, it must send another ARP packet. Normally, packets sent off the network segment are forwarded to the same router, whose link-layer address is cached, so it requires only one ARP request to get the link-layer address for all such machines.

Pitfalls of Shared-Network Configurations

DHCP administrators commonly make the mistake of omitting necessary information when they first set up a shared-network configuration. For example, say you have a network segment with two IP subnets and you intend to supply IP addresses for one subnet with one DHCP server and IP addresses for the other subnet with a second DHCP server. You might be tempted to just write a subnet declaration in each server's configuration file for the subnet that server is intended to serve and not mention the other subnet. The problem with this is that DHCP servers are responsible for telling DHCP clients whether their IP address is correct for a given network.

`DHCPNAK` Message Wars

Imagine that a `DHCP` client on the shared network described in the preceding section broadcasts a `DHCPDISCOVER` packet. Both DHCP servers may respond with a `DHCPOFFER` packet, each offering an IP address on a different subnet. The client chooses one and sends out a `DHCPREQUEST` for that IP address. The DHCP server whose address was not chosen sees that the requested address is not on the subnet that it knows is connected to the network segment on which the `DHCPREQUEST` was sent. It therefore sends the client a `DHCPNAK` message. If that `DHCPNAK` message arrives before the other server's `DHCPACK` message, the client does not get its address and must start over. This `DHCPNAK` war can go on indefinitely.

If both servers know about both subnets, however, neither server will send `DHCPNAK` messages in response to `DHCPREQUEST` messages for IP addresses on the subnet being served by the other server. Clients can, therefore, be configured successfully.

If both DHCP servers are being managed by the same administrator, it is possible to see what is happening by examining the server logs; if one or both of the servers are sending `DHCPNAK` messages in response to client requests that are correct, their configurations must be fixed. If one network administrator controls one of the DHCP servers and a second administrator controls the other, a network analyzer could be needed to solve this problem. Chapter 24, "Debugging Problems with DHCP," discusses this in more detail.

Cable Modem Networks

Cable modem networks are broadcast networks, just like Ethernet networks. Most ISPs that provide cable modem service use DHCP to perform address assignment on their networks. Some cable modem networks have extensive filtering so that a subscriber sees only packets meant for his or her home. However, many cable modem networks are just big LANs, where a subscriber can see every packet sent to anybody on his or her network segment, including DHCP packets. If you set up a DHCP server but do not configure it carefully, it is likely that the server will see a `DHCPREQUEST` message sent by a computer belonging to your neighbor down the street, deduce that the machine is asking for an IP address on the wrong network, and thus send it a DHCPNAK message. Your neighbor is then unable to use the Internet.

Private DHCP Server at an Office

You might also be tempted to set up a DHCP server at your office to serve your own set of clients on a subnet setup that just happens to run on the same network segment as your own workstation. Say, for example, that the network drop in your office connects to a network segment with a network number of 10.0.128.0 and a netmask of 255.255.255.0. Now, say you decide to set up a subnet for your own use on the same network segment with a network number you know is not in use—for instance, 10.192.0.0 with a netmask of 255.255.255.240. You configure your DHCP server as illustrated in Example 15.18.

Example 15.18

```
subnet 10.192.0.0 netmask 255.255.255.240 {
        option domain-name-server 10.192.0.1;
        option routers 10.192.0.1;
        range 10.192.0.2 10.192.0.14;
}
```

The minute you turn on the network, your DHCP server starts listening for DHCP messages. DHCP doesn't generate much traffic on a typical network. After a DHCP client has its initial configuration, all further exchanges with the DHCP server are unicast, which means the DHCP server you have installed does not see requests from clients unless they are just starting out on the network. If you set up a server in this way, chances are that the first time it sees a DHCP request from a computer other than your own will be the next morning, when computers that were powered down for the night are powered on again.

CONFIGURATION HAZARDS

Recently, a new ISC DHCP server user set up an unauthorized DHCP server on the computer at his desk at work. Then he went home. When he came back the next day, he found that nobody could use the network, and the system administration staff could not figure out why.

The system administration staff didn't know about the other DHCP server. When they looked at their own server logs, they saw a perfectly normal sequence of events. As each user arrived in the morning, he or she turned on his computer. It broadcast a DHCPREQUEST message, and the DHCP server sent back a DHCPACK message. Unfortunately, what the system administrators did not see in their server's log was the DHCPNAK message coming from the unauthorized DHCP server.

The lesson here is that when you are setting up a server such as this, the configuration in Example 15.19 is preferable.

Example 15.19

```
shared-network MY-OFFICE {

    not authoritative;
    subnet 10.192.0.0 netmask 255.255.255.240 {
      option domain-name-server 10.192.0.1;
      option routers 10.192.0.1;
      range 10.192.0.2 10.192.0.14;
    }

}
```

The unfortunate culprit in this story configured his DHCP server so that it believed it knew the configuration of the network to which it was attached, when in fact it knew only about the private subnet the user wanted to set up. Unfortunately, other DHCP clients were connected to the same network segment and used the officially supported DHCP server to obtain their IP addresses. Whenever these clients sent `DHCPREQUEST` packets, asking for IP addresses on the official subnet, the unauthorized server decided that these `DHCPREQUEST` packets were invalid and responded with `DHCPNAK` packets. Because the unauthorized server was faster than the primary DHCP server for the network, clients could not get IP addresses—they were always stopped halfway through the process.

When you add the `not authoritative` directive to the `shared-network` declaration, as shown in Example 15.19, the DHCP server is told that the `shared-network` declaration is not complete. This prevents the DHCP server from aggressively sending `DHCPNAK` packets when it sees `DHCPREQUEST` packets for IP addresses on networks it does not know about.

In version 3.0 of the ISC DHCP server, the server assumes that it is not authoritative for any network; it must be explicitly told to send `DHCPNAK` messages whenever it sees an address that doesn't belong. Without this behavior, DHCP clients moving from network to network cannot detect that they have moved until their leases expire. When you are setting up an official DHCP server that knows the layout of the network, you should always specify the `authoritative` directive.

One last problem exists with the configuration shown in Example 15.18. It declares a range of addresses for dynamic allocation. Any client attached to the `MY-OFFICE` shared network will be offered an address on the 10.192.0.0 subnet. The solution to this is for the owner of the unauthorized server to explicitly specify to which clients IP addresses may be assigned. Chapter 16 demonstrates this in more detail.

The example in the sidebar titled "Configuration Hazards" demonstrates why, if you are not the network administrator for a given network segment, it is not a good idea for you to set up a DHCP server for yourself on that segment. Even if you set up your server as shown in Example 15.19, you still run the risk that the official DHCP server will send a `NAK` message to *your* DHCP clients when they get addresses from your DHCP server.

Finding Unauthorized DHCP Servers

If you are the network administrator for a network segment, and users start complaining that they are being denied access to the network, you should look out for the situation described in the sidebar titled “Configuration Hazards.” If you look in your server log for a particular user’s machine and you see a completely normal start sequence, but that user’s machine isn’t getting its IP address, it might be time to break out a network analyzer and look for unauthorized or otherwise misbehaving DHCP servers. Chapter 24 discusses this in greater detail.

Summary

To configure a DHCP server, the network administrator must describe to the server all the network segments for which it is to provide DHCP service. The server must also be configured with a set of options to send to clients so that the clients can operate on the network, and it must be provided with IP addresses on each network segment so that it can allocate IP addresses for clients.

If the DHCP server is managing more than one network segment, the network infrastructure must be configured to deliver requests from every network segment to the server. You can either attach the DHCP server directly to every network segment it supports or set up DHCP relay agents on all the network segments to which the DHCP server isn’t directly attached.

You can configure more than one IP subnet on each network segment and you can configure the DHCP server to provide addresses on each of those subnets as well as to provide different parameters for different subnets. If a DHCP server is set up on a network segment but does not have authoritative information about the network configuration, the DHCP server must be configured not to send inappropriate `DHCPNAK` messages.

The next few chapters talk about how to refine and customize this configuration to meet site-specific needs.

16

Client Identification and Fixed-Address Allocation

IN THIS CHAPTER

- Identifying Clients
- Static Allocation
- Mixing Static and Dynamic Allocation
- Automatic Allocation
- Access Control

Chapter 15, "Configuring a DHCP Server," describes how to set up a simple DHCP server configuration in which all addresses are allocated dynamically. This chapter describes how the DHCP server associates a lease with a particular client, and how to

- Assign a client a static IP address, either manually or automatically
- Mix static and dynamic IP address allocation on the same network
- Control access to the DHCP server's resources, based on each client's identification

Identifying Clients

DHCP servers keep track of the association between leases and DHCP clients using identifiers that are unique to each client. This is true even for dynamic allocation, as described in Chapter 14, "The ISC DHCP Server." If you want to control a server's behavior with respect to particular clients, you must know how clients are identified.

A DHCP client can identify itself to a server in two ways:

- The client can use the `dhcp-client-identifier` option to send an arbitrary identification string for itself to the server.
- If the client does not use the `dhcp-client-identifier` option, the server uses the client's link-layer address to identify the client. The DHCP server gets the link-layer address from the `chaddr`, `htype`, and `hlen` fields of the DHCP packet.

Each of these ways of identifying a client has advantages and disadvantages, which are discussed in the remainder of this section.

Using the `dhcp-client-identifier` Option

DHCP specifies the `dhcp-client-identifier` option, which is used to uniquely identify clients. DHCP clients are expected to send this identifier whenever they communicate with the server. DHCP allows a client to send this option when it first contacts a server and requires that if it uses the option once, it must continue to use it thereafter. If a client chooses to send this option, the server can use the data from the option to differentiate between clients.

There are some problems with client identifiers. Although the protocol requires that client identifiers be unique on any given network segment, no mechanism is defined for ensuring that they actually are unique. RFC 2132 recommends that the DHCP client identifier consist of a network hardware type and the client's link-layer address but permits it to have any value. Many DHCP clients allow users to configure the client identifier themselves. If a user chooses an identifier that is already in use by another client, a *collision* (that is, the use of the same identifier for two or more different clients) can occur—and the DHCP server treats both clients as if they are the same. DHCP provides no mechanism for preventing collisions. It is simply up to the user to ensure that collisions do not occur.

PROBLEMS WITH USING THE LINK-LAYER ADDRESS AS A UNIQUE IDENTIFIER

Most DHCP clients follow RFC 2132's recommendation with respect to choosing client identifiers. Choosing a client identifier based on a link-layer address can have some negative consequences, however. Consider a typical laptop computer with an Ethernet interface and an 802.11 wireless Ethernet interface; this is an increasingly common configuration because many laptops come with built-in Ethernet cards and many laptop users want wireless service. When this computer connects to the network using the wireless interface, the client identifier is based on the wireless Ethernet adapter's link-layer address. When the computer connects to the network via the Ethernet interface, the Ethernet interface's identifier is used.

This is a problem if the user of the laptop wants a consistent IP address, and it's also a problem for sites where the DHCP server is configured to update the DNS (see Chapter 11, "DHCP–DNS Interaction"). Both the IP address assignment and the DNS information are based on the client identifier. So when the computer connects to the network, it gets a different IP address for its wireless interface than for its Ethernet interface. If the DHCP client or server is doing DNS updates, the client is not able to have a consistent domain name.

Sites can avoid this problem by configuring their DHCP clients to send a client identifier that is not based on the link-layer address. To do this, the user must manually configure a client identifier. Unfortunately, Microsoft DHCP clients prior to Windows 2000 do not provide this capability. To work around this missing feature, some sites have made nonstandard modifications to their DHCP servers, to use the `host name` option as a client identifier. These modifications are discussed in detail later in this chapter, in the section "Specifying Client Identification in a DHCP Server."

Example 16.1 shows how to specify the client identifier for a particular client, when that client uses its network hardware type and link-layer address to construct the `dhcp-client-identifier` option it sends. Most DHCP clients generate their client identifiers from their link-layer addresses. In Example 16.1, the hardware type is 1, for Ethernet, and the remaining six hexadecimal numbers following the 1 are the Ethernet address.

Example 16.1

```
option dhcp-client-identifier 1:8:0:2b:4c:72:17;
```

Example 16.2 shows how to configure a DHCP server with a client identifier when the client identifier is an ASCII text string. RFC 2132 recommends that clients that send text strings as identifiers place an identifier byte with a value of zero in front of the text string. In Example 16.2, the zero byte is specified with a special escape sequence, `\0`, in the string that defines the identifier. Not all DHCP clients follow this convention, so you might have to try to configure the identifier with and without the leading zero byte.

Example 16.2

```
option dhcp-client-identifier "\0joe's computer";
```

Using the Link-Layer Address as an Identifier

RFC 2131 does not require that the client send a `dhcp-client-identifier` option, and some DHCP clients do not send the `dhcp-client-identifier` option. When the server does not find a `dhcp-client-identifier` option in a request from a client, it uses the client's network hardware type and link-layer address instead.

Example 16.3 shows how you can configure a DHCP server with a client's link-layer address and network hardware type when the client is identified through its link-layer address. For comparison, Example 16.3 illustrates the identification of the same client as in Example 16.1.

Example 16.3

```
hardware ethernet 8:0:2b:4c:72:17;
```

Notice that in Example 16.1, the network hardware type is specified with the number 1, but in Example 16.3, it is specified as the name of the type of network hardware (`ethernet`).

How the DHCP Server Identifies Clients

Because all clients send their link-layer addresses and network hardware types but not all clients send a `dhcp-client-identifier` option, the server must choose one of the forms of identification. There are several schools of thought on how to handle this. The ISC DHCP server simply records both identifiers but uses the client identifier if it is there. The Microsoft DHCP server looks to see if there is a client identifier, and if there is not, it makes one up, using the client's link-layer address.

When a server initially allocates an address to the client, it creates an entry in its database for the client. The entry includes the `dhcp-client-identifier` option, if supplied, and on the ISC server, it uses the link-layer address. If a lease is recorded that has a link-layer address and a client identifier, the ISC server ignores the link-layer address.

Because the DHCP server prefers the client identifier to the link-layer address, a DHCP client can acquire more than one IP address.

This can happen when a user's computer is configured to start up two or more different operating systems, both of which obtain their IP addresses by using a DHCP client. If both DHCP clients send the same identification information, the computer has the same IP address, regardless of the operating system it is running. This can be advantageous in some cases and disadvantageous in others.

Consider a system that is configured so that it can boot two operating sysems, such as Linux and Windows 2000. When the system is running Linux, it is configured to provide some network services, but those services are unavailable when the system runs Windows. Some network services react badly if the server is reachable but refusing connections. To avoid having this happen, you might want to arrange for the Linux DHCP client to get a different IP address than the Windows 2000 DHCP client.

To prevent both DHCP clients from getting the same IP address, you configure either the Linux or Windows 2000 client to send a different client identifier. For example, if the computer uses the ISC DHCP client to get its IP address when it runs Linux, the statement in Example 16.4 configures the Linux client with a different client identifier.

Example 16.4

```
send dhcp-client-identifier "my-linux-box";
```

Specifying Client Identification in a DHCP Server

The preceding section describes how to configure the DHCP clients so that the server will allocate addresses dynamically and assign a different IP address, depending on which operating system is running on the computer. However, if static address allocation is performed, the network administrator must also configure the DHCP server to work with the identification information that the user configured into the DHCP clients.

The Microsoft DHCP client chooses a client identifier that consists of the network hardware type followed by the link-layer address. If the link-layer address is `0:a0:4c:2b:e9:ac` and the network hardware type is `ethernet` (that is, `1`), and if the Linux client is configured to send the text string `my-linux-box` as its client identifier, the two host declarations given in Example 16.5 match the Windows and Linux clients, respectively.

Example 16.5

```
host ardmore-win2k {
  option dhcp-client-identifier 1:a0:4c:2b:e9:ac;
}
host ardmore-linux {
  option dhcp-client-identifier "my-linux-box";
}
```

Problems that occur when DHCP users choose their own client identifiers are discussed in the next section.

NONSTANDARD MEANS OF IDENTIFYING CLIENTS

Administrators at a number of sites have decided that neither the link-layer address nor the client identifier option is a viable way of identifying clients. Instead, they have chosen to implement nonstandard solutions based on the known behavior of the Microsoft DHCP clients.

Users cannot easily configure Microsoft DHCP clients to send a client identifier other than one formed from the link-layer address. Microsoft DHCP clients do send a host name option (that is, `12`) in every request, however, and users can configure this easily, by using the Identification section of the Network Setup dialog box.

To avoid collisions, the host name the user supplies must be unique on networks on which the client is used. As with the client identifier, no protocol exists to ensure that the name chosen is unique—it is necessary to trust the user to arrange for this.

Some sites that use nonstandard client identification use the host name the client supplies in preference to the client identifier. This involves customizing the DHCP server. Several sites have customized the ISC DHCP server in this way.

This does to some extent solve the problem described earlier, in which a laptop has one wired and one wireless network interface. However, it works only with Microsoft DHCP clients; clients that do not send a host name option are identified by their client identifier or possibly not at all. In addition, with nonstandard identification methods, the potential for client name collisions can be a major problem.

Client Identification Name Collisions

As previously discussed, two clients can have the same identification. If the identification is a user-supplied text string (for example, a host name), a collision occurs when two users choose the same name. For example, if *The Lord of the Rings* has a lot of fans at any given site, several users might choose the name `gandalf` for their computers.

Using the client's link-layer address as an identifier mitigates this name collision problem; link-layer addresses are usually guaranteed to be unique, at least to a particular machine, if not to a particular network interface. Even if the link-layer address is used, however, it is possible to run into trouble because you can change the link-layer address on many network adapters. For instance, an Ethernet adapter that was widely used at a particular site had a device driver bug that caused its link-layer address to reset to all zeros. Whenever two cards entered this state at the same time, serious confusion ensued.

When a collision occurs between the identification of two clients, the DHCP server thinks the two different clients are a single client. If one client is connected to the network and sends a `DHCPDISCOVER` message with its identification, `gandalf`, the DHCP server allocates an address (for example, 10.240.17.7) and sends a `DHCPOFFER` message, to which the client responds with a `DHCPREQUEST` message. The server then sends a `DHCPACK` message, and the client begins using its new address.

However, if a different client with the same identification tries to obtain an address—by sending a `DHCPDISCOVER` message with its identification `gandalf`—the DHCP server looks up `gandalf`, finds an entry for the first client that started up with that identifier, and, not knowing any better, sends a `DHCPOFFER` message with the same address, 10.240.17.7. The client accepts the address, binds to it, and begins using it, at which point both clients receive packets intended for each other, and the users of the two computers begin experiencing seemingly random unreliability in their network service.

When two DHCP clients are accidentally configured with the same client identifier, you see one of two symptoms. Either you see the same two clients being assigned the same IP address, or you see that two clients are always associated with abandoned leases or `DHCPDECLINE` messages. If you see these symptoms, look at the lease entries for both DHCP clients. If they show the same client identifier, you need to configure one of the clients so that it has a different client identifier.

Static Allocation

Now that you know how the DHCP server identifies clients and how you can identify clients to it, it is time to discuss static IP address allocation. (The Microsoft DHCP server uses the term *reservation* instead of *static IP address allocation*.)

In static allocation with DHCP, the server administrator maintains a list of known DHCP client identification information and assigns an IP address (or possibly more than one IP address, if the client can roam from network to network) for each client thus identified. Static IP address assignments are not permanent; the server leases the IP address to the client for a limited time, which the network administrator can specify. Chapter 19, "Tuning a DHCP Service," discusses the use of static IP address assignment in detail.

Completely static address allocation is surprisingly prevalent, considering how much work it involves. The most common reason people give when asked why they use static IP address allocation is that the network administrator (or, often, management) at a site wants to control which clients have access to the network; this can seem natural at sites that perform static IP address allocation without the help of a DHCP server. Switching to the DHCP server can save quite a bit of work, even though the protocol's full labor-saving capabilities are still not exploited.

When a client tries to obtain a lease, the DHCP server looks it up in the list of clients the administrator provides. If the server finds an entry for the client, it sees whether any of the IP addresses reserved for the client are appropriate for the network to which the client is connected. If it finds an appropriate address, it assigns it to the client.

If you have configured the DHCP server to perform only static address assignment and the server does not find an appropriate address for the client, it simply does not respond to the client's requests. Exceptions do exist, however; if the client sends a `DHCPREQUEST` message for an address that the server knows is invalid for the network segment to which the client is connected, the server always responds with a `DHCPNAK` message. The server also sends a `DHCPNAK` message if the requested address is known to the DHCP server and is not available for the client. If the server finds an appropriate address for the client, it offers that address to the client.

Mixing Static and Dynamic Allocation

It is common for a site to define fixed IP addresses for its DHCP clients that are not mobile or that are otherwise known to the network administrator and still enable dynamic addressing for clients that are mobile or that are not known to the network administrator. To allow dynamic address assignment for other clients on a subnet with static allocations for some clients, the network administrator simply supplies a range of IP addresses in each subnet declaration for subnets on which IP addresses will be allocated.

Chapter 3, "Configuring the DHCP Server," describes the internal network at Generic Startup, Inc. (GSI). The DHCP server for the network was configured with a pool of available addresses for each subnet. Example 16.6 shows the configuration file for the server subnet, with two static allocation definitions for the DHCP and DNS servers on that subnet. With those definitions in place, the computers on which the DHCP and DNS servers run can use DHCP and be guaranteed to receive a consistent, well-known IP address.

Example 16.6

```
# Server subnet
subnet 192.168.11.0 netmask 255.255.255.0 {
    range 192.168.11.1 192.168.11.251;
    # 192.168.11.252 reserved for DHCP server
    # 192.168.11.253 reserved for DNS server
    # 192.168.11.254 reserved for router interface
    option routers 192.168.11.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.11.253;
    }
host dhcp-server.genericstartup.com {
    fixed-address 192.168.11.252;
    hardware ethernet 00:20:78:11:F9:14;
}
host dns-server.generic-startup.com {
    fixed-address 192.168.11.253;
    hardware ethernet 00:C0:6D:16:68:A2;
}
```

When a client broadcasts a `DHCPDISCOVER` packet on the server subnet, the DHCP server there first tries to match the client identification information to one of the `host` entries defined in the configuration file in Example 16.6. If the client's identification matches one of these entries, the DHCP server uses that entry to determine what address to send to the client.

If the client's identification does not match one of these entries, the DHCP server allocates an IP address from the list of available leases on the network and offers that IP address to the client.

Moving a Client from Dynamic to Static Address Allocation

Some sites may want to register clients by enabling them to obtain dynamic leases, extracting the client identification information from the lease database, and creating a static address allocation for each client. This chapter uses a simple case to describe how to convert a client from dynamic to static allocation by simply hand-editing the configuration file.

The ISC DHCP server stores leases in an unstructured text file that provides a history of all lease assignments during server operation.

Each lease is stored as a `lease` statement, and the last `lease` statement in the file for any given IP address is the most current lease for that IP address. Consider what happens when an employee of GSI buys a new computer and plugs it in to the network for the first time. When the computer receives an address, the lease is logged in the `dhcpd.leases` file. (Chapter 14 explains how to find out where the ISC DHCP server stores the `dhcpd.leases` file.) Example 16.7 shows what the client's `lease` statement might look like.

Example 16.7

```
lease 192.168.11.11 {
  starts 3 1999/03/10 00:34:38;
  ends 3 1999/03/10 00:40:38;
  hardware ethernet 08:00:2b:81:65:56;
  uid "ntp-server";
}
```

In a `lease` statement, the `dhcp-client-identifier` option is stored by using the `uid` keyword. In Example 16.7, the DHCP client sends a client identifier option that is not derived from the client's link-layer address. The network administrator wants to turn this `lease` statement into a fixed allocation, so she writes a host declaration such as the one in Example 16.8.

Example 16.8

```
host ntp-server.genericstartup.com {
  option dhcp-client-identifier "ntp-server";
  fixed-address 192.168.11.11;
}
```

Notice that the network administrator does not include a `hardware` statement because the `dhcp-client-identifier` option statement supersedes it. The client identifier that the client sent is an ASCII text string, and the network administrator can enter it into the `host` declaration either as a text string or as a list of hexadecimal numbers.

The network administrator could have followed his usual policy of using the link-layer address as a client identifier. In this case, because the `client identifier` option always supersedes the `hardware` statement, she would have to delete the client's lease from the lease database file. Example 16.9 shows what the `host` declaration looks like when it uses the `hardware` statement.

Example 16.9

```
host ntp-server.genericstartup.com {
  hardware ethernet 08:00:2b:81:65:56;
  fixed-address 10.152.204.75;
}
```

After the static entry for a host is set up, the DHCP server must be told about the new information. Normally, you do this by terminating the DHCP server process and restarting it. When the server restarts, it rereads the `dhcpd.conf` file, and the `dhcpd.leases` file and picks up the new `host` declaration. If the `dhcpd.leases` file must be modified, you should stop the server *before* it is modified and not restart it until the modification is complete.

At some point after the DHCP server reads the host declaration for the client, the DHCP client will attempt to renew its lease. Remember that although the server was told about the client's fixed address, the client is still unaware that anything has changed. So it sends a `DHCPREQUEST` message to the server, asking for the renewal of its old dynamically assigned address.

When the ISC DHCP server finds the host declaration for the client, it assumes that the declared IP address is the one that the network administrator wants the client to have. Therefore, it does not renew the client's lease. Indeed, if it can, it sends a `DHCPNAK` message, forcing the DHCP client to immediately try to obtain a new IP address. If it cannot, the client runs to the end of its lease without getting a renewal. After the lease expires, the client automatically obtains the new IP address.

No matter how the client gets to the `INIT` (reinitializing) state, it broadcasts a `DHCPDISCOVER` packet. The server looks for a matching `host` entry, finds it, and sends the fixed IP address in that entry to the client, at which point the client has its permanent address.

Converting a DHCP Server from Static to Dynamic Allocation

The ISC DHCP server assumes that statically allocated IP addresses are allocated permanently, so it doesn't actually record lease entries for such addresses. Instead, it assumes that the static entry will be available the next time the client contacts the server—when it must renew the lease. This can cause problems for sites that decide they want to convert from static to dynamic address allocation.

People report two common problems in performing such a conversion:

- Moving formerly fixed addresses into a dynamic pool
- Getting clients to take dynamic address allocations after their static allocations are deleted

Problems Moving Static Addresses to a Dynamic Pool

When you move statically assigned IP addresses into a dynamic pool, the server does not know when statically assigned leases expire. When you delete the static assignment from the server's configuration file, it forgets that assignment existed. The client, on the other hand, is probably still using the address. Therefore, there is always a period of time when the client thinks it is entitled to use its static IP address but the server is not aware of it.

It is not a problem if the client extends the lease on its address before the server tries to assign it to another client. However, if the server tries to allocate the lease to another client before the old client renews its lease, the server will most likely abandon the lease. At a minimum, this generates an unexpected warning message to the server administrator. In a worst-case scenario, one or both clients might have trouble using the network until the conflict is resolved.

The easiest way to avoid this conflict is to remove the static IP address assignment but not put that address back into the pool for allocation until you are certain that any lease the client is holding has expired. After you are certain that the client lease has expired, you can add the address to the pool of available addresses. You can configure a default maximum lease length into the DHCP server; if you do not do this, the default is one day on the ISC DHCP server and three days on the Microsoft server. After you remove the entry, you can simply wait for that amount of time before reusing that IP address. Of course, you can also contact the owner of the machine and ask him or her to release the lease.

Problems Getting a Client to Accept a New Address

Some older DHCP clients do not accept a different address than the one they are bound to unless the DHCP server tells them that the address they have is no longer valid, by sending them a `DHCPNAK` message. If the static IP address allocation is deleted for a client, the server simply ignores renewal requests from the client until the client goes back into the `INIT` state, at which point the server offers the client a new IP address. The client does not take this new address because it wants the old address and has not received a `DHCPNAK` message. Thus, it continues to send `DHCPDISCOVER` messages indefinitely.

The ISC DHCP server sends a `DHCPNAK` message to a client, trying to get an address that belongs to a different client. To create a situation in which `DHCPNAK` messages are generated, you might want to modify the client identification information on static assignments you are converting so that the identification does not match a real client. When a client whose static assignment is doctored in this way tries to renew its lease, the server sends it a `DHCPNAK` message, at which point it always gives up that IP address and returns to the `INIT` state.

Automatic Allocation

In some cases, you might want the benefits of static address allocation without the costs; that is, you might want clients to have permanent, fixed IP addresses, but you do not want to configure them into the DHCP server. DHCP defines an address allocation strategy that enables automatic allocation of static addresses. RFC 2131 refers to this mode as *automatic allocation*.

When a client first sends a `DHCPDISCOVER` message on a network segment, the server allocates it a lease from an address range, just as in dynamic allocation. However, instead of offering a limited lease, the server offers an unlimited lease. DHCP specifies that a lease interval of 4294967295 (FFFFFFFF) must be treated as having an infinite duration—that is, the lease never expires.

Until the version 3.0 release, the ISC DHCP server did not explicitly support this mode. However, in version 3.0, the `infinite` keyword was added, allowing leases to be specified as unlimited. The DHCP server does not expire an unlimited lease, which means that an unlimited lease has the same effect as a static IP address allocation.

Access Control

In addition to identifying DHCP clients and assigning fixed addresses to them, you might want to identify clients for other reasons. For instance, some sites want to control access to leases that the DHCP server supplies. Even though these sites support dynamic IP address allocation, they do not want to allocate IP addresses to clients that the network administrators do not know. Some sites might want to group clients in some way—for example, by allocating IP addresses for known clients on a particular network segment on one subnet and allocating IP addresses for unknown clients on the same network segment from a different subnet. Chapter 20, “Conditional Behavior,” explains this second scenario in detail.

If you want to set up a DHCP server that provides IP addresses only to clients it knows, you need to write `host` declarations for all known hosts, but do not specify fixed addresses for those hosts. Then configure the DHCP server not to provide IP addresses to unknown clients. Example 16.10 shows a simple server configuration file that limits access to the DHCP server to clients for which host entries exist, and it allocates addresses dynamically. This works whether the client stays in the same location or moves from one network segment to another.

Example 16.10

```
option domain-name "acl.example.com";
subnet 10.227.94.0 netmask 255.255.255.0 {
  pool {
       deny unknown clients;
```

Example 16.10 Continued

```
    range 10.227.94.2 10.227.94.253;
  }
  option routers 10.227.94.254;
  option domain-name-servers 10.227.94.1;
  option broadcast-address 10.227.94.255;
}
host blaznorf {
  hardware ethernet 00:2b:5c:e9:ad:11;
}
host gzarond {
  hardware ethernet 00:e9:ac:22:08:ee;
}
```

In Example 16.10, the `deny unknown clients` statement within the `pool` declaration prevents the DHCP server from allocating addresses from that pool to clients it does not recognize. The `pool` declaration provides a set of addresses to allocate and the subnet declaration provides some options to send. The two `host` declarations define the two DHCP clients with which the server is willing to communicate. Most DHCP server configurations of this type define more than two hosts, of course.

The ISC DHCP server also enables you to deny access to specific clients. You may want to provide general access to clients without a registration process, but you might not want to assign addresses to some specific DHCP clients. Consider the GSI internal network, which we discussed in the section "Static Allocation." Suppose there is a networked printer connected to the server subnet that the network administrator does not want to assign an IP address; she wants it to be accessible only through AppleTalk (EtherTalk). She can set up a host declaration for that printer (as shown in Example 16.11), and it will not be assigned an IP address.

Example 16.11

```
host laserwriter {
  option dhcp-client-identifier "treekiller";
  ignore booting;
}
```

Unfortunately, using DHCP, you cannot get the printer to stop requesting an address. However, you can prevent the server from giving it one. Example 16.11 uses the `ignore` statement. This prevents the DHCP server from recording its refusal to provide an address to the client in the system log. If a client is requesting addresses frequently, this can be very helpful. If you want to see a log message when the client is refused, you can use the `deny` statement instead of the `ignore` statement.

NOTE

Access control is not synonymous with *authentication*. As mentioned previously, some DHCP clients can be reconfigured with different client identifiers. It is also possible with almost any network adapter to supply a link-layer address other than the one assigned to that adapter. This means you can set up access control based on the DHCP client identifier or the client's link-layer address, but you still must trust that the client is telling the truth—and a malicious client can easily fool the DHCP server.

The DHCP authentication mechanism, described in Chapter 7, "Transmitting DHCP Messages," provides a way in which you can actually authenticate the client's identification. By using this mechanism, you can ensure that malicious clients do not masquerade as legitimate clients. However, even with authentication and access control, you still cannot prevent unauthorized users from simply choosing an address they know is not in use at the moment and using it.

Summary

The DHCP server must uniquely identify DHCP clients, and it does so by using the `dhcp-client-identifier` option, the client's link-layer address, or a nonstandard mechanism. If the mechanism does not guarantee the uniqueness of the identifier, conflicts can occur. DHCP does not provide a way to deal with these conflicts; it simply mandates that clients' identifications must be unique.

Static address allocation allows you to configure DHCP servers with user-supplied mappings between IP addresses and client identification. You can also configure servers to automatically make permanent assignments between IP addresses and client identification, using automatic address allocation. You can also configure servers to perform a combination of static and dynamic address allocation.

In addition to using the client's identification as a key to its IP address, you can also configure DHCP servers to enable or deny access to leases by using the client's identification; however, without an authentication mechanism, this method is not considered a reliable way to control access to a network.

17

Setting Up a Reliable DHCP Service

IN THIS CHAPTER

- Determining Your Level of DHCP Service Reliability
- Specific Failures in DHCP Service
- Improving Reliability by Using Long Leases
- Setting Up a Secondary DHCP Server
- Problems with Setting Up Redundant Servers

As discussed in Chapter 1, "An Introduction to DHCP," loss of DHCP service can be a major problem if you depend on such service for automated management of IP addresses and computer configurations. When DHCP service is unavailable, new computers and computers that move to new network segments may be unable to use network services. Applications running on computers that depend on network service are also disrupted.

This chapter discusses specific ways in which the DHCP service might fail and presents solutions for those failure modes. It also describes some general DHCP service implementations that provide additional reliability through redundant DHCP servers.

Determining Your Level of DHCP Service Reliability

Before you decide on an implementation strategy for providing reliable DHCP service, it is appropriate to review how a loss of DHCP service will affect you and to determine the appropriate level of reliability for your organization and network infrastructure. The loss of DHCP service has two major effects: DHCP clients are unable to obtain new addresses, and they are unable to extend leases on addresses that were previously assigned through DHCP.

The Effects of Loss of Service

In many circumstances, the loss of DHCP service does not immediately affect most DHCP clients and network users and, in fact, may be less disruptive in the short run than the loss of DNS or network file services. DNS service is used with every new connection that requires resolution of a DNS name, so loss of DNS service is immediately obvious to users. On the other hand, a DHCP client that was assigned an address will continue to function normally and won't attempt to contact the DHCP server until half the duration of its lease expires. Even after it begins to request an extension on its lease, the DHCP client will continue to use its address. Only if the DHCP service is still unavailable when the lease actually expires must the DHCP client stop using its address and terminate network connections.

DHCP clients that restart while DHCP service is unavailable simply continue to use their previously assigned addresses until their leases expire. As long as a computer isn't moved to a new network segment, the computer can use its old address. The loss of DHCP service has an effect only when a lease actually expires.

Of course, clients that require DHCP service cannot access the network until such service is restored. If you administer a network to which laptop computers are frequently connected and disconnected, or if clients that you support request a short lease duration, many of your clients may be quickly affected by the loss of DHCP service. And, unfortunately, this loss of service may cause DHCP clients to fail in ways that your end users might not understand (and might not be patient about!). Different DHCP clients react in different ways when they cannot contact a DHCP server. In many cases, the user experiences long startup times while the DHCP client attempts to contact the DHCP server, along with unexplained loss of network access.

NOTE

Recent DHCP clients from Microsoft and Apple have an additional feature that may cause confusion when DHCP service is unavailable. If these clients cannot contact a DHCP server, they choose an address from a range of addresses that are reserved for autoconfiguration. Unfortunately, a computer that performs this autoconfiguration appears to be operating normally, but in effect it is using an IP address that cannot be used to reach destination computers that are not connected to its local network segment. The user has no indication of network initialization failure, but it cannot access network services. A DHCP client that performs autoconfiguration assigns itself an IP address on the 169.254.0.0/16 network. This mechanism for autoconfiguration is currently documented as the Internet Draft "Automatically Choosing an IP Address in an Ad-Hoc IPv4 Network," and it is available as `draft-ietf-dhc-ipv4-autoconfig-04.txt`.

You must determine your own requirements for the reliability of your DHCP service based on the ways in which your clients access your network, the length of the leases you choose, and your tolerance for calls to your help desk. Although no single solution exists that fits the needs of every network, the next section covers some specific failure modes and suggests some solutions for those scenarios.

Specific Failures in DHCP Service

DHCP service can fail for quite a few reasons. The most obvious is that the computer on which the DHCP server is running fails. Another obvious reason is that network connectivity may be lost between the DHCP client and the DHCP server. Less obvious are failures of relay agents and IP address starvation caused by buggy clients, misconfigured servers, and denial-of-service attacks. Chapter 24, "Debugging Problems with DHCP," and the section "Authenticated DHCP Messages" in Chapter 7, "Transmitting DHCP Messages," discuss debugging incorrectly configured servers and denial-of-service attacks.

Server Failures

The most common server failure is simply a power outage. This can result from a power loss on the electrical circuit to which the server computer is connected or an accident that causes the computer to power off or causes a sitewide power loss. Server hardware failures can also cause interruption (or, in some cases, complete loss) of service.

Limited Power Failure

In the event of a power failure that is limited to the server machine and a small number of other machines, few of which are DHCP clients, there shouldn't be much of a problem. DHCP requires that the server record leases on some kind of stable storage (for example, a hard-disk drive) before confirming them. As long as the DHCP server vendor takes this requirement seriously, the server can simply be restarted.

The easiest way to mitigate this problem is to make sure that the duration of the power failure is short and to fix the problem quickly. The longer the server is powered off, the more likely it is that a DHCP client without a valid lease will attempt to acquire one. Such clients must wait until DHCP service is restored before they can use the network. Because of the way lease renewal works, clients with existing leases usually have at lease one-half of the lease duration left at any given time, so you can safely take a little less than one-half of the lease duration you assign to restore service.

Major Power Outages

When power goes common off throughout a site, the recovery process is a bit different than it is for failures in which only the DHCP server is affected. The problem in this scenario is that all the DHCP clients powered off at the same time and will likely power back on at the same time. If the DHCP server is not available when the clients' power is restored, the clients may not get addresses and will continue to retry. This creates the following three problems:

- Until the clients are assigned addresses, they can't use the network.
- While the clients try to obtain addresses, they might create a significant amount of broadcast traffic on the network.
- All the DHCP messages from the clients are directed at the DHCP server. When it comes back online, the server might experience an even higher load than if it had been available when the first clients started requesting addresses.

For these reasons, it is recommended that you arrange for your DHCP server to be available before the clients start requesting service. The easiest way to do this is to power the computer that hosts your DHCP server by an *uninterruptable power supply* (UPS). This is a lovely misnomer for a battery-backed-up power supply that, although not actually uninterruptable, can be purchased in configurations that enable your server computer to run for a reasonable period of time during a power outage. If you can afford it, you might choose to provide backup power for your network equipment, using a generator. If you have limited funds, however, you might find it useful to run the DHCP server on a computer with very low power consumption, to reduce the cost of the UPS you will need to keep it running for whatever duration you plan.

NOTE

Laptops come with their own built-in UPSs. Although laptops are usually thought to be inappropriate for network services because they are often limited in processing and storage capacity, they may be suitable for supplying DHCP service. In many installations, the number of DHCP requests and the computing power required to process them are low enough that an inexpensive, obsolete laptop is sufficient. If you leave the laptop plugged in all the time, you'll get a few hours of continuous service in the event of a power failure. If you power the computer through UPS, it will take a very long time before the UPS's battery dies.

Another solution is to put your DHCP server on a computer that starts up more quickly than your DHCP clients. If you run DHCP on a machine that is running Unix, BSD, or Linux, you might be able to arrange to start the DHCP server very early in the startup process. If you do this, you must arrange for the disk partition on which the server stores its files to be cleaned early in the startup process.

A very inexpensive solution to the problem of extremely high server load after a power outage is to simply ask users to shut down their computers if power fails. Obviously, not everybody will comply, but if you get even 50% compliance, you will cut your power-on DHCP server load in half.

Hardware Failures

Hardware failures are usually less common than power failures, but they have much the same effect: A DHCP server stops running and doesn't come back again until some problem is corrected. In the worst case, a hardware failure may actually cause the loss of the DHCP database, and it may be necessary to recover the database from backups.

Planned Outages

You may choose to shut down the DHCP server to perform maintenance on it. Unless the maintenance takes a long time, the potential problems that occur in such cases are much the same as in a power outage. DHCP clients without leases are unable to obtain them while the server is down. As long as the server is working again before half of the lease duration is assigned to clients, however, clients with valid leases when the server is down continue to operate normally.

At some point you might also choose to move your network's DHCP service to a different computer with a different IP address. The easiest way to make this infrastructure change is to shut down the DHCP server on the old computer, copy all the configuration files to the new computer, and restart the DHCP server on the new computer. The new server then acquires all assigned addresses and lease information. Computers using that DHCP service discover the new server when they restart or extend existing leases.

Resource Starvation

For most sites, DHCP is a fairly low-demand service that is unlikely to place a heavy load on a DHCP server machine. However, if the DHCP server machine also provides other services, some of which are high demand, resource starvation caused by those services can affect the DHCP server. Also, if a DHCP server is configured to support enough clients, it is possible to create a load so large that the server is unable to keep up (although this has not been observed in practice). Finally, having a great deal of broadcast or multicast traffic on the network to which the DHCP server machine is connected creates a load as well.

To avoid resource starvation caused by competing network services on the same host computer, make sure not to overcommit the machine on which DHCP service is running. If you envision providing high-demand NFS service, consider running the DHCP server on a different computer. If you run a very large DNS service, run it on a separate computer (this should be a problem for only very large sites).

To avoid resource starvation caused by a very high DHCP client load, you might want to install DHCP service on multiple computers rather than on a single DHCP server. Load-sharing can be provided between DHCP servers by using the failover protocol described in the Chapter 18, "Failover Configuration."

BROADCAST- AND MULTICAST-INDUCED LOADS

Broadcast and multicast traffic on a network can require computers connected to the network to process every broadcast or multicast packet, even if the packet is of no interest. In general, IP networks have relatively little broadcast traffic, but networks running other protocols in addition to IP may see more broadcast traffic.

Multicast traffic is intended to reach only computers with clients that are interested in such traffic. Unfortunately, the way multicast is implemented on some network cards requires that the computer examine all multicast traffic, even if it is not of interest. If you run heavy multicast traffic on the network to which your DHCP server is connected, make sure that the DHCP server does not subscribe to this traffic. Also, be sure that the network adapter and the driver for that adapter have efficient multicast filters and do not require the computer to perform multicast filtering. Even if the network interface card correctly supports multicast, the presence of high-bandwidth multicast traffic on a broadcast network can consume enough bandwidth that all other services experience a loss of reliability.

Network Infrastructure Failures

DHCP service requires a working network connection between the DHCP client and the DHCP server. If the client and server are not connected to the same network segment, a working DHCP relay agent must exist.

Failure of Network Hardware

If you experience a network outage that prevents computers holding valid IP addresses from using the network, it probably doesn't matter if DHCP service is interrupted, nobody can use the network anyway. However, if you have local services on a network that a DHCP user might access, you probably want the DHCP service to be at least as reliable as the network to which that user is connected. DHCP service should always be at least as reliable for any given DHCP user as the network connection between that user and the services he or she uses.

The easiest way to meet this goal is to simply have a reliable DHCP server close to the other services that any given DHCP user needs. For example, if you have several LANs, each with its own servers and clients, a DHCP server should run on each LAN. If you have several LANs with client machines and a single LAN in a machine room with all the servers, you really need only a DHCP server in the machine room.

Another way to avoid losing DHCP service when other services are still available is to increase the network's reliability. Within a single site, you might want to set up redundant power sources for routers, switches, and bridges. If you run a central DHCP server to manage addresses across a large corporate network, serving sites connected only by wide area links, you can establish redundant paths. This means that any given site must be connected to the central DHCP site by more than one wide area link. This is harder to accomplish than it sounds; it is quite common to buy WAN links from two different telephone companies, only to learn that both links run through the same conduit between your site and some central distribution point. One company might also be buying bandwidth from the other, in which case both links could be running on the same optical fiber from one site to the other.

Even having each site connected to two or more different sites does not prevent a single point of failure. Links may still run through the same conduit to a single switching office before they are routed to two different sites. A catastrophic failure at the switching site could take down your network *and* your DHCP service, which means that your site wouldn't be able to talk to the outside world or operate independently while steps were being taken to restore connectivity.

Misconfigured or Failed Relay Agents

Most relay agents are embedded in dedicated routers and most likely will not fail unless the routers themselves fail. Relay agents can be misconfigured; for example, if the DHCP server receives a new IP address, relay agents that aren't reconfigured with the server's new address will fail. When changing the DHCP server's IP address, you must update all relay agents as well.

Improving Reliability by Using Long Leases

As mentioned earlier, one way to maintain DHCP service across an outage is to define leases so that they don't expire during a normal outage. Configuring lease durations is discussed in more detail in Chapter 19, "Tuning a DHCP Service."

Setting Up a Secondary DHCP Server

Another way to maintain DHCP service in the presence of a partial power loss or partial network outage is to set up two DHCP servers and enable them to both serve the same network. It might be worthwhile to set up each server on a different network. In this case, if you lose connectivity to or power for one network but not the other network, DHCP service continues.

In order for two DHCP servers to provide DHCP service for the same network segment (or segments), the servers must coordinate their behavior. Each server must either know what the other is doing or be configured so that it can operate without knowing what the other is doing. In order for each server to know what the other is doing, you must use the DHCP failover protocol or a protocol that provides similar functionality. Chapter 10, "Failover Protocol Operation," and Chapter 18, "Failover Configuation," describe the operation and use of the DHCP failover protocol.

NOTE

In this chapter, it is assumed that the servers are unable to coordinate and thus must be configured to operate without knowledge of one another's actions. Although the failover protocol solves many of the problems described in this chapter much more elegantly than the solutions presented here, it is still under development. At the time of this writing, two vendors are known to have deployed implementations of a previous version of the failover protocol, and the protocol draft was undergoing a major revision. By the time this book is published, some preliminary implementations of the new failover protocol may be available.

Dynamic Address Allocation

In order for two servers to perform dynamic (or automatic) address allocation on the same network, they must allocate addresses from separate address pools. If both servers are configured with the same address pools, over time they will independently give the same addresses to different clients. DHCP is a fairly robust protocol in the face of this; you may not notice that anything is wrong, but you might see an abundance of abandoned leases. Unfortunately, over time this means that both servers will perform less and less well and may eventually stop working entirely. The solution is to give each server a separate, nonoverlapping address pool on each network segment. For example, you might configure Server 1 with a range that includes addresses 10.127.42.5 through 10.127.42.130 and Server 2 with a range that includes addresses 10.127.42.131 through 10.127.42.253. When a client tries to start up for the first time, both servers try to offer addresses. Server 1 might offer 10.127.42.5, and Server 2 might offer 10.127.42.131. The client chooses one of these addresses, requests it from the appropriate server, and starts using the lease. As long as the client needs the lease and the server from which it got the lease is working, the client operates normally. Because the two servers are assigning addresses from different ranges, there is no chance that each server will assign the same address to different clients.

If the server fails, the client continues using the address it was assigned until its lease expires. During that period, it tries from time to time to renew its lease. As long as the server comes back before the lease expires, the client continues to operate normally. If the server isn't reachable by the time the lease expires, the client acquires a new lease from the other server, assuming that the server is running.

Static Address Allocation

When assigning addresses to clients statically, as described in Chapter 16, "Client Identification and Fixed-Address Allocation," both servers should be configured with the same list of static addresses for clients.

If both servers are running when a client requests an address, both offer it the same address. The client selects one of the two servers, requests the address from that server, and contacts that server for a lease extension. If the server the client initially selected is unable to respond, the client extends its lease through the other server.

In a sense, when you set up two DHCP servers to provide static address assignment for the same network, you set up communicating DHCP servers. The servers communicate through whatever mechanism you choose to use to provide them with identical static address configurations. The effectiveness of this strategy depends on the degree to which the server configurations remain synchronized.

Hybrid Allocation Models

As mentioned in Chapters 15 and 16, you can set up a DHCP server that performs static address allocation for some clients on some networks and dynamic allocation for other clients on other networks. If you do this, you can still set up redundant DHCP service.

For static address assignments, you simply maintain duplicate DHCP configurations on both servers. For dynamic assignments, you maintain separate address allocation pools for each server. Clients with statically assigned addresses see no apparent interruption in service when a server goes down for longer than their lease duration. Clients with dynamically assigned addresses see an interruption if the server from which they got their addresses goes down for longer than their lease duration.

NOTE

When you run a redundant DHCP server configuration that mixes static and dynamic addressing, it is vitally important that the static address assignments for each server be consistent. If both servers are running and one server has a static address assignment for a particular client and the other has no such assignment, when that client tries to start up, each server offers it a different address. This could mean that the server that knows it has a valid static address would send a `DHCPNAK` message whenever the client sends a `DHCPREQUEST` message for the nonstatic address offered by the other server.

Also, if the client gets the dynamically assigned address from Server 1, and then Server 1's static address mapping is updated to include a mapping for that client, the client is forced to reinitialize with a new IP address, resulting in an interruption of network service for the user.

To avoid this, you should shut down both servers when installing a new server configuration, install the new configuration on both servers, and then start up both servers. As long as you follow this sequence, both servers can run at the same time with consistent states. Of course, this also means that both servers will be down at the same time, but as long as the startup time on both servers is reasonably short, this shouldn't be a problem.

Problems with Setting Up Redundant Servers

Setting up redundant servers that don't speak an interserver protocol is not a good solution to the problem of providing redundant service. If you can find a server that provides a reliable interserver protocol, such as the IETF DHCP failover protocol, it is best to use that protocol because it solves all the problems described in this section.

Address Consistency Rule Violations

When two servers are set up in a redundant configuration with dynamic address allocation, clients may not be assigned the same IP address every time they start up. This can happen even in situations in which no outage exists; if a client relinquishes its address on shutdown or simply doesn't remember its previous address across restarts, each time it comes up, both DHCP servers compete to give it an IP address. This means that clients do not have consistent addresses across restarts. Because DHCP tries to ensure that the client always gets the same IP address, a redundant DHCP server setup with dynamic allocation behaves differently than a single server with dynamic allocation—and this might be a problem for some users.

Loss of Address While in Use

Using multiple servers with separate address ranges doesn't solve the problem of a DHCP client losing its IP address when the lease expires. If a client is assigned an address from one server and is unable to renew the lease with that server before it expires, the other server assign it a different IP address. Any network connections that were in use when the lease expired are broken, probably without warning. For this reason, it is a good idea to use longer leases—even with a redundant server configuration—as an additional strategy for improving reliability.

Dynamic Allocation Pool Starvation

If you set up two DHCP servers to perform dynamic address allocation on a particular network segment, you must assign each server a part of the address pool. In the event of an outage, it might be necessary to support the entire network segment with only the addresses assigned to the server that has not failed. This means you may have to allocate twice as many addresses as are actually needed. If addresses are in short supply, you might not have enough IP addresses to adequately serve that network segment.

Duplicate Responses from Redundant Servers

Whenever a client transmits a `DHCPDISCOVER` message on a network with redundant DHCP servers—whether the servers use static or dynamic address allocation—both servers respond. This means the client generally must choose between two offers, one from each server. This creates extra network traffic and causes both servers, rather than just one server, to do work. It also means that you really cannot have a primary and a secondary server; the two servers compete as equals.

When a DHCP client sends a request, the `secs` field in the DHCP packet is initialized to the number of seconds that have elapsed since the client first started sending the request. One possible way to set up a primary server and a secondary server is to have the secondary server respond only if the `secs` field is larger than some predetermined amount. Because the `secs` field is zero for the first request a client sends, the secondary server does not respond to it, but it does respond if the client sends a second request. You can configure version 3.0 or later of the ISC DHCP server to do this, as shown in Example 17.1.

Example 17.1

```
min-secs 15;
```

In Example 17.1, the server is configured not to respond unless the `secs` field in the client's request has a value of at least 15. Whether all clients use the `secs` field correctly is unknown. This is something to be aware of: If redundant DHCP service is set up in this way, clients that don't correctly set the `secs` field are simply ignored by the secondary server.

Summary

The major causes of DHCP server failure are power failures, hardware failures, and network problems. To improve reliability, you can use long lease times and set up secondary DHCP servers. These tactics work best when you combine them, although you can also use them separately. Long lease times by themselves are a very easy solution, but they don't solve all the reliability problems that can come up; in particular, they work poorly for mobile clients and for clients that are powered on during DHCP service outage.

You can improve reliability by having more than one DHCP server serve a given network segment. This is advantageous because it means that even clients that lose their addresses while one DHCP server is unavailable can still use the network. Redundant DHCP service is not an ideal solution; the DHCP failover protocol is a solution that better handles the problem of setting up redundant service.

18

Configuring a Failover Server

IN THIS CHAPTER

This chapter discusses failover in practical terms. In order to get the maximum benefit from this chapter, you should also read Chapter 10, "Failover Protocol Operation," which describes the details of the failover protocol and how it operates.

This chapter does not describe how to set up a really complicated failover configuration—just the basic configuration. This should be enough to provide you with the building blocks necessary to produce a more complicated configuration if you need one.

This chapter describes the three basic kinds of failover pair configurations and the uses of each. It then describes the process of adding support for failover in your environment, based on the kind of DHCP service you presently have. It then walks you through the process of configuring a pair of ISC DHCP servers that were not using failover so that they do use it. This chapter then describes some of the issues that can come up if you operate a failover pair and how to deal with them. Finally, it describes issues that are specific to the ISC DHCP server.

Types of Failover Relationships

There are three main categories of failover relationships that you might want to set up:

- Cooperating partners
- Failover
- Backing store

Remember that failover servers always appear in pairs. These three categories of relationships define how each member of a failover pair behaves. Which type of failover relationship you choose depends on what you are trying to accomplish. The type of failover relationship you choose determines some aspects of how you configure your server, as shown in Table 18.1.

TABLE 18.1 Parameters That Determine the Failover Server Relationship Type

Role	Listens	Responds	Free Address Balance
Cooperating Partners Relationship			
Primary	Yes	Yes	50%
Secondary	Yes	Yes	50%
Primary/Backup Relationship			
Primary	Yes	Yes	50%
Backup	Yes	Sometimes	50%
Backing Store Relationship			
Active	Yes	Yes	100%
Backing store	No	Never	0%

The Cooperating Partners Relationship

In a cooperating partners relationship, the two failover servers are both acting as DHCP servers for a common set of network segments, all the time. Both DHCP servers receive every client message. The servers should be using load-balancing, and the load should be shared equally between the two failover partners.

A cooperating partners relationship works well in environments where you want both redundancy and load-balancing; if one server fails, you want DHCP service to continue, but while both servers are running, you want them to share the DHCP service load.

The Failover Relationship

A primary/backup relationship provides redundancy but not load-balancing. In a primary/backup relationship, one of the two servers (not necessarily the server whose failover role is “primary”) is the primary server, and the other server acts as a backup. Both servers hear all DHCP requests for the network segments that they serve, but normally only the primary responds.

The backup server responds if a client indicates that it has been unable to contact the primary server for more than a predetermined amount of time. The backup server also responds if it is out of contact with the primary server.

When the backup server fails, the primary server has to follow the same failover rules that apply for cooperating partners—it can't allocate leases that are in the `BACKUP` state. When the primary server fails, the backup server takes over service, but it also has to follow the failover rules, and it can't allocate leases that are in the `FREE` state.

It's impossible to know which server will fail, so even though one server is doing all the allocations, it's still important to keep the pools roughly in balance—that is, there should be roughly the same number of backup leases as free leases. Because the primary server takes backup leases from the secondary server as its pools become depleted, the pools tend to stay in balance as long as the servers are communicating.

The Backing Store Relationship

When two servers are in a backing store relationship, one server, the active server, uses the backing store server to back up its database. The other server, the backing store server, never serves any clients, nor does it hear requests from clients. In this configuration there is no failover-style redundancy; if the active server fails, DHCP service stops until the active server comes back.

The advantage of this type of failover configuration is that it allows you to deploy a dataless DHCP server—that is, a DHCP server that runs in a device that has no nonvolatile storage for its lease database. This server uses the backing store server to save its lease database.

Normally, the active server behaves just like a regular DHCP server, but it maintains a failover relationship with the backing store server. When the backing store server goes down, the active server continues to operate in communications-interrupted mode.

When the active server is powered down or rebooted, it loses its lease database. So when it starts back up, it contacts the backing store server and rereads its configuration database. It can then proceed to allocate IP addresses as usual.

According to the failover protocol, if the backing store server is unreachable when the active server comes up, the active server can't serve DHCP requests until it regains contact with the backing store server. This is a major weakness in this kind of failover relationship. One solution to the problem is to allow the server to extend leases by the minimum client lead time (MCLT) when clients request it but not to allocate any new leases.

Setting Up Failover Service for the First Time

There are three possible scenarios for a new failover installation:

- You currently have no DHCP service.
- You have one DHCP server providing all service.
- You have two DHCP servers providing service out of disjoint pools.

The failover protocol does not provide a mechanism for synchronizing server configurations—all it provides is a way to share IP address allocation pools. So in all three cases, you must create a master DHCP configuration that applies to both members of the failover pair.

If you have no DHCP server, you can simply create a DHCP configuration as described in Chapters 15, "Configuring a DHCP Server," and 16, "Client Identification and Fixed-Address Allocation." You might want to test the configuration with a single DHCP server before you proceed to setting up a failover pair.

If you currently have a single DHCP server, the master configuration is that DHCP server's configuration. If you have two DHCP servers serving addresses out of disjoint allocation pools, then you must combine the two server configurations to produce a single configuration.

After you have created the master DHCP configuration, you must create an individual configuration for each of the two servers, based on the master configuration. The only difference between the two configurations is the failover configuration information—everything else should be the same.

A DHCP configuration file for a DHCP server that does dynamic allocation must either explicitly or implicitly declare one or more address allocation pools. If you want to share these pools using failover, you must mark each pool that you want to share because it is possible for the DHCP server to share some pools and not share others.

If you already have DHCP service operating without failover, whether it is a single DHCP server or a pair of servers with disjoint allocation pools, you do not need to do anything with the lease files—just leave them as they are. The failover protocol assumes that the failover pair as a whole is completely in control of the lease database. The lease database on an individual server does not reflect the complete state of the lease database. Therefore, it is *very* important not to change the lease database on an individual server yourself.

NOTE

It is very common for someone setting up failover for the first time to provide failover configuration information for both servers but not configure the allocation pools to be shared. However, if someone did do this, both servers would behave as if they were in a failover relationship, in the sense that they would log messages about connecting and about state changes, but they would not actually share their address allocation pools. Because they would both be configured with the same address allocation pool, this would be disastrous: Both servers would allocate IP addresses out of the same pools without cooperating.

Configuring the ISC DHCP Server to Do Failover

To illustrate the failover configuration process, let's use the ISC DHCP server as an example. The ISC DHCP server currently supports only the cooperating partners and primary/backup relationships. This example describes the cooperating partners relationship, but it also describes in a general way how the configuration would need to be changed in order to implement a different relationship.

If you do not currently have DHCP running at your site, we will assume that you have already written a master DHCP configuration by using the information in Chapters 14, "The ISC DHCP Server," 15, "Configuring a DHCP Server," and 16, "Client Identification and Fixed-Address Allocation."

Merging Configuration Files

If you have two DHCP servers serving IP addresses out of disjoint pools, you must merge the DHCP configurations. In this example, we refer to the two disjoint servers as Server A and Server B. The configuration file for Server A is shown in Example 18.1.

Example 18.1

```
option domain-name "example.org";
option domain-name-servers 10.0.1.17;

subnet 10.0.1.0 netmask 255.255.255.0 {
 option routers 10.0.1.1;
 pool {
      deny dynamic bootp clients;
      range 10.0.1.10 10.0.1.109;
 }
}
subnet 10.0.2.0 netmask 255.255.255.0 {
 option routers 10.0.2.1;
```

Example 18.1 Continued

```
 pool {
      deny dynamic bootp clients;
      range 10.0.2.10 10.0.2.209;
 }
}
```

The configuration file for Server B is shown in Example 18.2. As you can see, there are four differences between the two configuration files:

- The `domain-name-servers` option is different.
- The allocation range for subnet 10.0.1.0 is different.
- Server A serves subnet 10.0.2.0, and Server B does not.
- Server B serves subnet 10.0.3.0, and Server A does not.

Example 18.2

```
option domain-name "example.org";
option domain-name-servers 10.0.1.18;

subnet 10.0.1.0 netmask 255.255.255.0 {
 option routers 10.0.1.1;
 pool {
      deny dynamic bootp clients;
      range 10.0.1.110 10.0.1.209;
 }
}

subnet 10.0.3.0 netmask 255.255.255.0 {
 option routers 10.0.3.1;
 pool {
      deny dynamic bootp clients;
      range 10.0.3.10 10.0.3.209;
 }
}
```

In order to merge these two configuration files into one, you must resolve the differences between them. The first difference is actually the most difficult to resolve. Each DHCP server sends a different `domain-name-servers` option. This ensures that each name server serves half of the DHCP clients on the network. There is no way to preserve the exact behavior of the two disjoint servers in this case, but what you can do is use different name servers for different pools. Example 18.3 shows one way to do this.

Example 18.3

```
option domain-name "example.org";

subnet 10.0.1.0 netmask 255.255.255.0 {
 option routers 10.0.1.1;
 pool {
      option domain-name-servers 10.0.1.17;
      deny dynamic bootp clients;
      range 10.0.1.10 10.0.1.109;
 }
 pool {
      option domain-name-servers 10.0.1.18;
      deny dynamic bootp clients;
      range 10.0.1.110 10.0.1.209;
 }
}

subnet 10.0.2.0 netmask 255.255.255.0 {
 option routers 10.0.2.1;
 option domain-name-servers 10.0.1.17;
      pool {
      deny dynamic bootp clients;
      range 10.0.2.10 10.0.2.209;
 }
}

subnet 10.0.3.0 netmask 255.255.255.0 {
 option routers 10.0.3.1;
 option domain-name-servers 10.0.1.18;
      pool {
      deny dynamic bootp clients;
      range 10.0.3.10 10.0.3.209;
 }
}
```

The second difference is that each server is serving a different pool on subnet 10.0.1.0. There are two ways to solve this problem. The first is to copy the pool declarations into the master configuration file unchanged, as shown in Example 18.3. The second is to merge them. In this example, we have chosen to keep the two pool declarations separate so that we can send a different `domain-name-servers` option, depending on the pool from which a client's address comes.

The third and fourth differences are easily solved. We simply copy the two subnet declarations, along with their pools, into the master configuration. In order to preserve the `domain-name-servers` behavior from the disjoint configuration, we move the `option domain-name-servers` statement into each subnet declaration. The final result is shown in Example 18.4.

Example 18.4

```
option domain-name "example.org";

subnet 10.0.1.0 netmask 255.255.255.0 {
 option routers 10.0.1.1;
 pool {
      option domain-name-servers 10.0.1.17;
      deny dynamic bootp clients;
      range 10.0.1.10 10.0.1.109;
 }
 pool {
      option domain-name-servers 10.0.1.18;
      deny dynamic bootp clients;
      range 10.0.1.110 10.0.1.209;
 }
}

subnet 10.0.2.0 netmask 255.255.255.0 {
 option routers 10.0.2.1;
 option domain-name-servers 10.0.1.17;
 pool {
      deny dynamic bootp clients;
      range 10.0.2.10 10.0.2.209;
 }
}

subnet 10.0.3.0 netmask 255.255.255.0 {
 option routers 10.0.3.1;
 option domain-name-servers 10.0.1.18;
pool {
      deny dynamic bootp clients;
      range 10.0.3.10 10.0.3.209;
 }
}
```

After you have created your master DHCP configuration file, you must establish some kind of discipline for how you are going to maintain it. If you do not, the two configuration files will change over time and become different, and that makes them difficult to maintain. Many ISC DHCP users keep their DHCP configurations in an open-source product called Concurrent Versioning System (CVS; see `www.cvshome.org`).

Configuring the Cooperating Partners Relationship

The configuration for each member of a simple DHCP failover pair is almost identical, but there is one major difference: the portion of each configuration that describes the failover relationship. The failover configuration differs on the two servers for two reasons. First, each server's configuration has to describe how to contact the other server, and second, one server is primary and one is secondary. Other than this, the two configurations should not be different from one another.

Failover Configuration Parameters

To configure a failover relationship on the ISC server, you need to write a failover peer declaration. If two DHCP servers have a failover relationship, a failover peer declaration for that relationship must appear in the configuration file of each server. Each failover peer declaration has a name and a sequence of data defining the relationship. For our configuration example, we use the values shown in Table 18.2, which are consistent with the cooperating partners relationship.

TABLE 18.2 Failover Peer Declaration Information

Setting	Value on Primary	Value on Secondary
Contact address	10.0.1.1	10.0.1.2
Contact port	847	647
Partner address	10.0.1.2	10.0.1.1
Partner port	647	847
Contact timeout	180	180
Maximum pending updates	100	100
MCLT	1800	Not applicable
Free address balance	50%	50%
Load balance split	50%	Not applicable
Load balance override	3	3

The following sections describe these settings.

Setting Server Roles

The role setting defines the role that the server plays in the failover pair. One server is always defined as the primary server, and one server is always defined as the secondary server. The designation of primary or secondary is used to determine which server goes first in certain protocol negotiations. In the ISC DHCP server, there is never any operational difference between how the primary and secondary servers act, so it doesn't matter which server is primary and which is secondary. The role setting in the ISC server is specified with the `primary` or `secondary` keyword.

Setting Contact and Partner Address and Port

Each server has a contact port and contact IP address—these are the port number and IP address on which the server listens for connections from its partner. The partner address and port are the IP address and port to which each server tries to connect when it is out of communication with its peer. The contact address and port of the primary server is always the partner address and port of the secondary, and vice versa. This is a requirement, even for a server that has more than one IP address, because the contact address is used as an identifier for the connection; if either server sends a contact address that is not the one the other server is expecting to receive, the other server refuses the connection. You should use the port numbers shown in Table 18.2 unless you have some specific reason to use different port numbers.

In the ISC server, the contact address is specified with the `address` statement, and the contact port is specified with the `port` statement. The partner address is specified with the `peer address` statement, and the partner port is specified with the `peer port` statement.

Contact Timeout

The contact timeout determines how long a server will wait without receiving any messages from its partner before it assumes that the connection to its partner has failed. In this example, we have chosen a timeout of 180 seconds, or 3 minutes. This allows either server to quickly notice a connection failure with its partner, but it prevents a temporary network outage—for example, a wire being unplugged from one port and plugged into another—from breaking the connection between the servers prematurely. In the ISC server, the contact timeout is specified with the `max-response-delay` statement.

Maximum Number of Pending Updates

The maximum number of pending updates defines the number of updates that the server can accept without blocking input. This parameter might be a hard limit configured by the server, but the ISC DHCP server always processes updates as they arrive, so the only reason to choose a particular value for this parameter with the ISC server is to avoid having the partner send too *few* updates at a time. In general there is no need to configure this parameter, but we mention it here for completeness. If your DHCP server does not insist that you choose a value for this parameter, you shouldn't try to configure it. You do not need to configure it on the ISC server; 100 is the default. But if you want to configure this in the ISC server, use the `max-pending-updates` keyword.

MCLT

The MCLT is the maximum amount of time by which either server can extend a lease without contacting the other server. This value has to be a compromise between client lease time and recovery time. You should choose a value that is reasonably long so that clients that get a lease that is MCLT seconds long have a useful lease that won't lead to instability for them. You must not choose a value that is too long because the MCLT is also the recovery interval for the server. That is, the longer the MCLT is, the longer it takes to return to normal failover operations after a server failure.

One other point to consider is that if clients generally get short leases, they need to renew more often than if they had longer leases. If the normal lease interval for a client is 5 hours and the MCLT is 30 minutes, when the servers are operating in the `COMMUNICATIONS-INTERRUPTED` state, the load on each server is 10 times as great. If the load on the servers is very light, as is the case with most DHCP servers, this really isn't a problem. However, if your DHCP server is serving a very large network, it might not be able to gracefully handle a tenfold increase in load.

NOTE

To better understand the implications of using short leases, refer the section in Chapter 10 titled "Operation in the `PARTNER-DOWN` State," particularly the part that talks about the start time of state. You should also read the description of the section of Chapter 10 titled "Operation in the `RECOVER` State." The section titled "Operation in the `COMMUNICATIONS-INTERRUPTED` State" is helpful for understanding why a really short value for MCLT could be a problem.

The MCLT is configured only on the primary server, in order to avoid disagreements between the primary and secondary servers about its value. On the ISC server, MCLT is configured by using the `mclt` statement.

Free Address Balance

The free address balance is the balance that the primary server tries to strike between IP addresses in the `FREE` state and in the `BACKUP` state. Addresses in the `FREE` state are available for allocation on the primary server, and addresses in the `BACKUP` state are available for allocation on the secondary server. The ISC DHCP server does not support configuring the free address balance; it only supports a balance of 50% free/50% backup.

Some other DHCP servers that implement the failover protocol support other free address balances. The 50% free/50% backup balance works both for the cooperating partners relationship and the primary/backup relationship. For the backing store relationship, the balance should be 100% free/0% backup, assuming that the backing store server is configured as the secondary server.

Load Balance Split

The load balance split tells the primary server what portion of all clients it should serve. Each client is assigned to one of 256 different groups, according to the identification information it sends. Using the load balance split, the primary server constructs an array of 256 values and sets some elements of the array to one and some to zero. If a client's entry in the array is one, the primary server serves that client. If it is zero, the secondary server serves the client. This allows the two failover peers to split the workload. The ISC DHCP server allows you to specify either the load balance split or the individual values in the 256-element array. In general, there's no reason to specify the bitmask directly, and in this example we specify the split.

You can use the load balance split to put two DHCP servers into a primary/backup relationship. In a primary/backup relationship, the primary server serves all clients and the backup server serves none, unless the primary server doesn't respond for some reason. This corresponds to a split value of 256—that is, all 256 elements in the array are set to one.

On the ISC server, the load balance split is configured with the `split` keyword, and if you want to specify the load balance array directly, you use the `hba` keyword.

Load Balance Override

The load balance override parameter determines when the primary or secondary server will bypass load balancing and respond to the client even if the client is supposed to be served by the other server. Every message from a DHCP client includes a `secs` field, which indicates for how many seconds the DHCP client has been trying to contact a DHCP server. If the value of the `secs` field in a DHCP message is greater than the load balance override parameter, the DHCP server always attempts to respond to the client, regardless of the load balance split. On the ISC server, the load balance override is specified by using the `load balance max seconds` keyword.

The ISC Failover Configurations

Given the parameters defined in the preceding sections, we now need to write two failover declarations. The first, shown in Example 18.5, is the configuration for the primary server. In order to combine the primary failover configuration with the master DHCP server configuration, the primary configuration file needs to include the master configuration. In this example, the primary configuration is in the file `/etc/dhcpd.conf`, and the master configuration is in the file `/etc/dhcpd.master`.

Example 18.5

```
failover peer "example" {
 primary;
 address 10.0.1.1;
 port 847;
 peer address 10.0.1.2;
 peer port 647;
 max-response-delay 180;
 mclt 1800;
 split 128;
 load balance max seconds 3;
}

include "/etc/dhcpd.master";
```

The failover configuration for the secondary server is shown in Example 18.6.

Example 18.6

```
failover peer "example" {
 secondary;
 address 10.0.1.2;
 port 647;
 peer address 10.0.1.1;
 peer port 847;
 max-response-delay 180;
 load balance max seconds 3;
}

include "/etc/dhcpd.master";
```

Final Failover Configuration Details

The example configuration files presented so far are very nearly complete, but there are two last details that we haven't talked about yet. First, because the ISC server supports arbitrarily complex failover configurations, it does not assume that every address pool mentioned in the configuration file is part of a single failover relationship defined at the top of the file. You must explicitly define the failover relationship for each pool. You do this by writing a failover reference within the `pool` statement. Example 18.7 shows an example of a failover reference statement in one of the pools from Example 18.3.

Example 18.7

```
pool {
 failover peer "example";
 option domain-name-servers 10.0.1.17;
 deny dynamic bootp clients;
 range 10.0.1.10 10.0.1.109;
}
```

The second detail is that the ISC DHCP server does not support failover on address allocation pools that contain addresses allocated to BOOTP clients. So if you try to configure a pool for failover but leave out the `deny dynamic bootp clients` statement, the DHCP server reports an error and refuses to run. To correct this error, you simply add a `deny dynamic bootp clients` statement to the `pool` declaration. Be careful not to confuse the `deny dynamic bootp clients` permit statement and the `deny bootp` statement. The `deny bootp` statement does not work in a `pool` declaration, and it does not correct this error.

Operating a Failover Pair

After you have configured your failover servers, you can begin to use them. The first step in using them is to get them to communicate with one another. After you do this, they are operational.

Starting the Servers for the First Time

When you start a failover pair for the first time, the two servers generally refuse to do anything until they have synchronized with each other. So the first order of business is to get them to talk to each other. If you have configured the servers correctly, this should work without any trouble. If the two servers don't seem to be able to communicate, you should check the address and port settings carefully.

The ISC DHCP server does not trust the timekeeping protocol described in the failover protocol specification. Instead, it requires that you keep the system clocks on both failover partners synchronized. If the system clocks on the two servers are not synchronized, the servers refuse to talk to each other, and you see messages in the system log telling you that the clocks aren't synchronized. The ISC server doesn't require any particular synchronization mechanism—it is fine to synchronize them by hand. However, it's much easier to synchronize them by using the Network Time Protocol (NTP). NTP clients are available for most operating systems, so this should not be a serious problem.

When the two servers start for the first time, they start in the `RECOVER` state. After they have established communications, each server sends the other server a complete list of all the leases it has. Through this process, the two servers synchronize their lease databases. This is why it's a mistake to copy the lease database from a stand-alone server to its partner when you convert it to a failover pair; if you do that, both servers have identical lease files, and they take twice as long to synchronize.

When the servers are synchronized, they might both wait out the MCLT before beginning to serve clients. This is the behavior required by the failover protocol when a server is in the `RECOVER` state. However, if you are starting up for the first time, both servers are in the `RECOVER` state, which isn't a desirable situation. Some DHCP servers, including the ISC DHCP server, bypass the waiting period if they detect that both servers are in the `RECOVER` state because this can usually only happen the first time two servers are configured to do failover.

Normal Operations

After the servers have synchronized, they begin normal operations. This doesn't mean the `NORMAL` failover state. Normal operations refers to all the failover states described in Chapter 10. During normal operations, two sorts of failover log messages are worth watching for: lease update messages and failover state messages.

When the state of either failover partner changes, you see a message in the log for that state change. The most usual state changes are from `NORMAL` to `COMMUNICATIONS-INTERRUPTED` and from `COMMUNICATIONS-INTERRUPTED` to `NORMAL`. You see a message about this on one server whenever the other server is stopped. More rarely, you see this message when the network connection between the two servers has failed.

The second sort of log message is a binding update message. The ISC DHCP server is usually quiet about binding update messages. The only time you hear about them in the log is when they fail. The only real reason a binding update would fail is if the server is buggy or the two servers have lease databases that have gone out of sync.

Operational Problems

During operations, a variety of problems can come up. Some of them have to do with the fact that the failover protocol is very new, and existing implementations might still have bugs to work out. Others are just normal operational problems that can come up even if the DHCP servers are not at all buggy.

Server Down

When one server in a failover pair goes down, the other server continues to provide service, but in a limited mode called the `COMMUNICATIONS-INTERRUPTED` state. To learn more about this state, see Chapter 10. Because of the limitations of `COMMUNICATIONS-INTERRUPTED`, if the server that has gone down isn't expected to come back up quickly, it's good to put the other server into the `PARTNER-DOWN` state. In the `PARTNER-DOWN` state, the remaining DHCP server can, after waiting for the MCLT, completely take over DHCP service on the network, including reclaiming all of the down server's IP addresses.

OMAPI

To put an ISC DHCP server into the PARTNER-DOWN state, you need to use the OMAPI protocol. The OMAPI protocol is a control protocol that allows you to connect to the ISC DHCP server and change its state without stopping it.

To use OMAPI, you need to have omapi-port and omapi-key statements in your dhcpd.conf file. The omapi-port statement tells the server to listen for OMAPI connections on the port you specify, and the omapi-key statement tells the server what authentication key to use to authenticate your OMAPI connection. A sample configuration is shown in Example 18.8.

Example 18.8

```
key EXKEY {
 algorithm HMAC-MD5.SIG-ALG.REG.INT;
 secret pRP5FapFoJ95JEL06sv4PQ==;
};

omapi-port 520;
omapi-key EXKEY;
```

When the DHCP server is configured to use OMAPI, you can connect to it by using an OMAPI client and issue commands to the server. There is an interactive OMAPI client called omshell that is ideal for simple server changes like this. To use omshell to put the "example" failover peer on the primary server into the PARTNER-DOWN state, assuming the configuration shown in Examples 18.4, 18.5, and 18.7, you use the commands shown in Example 18.9.

Example 18.9

```
% omshell
> server 10.0.1.1
> key EXKEY pRP5FapFoJ95JEL06sv4PQ==
> port 520
> connect
obj: <null>
> new failover-state
obj: failover-state
> set name = "example"
obj: failover-state
> open
obj: failover-state
 …
> set local-state = 1
```

Example 18.9 Continued

```
obj: failover-state
 …
> update
obj: failover-state
 …
> ^D
%
```

Getting Out of the PARTNER-DOWN State

If you have been operating a single server in the PARTNER-DOWN state and are ready to bring back the other server, you don't need to do anything special except make sure the two servers can communicate. When you start the server that was down, it automatically connects to the running server, synchronizes with it, waits for the MCLT to expire, and begins serving clients.

You must be careful not to restart the failed server when the server that is in the PARTNER-DOWN state is not running. If the failed server starts up and is unable to contact the other server, it assumes that things are as they were when it went down, and it begins allocating IP addresses that the other server has reclaimed. So as a rule, you should have an administrative policy that before you put a server into the PARTNER-DOWN state, you make sure that its peer won't come back up unexpectedly.

Loss of the Database on One Server

The failover protocol provides a completely transparent mechanism for restoring a lost database. All you have to do is generate a configuration file for the server that lost its database and start it up. It automatically connects to the other server and downloads that server's entire database.

Lost Leases

With the ISC server, it's sometimes possible to get into a state where, because of software bugs, particularly in older versions, the lease databases are not in sync. This sort of problem can persist even after you upgrade to a newer version of the server without the bugs. The usual symptom of this is log messages such as "Peer holds all free leases" and "No free leases," when you know there are plenty of available IP addresses. Another symptom of a lost lease is that a client always gets leases that are the MCLT long, even though the servers are operating in the NORMAL state. The message "Invalid binding state transition: active to expired" might indicate that the server still has this bug. It might also indicate some other problem.

There is no one fix to this problem. The easiest thing to try, particularly if you know that your current software is working, is to force the two servers to resynchronize. You can do this by using the OMAPI method shown in Example 18.9, but you put *both* servers into state number 6 (RECOVER) instead of putting one server into state number 1 (PARTNER-DOWN). This forces the servers to resynchronize, and it can help.

If this doesn't work, a more drastic solution is to stop one of the servers and delete its lease database. Then restart it, and it recovers its database from its peer. At this point the two databases are definitely in sync. The problem is that by doing this, you lose all information about any out-of-sync leases. This can result in service interruptions for the clients that held those leases.

Pool Balancing

When the ISC DHCP server tries to balance its pools, it prints a message like this:

```
pool 80c3ea92 10.0.1.0 total 100 free 25 backup 5 lts 10
```

This indicates that a pool in the shared network named 10.0.1.0 has a total of 100 IP addresses. Of those, 25 are in the FREE state and 5 are in the BACKUP state. The `lts` value (leases to send) means that the primary server concludes that it needs to move 10 of its free IP addresses into the BACKUP state for the secondary server to allocate.

If the `lts` is less than 10% of the number of free IP addresses, the server does not attempt to balance the pools. This prevents the servers from rebalancing every time they allocate an IP address, which would be quite inefficient.

Pool balancing is a rough operation—it doesn't always get the pools exactly in balance. You can run into serious problems when there are a very small number of available IP addresses; for example, one server will have all the free leases and the other will have none, so when a new client comes along, it might not be able to get an IP address.

Leases

Failover changes the way leases are allocated. The first lease allocated to a client when it doesn't already have a valid lease is always for the MCLT, not the lease time the client asked for. Also, the DHCP server writes the lease to the database once before it sends the DHCPACK message to the client and then a second time when it gets the BNDACK message back from the failover peer. So every time a lease on an IP address is renewed, it is written to the lease database twice.

Issues Specific to the ISC DHCP Failover Implementation

In addition to the operational issues described so far in this chapter, there are also some limitations to the ISC server that affect failover.

The Version of ISC Software to Use

The ISC DHCP failover implementation is the only publicly available implementation of the current failover protocol at the time of this writing. The ISC implementation is still quite new, and the failover protocol specification itself is not even an official standard of the IETF. If you want to use failover, you should be very careful to run the latest version of the DHCP server. If there are two versions and one is less recent, it is probably more buggy than the other.

Ad hoc DNS Updates

The ISC DHCP server provides two ways of doing DNS updates on behalf of a client. The first is the ad hoc method, which is now deprecated, and the second is the interim method. If you have an ISC DHCP server that uses the ad hoc update method and you want to use failover, you must first switch to the interim method and then switch to failover—you can't do it in one step.

To switch from ad hoc to interim, simply edit your `dhcpd.conf` file and change the `ddns-update-style` statement to say `interim` instead of `ad-hoc`. Then wait until the maximum lease interval has passed, and all the hosts whose DNS registration was done using ad hoc updates either expire or are converted to the interim style.

Known Problems with the ISC DHCP Server and Failover

In some cases, features of the ISC DHCP server that were invented before failover do not too work well if you are using failover. There are also some limitations to the ISC failover implementation—portions of the protocol that were not a high priority when failover was being developed and therefore have not yet been implemented. The following is a list of these issues at the time of this writing:

- The ISC DHCP server provides a feature called *lease limits* that limits the number of leases that can be assigned to members of any specific class. When you are running failover, the two servers calculate this independently. Because of this, the lease limit feature doesn't work and mustn't be used with failover.
- The ISC DHCP server does not provide a way to disable load-balancing.
- The ISC DHCP server does not implement the safe period specified in the failover protocol.
- Information that is added to the lease database by using OMAPI is not shared between servers. This means that, for example, if you use OMAPI to add `host` statements to the server, you have to add the same `host` statement to both servers with OMAPI—failover does not keep this information synchronized.

- The ISC DHCP server does not keep track of static leases. As a result, the failover-style reserved IP addresses are not supported. This isn't a serious problem in most cases; you can simply define any host declarations in the master configuration file that is shared between the two servers, and both servers respond when a client with a reserved address tries to get its IP address.

Summary

The failover protocol provides a way to provide redundant DHCP service from two cooperating DHCP servers. Failover servers can be configured in three different relationships, depending on your needs. If you already have two DHCP servers, you can merge their configurations into a single failover pair configuration. If you have only one DHCP server, you can add a second to get a failover pair. If you don't have DHCP service, you can easily set up a new DHCP failover pair.

Failover servers must use a common configuration file, with a small amount of customization for each server, to describe the failover relationship.

A one-time synchronization process occurs when you first start up failover, and after you have completed this process, the failover protocol becomes operational.

The ISC DHCP server includes an implementation of the failover protocol. If you want to use this implementation, you should make sure to run the most recent version. Some ISC DHCP features do not work with failover.

19

Tuning a DHCP Service

IN THIS CHAPTER

- Network Device Configuration and Address Assignment Strategies
- Configuring Lease Lengths
- Monitoring the Server

The preceding chapters in Part III, "DHCP Servers and Clients," describe the mechanics of installing and configuring a DHCP server. This chapter describes how to tune a DHCP implementation to best meet the needs for DHCP service at your site, and it explains some of the trade-offs you need to consider. It also describes some ways to monitor your DHCP server's activities so that you can assess and monitor how well it is functioning.

Network Device Configuration and Address Assignment Strategies

Although DHCP may be the best choice for configuring many of the computers at your site, you might still need to (or choose to) configure some computers in other ways. In practice, some legacy devices that don't have implementations of DHCP require manual configuration of IP addresses and DNS hostnames. Other legacy devices that use BOOTP for configuration, such as printers and some network hubs, may require manual configuration or provision of BOOTP service. Servers such as the DHCP server, the DNS servers, Web servers, mail servers, and so on *should* (or must) be configured manually to avoid dependence on other services (such as DHCP) and to improve reliability. In addition, network infrastructure devices such as routers must be configured manually.

Manual Configuration

Computers that are assigned IP addresses through manual configuration can interact with a DHCP service in two ways. First, the DHCP service must be configured so that it doesn't try to assign IP addresses to DHCP clients that have already been assigned to other computers through manual configuration. The DHCP administrator has to keep track of the manually configured IP addresses and configure the ranges of available address in the DHCP servers to exclude those IP addresses.

Second, computers or other devices that are manually configured with IP addresses can use DHCP to obtain other configuration information. A network device can use the `DHCPINFORM` message to request other configuration information such as default routes, DNS servers, and printer servers from a DHCP server without being assigned an IP address. Use of the `DHCPINFORM` message allows a network administrator to reconfigure automatically servers and other devices that have been manually configured with IP addresses.

Strategies for Supporting BOOTP Devices

As mentioned previously, BOOTP is the predecessor of the DHCP protocol. A network may include legacy devices such as older printers that can only use BOOTP. Because DHCP and BOOTP are related to each other, it is possible to use both on the same network. RFC 1534 explains the interoperation and coexistence of BOOTP and DHCP in detail.

You can manage legacy BOOTP devices in several ways:

- Eliminate all BOOTP devices and use only DHCP.
- Provide static IP address assignments for BOOTP clients.
- Automatically allocate permanent IP addresses for BOOTP clients.
- Keep BOOTP and DHCP services separate.

Elimination of All BOOTP Devices

One method of dealing with BOOTP devices is to convert them so that they use DHCP. Where possible, this is an attractive method of dealing with the issue because it leaves the administrator with only *one* method of allocating addresses automatically/dynamically. Many computers and network devices that use BOOTP can also use DHCP. In some cases (as with many network printers), converting devices from BOOTP to DHCP is as simple as selecting `DHCP` from a network configuration screen. Firmware upgrades for BOOTP-only devices may be available that support DHCP. Many different kinds of BOOTP devices exist, so it is difficult to provide details on the method of converting any specific device from BOOTP to DHCP.

A certain amount of work is involved in identifying all BOOTP clients on a network and upgrading them to use the DHCP protocol instead. Likewise, if you decide to continue supporting BOOTP, you must do some extra work because BOOTP does not provide for any way of automating the maintenance of BOOTP address assignments.

Even if you choose to have the DHCP server automatically allocate these addresses, you cannot have it reclaim them when they are no longer in use. Whether you choose to convert from BOOTP to DHCP depends on which of these options is likely to require the least amount of work.

Static Address Assignments for BOOTP Clients

If you choose to provide BOOTP service with a DHCP server, you need to move the information about the deployed BOOTP clients to the DHCP server. Some BOOTP clients cannot transition to a new IP address and must continue to use the same IP address that was originally assigned by the BOOTP server. To continue providing all your BOOTP clients with their old IP addresses, you must use static address assignment by following these steps:

1. Configure the DHCP server to respond to BOOTP requests.
2. Examine the BOOTP server configuration file, and for each entry in the file, create a static address assignment on the DHCP server that includes the hardware address and the IP address to be assigned to that client.
3. Make sure that all the appropriate DHCP options are configured in the DHCP server to correspond with all the options specified in the `bootptab` file.
4. Turn off the old BOOTP server.
5. Restart the BOOTP devices.

NOTE

Often, entries in a BOOTP server configuration file are very regular. You might find it easiest to simply write a Perl or shell script to convert a `bootptab` file into the format that the DHCP server expects. Unfortunately, you cannot do this for the Microsoft DHCP server, but it does work with the ISC DHCP server and with DHCP servers provided by quite a few other vendors.

You might also want to ask whether your vendor already has scripts that you can use. The `bootptab` file format varies between different versions of the BOOTP daemon and from site to site. One site's script probably doesn't work for another site without changes, but it can still be useful to start from somebody else's conversion script instead of writing a completely new one.

Many BOOTP devices perform a second-stage startup process, in which they download a file from a TFTP server or an NFS server—for example, a startup image or a configuration file. BOOTP requires that the IP address of the server from which to load this file be included in the `siaddr` and `sname` fields of the BOOTP packet. However, some newer BOOTP clients actually use the DHCP `tftp-server-name` and `bootfile-name` options. You must make sure that you configure the DHCP server to send the second-stage startup information to the BOOTP client in the same way that the BOOTP server does.

You can tell very quickly whether you have this issue. Scan through the BOOTP server configuration file, looking for the tag symbol `sa=value`. This is the tag for the TFTP server address. If this tag is present, you should configure the DHCP server with the `tftp-server-name` option instead of telling it to use the `sname` field in the BOOTP header.

NOTE

Static IP address assignments do not enable BOOTP clients to roam across multiple subnets, unless you create multiple assignment entries for a given link-layer address. Also, if a network is renumbered, the address assigned to a particular BOOTP client might be incorrect for the network segment to which that client is connected. Therefore, static IP address assignment is effective for enabling BOOTP clients to continue receiving their old IP addresses, but it provides little of the benefit of DHCP.

Automatic IP Address Assignment for BOOTP Clients

Rather than completely eliminating BOOTP or providing and maintaining static address allocations for all BOOTP clients, you can have some DHCP servers automatically allocate IP addresses to BOOTP clients. This form of BOOTP service is sometimes called *dynamic BOOTP*.

Dynamic BOOTP is attractive because it enables you to serve BOOTP clients from the same pool of addresses as dynamic clients. It doesn't require you to create DHCP reservations, yet it isn't adversely affected if you do. This configuration has the added advantage of enabling BOOTP clients to function correctly even if they roam across multiple subnets or if the network is renumbered. The disadvantage of dynamic BOOTP is that the DHCP server must permanently reserve any IP addresses it assigns to BOOTP clients because BOOTP does not include any mechanism for automatically recovering IP addresses that are no longer in use.

NOTE

The ISC DHCP server provides some ways of working around BOOTP's inability to provide for automatic IP address reclamation. The first of these is the `dynamic-bootp-lease-length` parameter, which you can set to some length shorter than infinity. To use this parameter safely, you must be able to assume that any BOOTP client on the network is always powered off and back on again at least once during the specified interval. For example, if your site is an office that operates Monday through Friday, and all network devices in the office are always powered off on weekends, you can set this interval to one week and be sure that no BOOTP device retains its address for that long.

The ISC server also provides the `dynamic-bootp-lease-cutoff` parameter. This parameter specifies a specific date when all BOOTP leases end. You can use this parameter, for example, in a college dormitory environment, where it is known that all students leave the dorm on a certain day. If the cutoff is set to the next day, all IP addresses assigned to students in the dorm using BOOTP are reclaimed at once. Of course, it's important to reset this cutoff when it expires!

Separate DHCP and BOOTP Servers

Another strategy for supporting BOOTP devices is to leave the BOOTP server unchanged and provide service for all BOOTP devices through the BOOTP server. BOOTP and DHCP servers can interoperate on the same network or subnet; however, you must run the two servers on separate computers. Therefore, if you use this strategy, you must manage two servers. In such a scenario, making changes to the network topology is complicated because you must update both servers. If the number of legacy BOOTP devices is extremely large, or if for other reasons you do not want to move the BOOTP service to the DHCP server, you can leave the BOOTP server intact and configure the DHCP server not to respond to BOOTP requests.

NOTE

Whichever approach you select, be very careful not to assign IP addresses to a DHCP dynamic pool if they are either in use by a BOOTP server or reserved for a BOOTP device in some other scope. This can result in a duplicate IP address assignment.

Configuring Lease Lengths

DHCP lease length is a topic of heated debate among network administrators. Some networks run DHCP lease times of one year; some use 30 seconds. Generally speaking, the *right* lease time depends on the network's characteristics and performance requirements. However, you should consider a few relevant issues that, when weighed accordingly, can influence your decision about your optimal DHCP lease time.

Examples of Long and Short Lease Times

This section illustrates the pros and cons of long and short lease times by providing an example that shows two hypothetical networks that use extremely long and short DHCP leases: one with a lease duration of one year and the other with a lease duration of one minute.

A Network with One-Year Leases

On Day 1, several DHCP computers and devices start up in this network. The first, a DHCP-configured laser printer, sends two DHCP messages to the server to obtain its IP address. This printer never moves and is never powered off, so it keeps its address, and people use it via its IP address. After half of the lease expires, six months after Day 1, the printer renews its lease by sending a message to the DHCP server. Six months later, it renews its lease again. The DHCP server processes a total of four messages from the printer in the course of one year.

The second device, a DHCP-configured workstation, also starts up on Day 1. It receives its DHCP lease and is fully functional on the network. Each night, this device is turned off. Every morning, it is turned back on and sends one DHCP message to the server to confirm its IP address and to extend its lease on the address. The DHCP server receives one DHCP message per day from this device.

The third device, a mobile laptop, starts up in one subnet on Day 1 and sends two messages to the DHCP server to obtain an address with a one-year lease. The next day, it starts up in a different subnet and gets another address with a one-year lease. The third day, it starts up in a third subnet and gets yet another address with a one-year lease. The laptop, therefore, has an address with a one-year lease on each subnet to which it has been connected.

When the administrator changes a parameter for DHCP clients, the server returns that new value to DHCP clients in the next message it sends to each client. However, because the protocol relies on the *client* to initiate communication, any change to a DHCP option is not sent to a DHCP client until the client sends a message to the server. Therefore, on this network, the printer doesn't see a change to a DHCP option for six months. The workstation and desktop system receive the new parameter value more quickly because they contact the server each time they are powered on, but even 24 hours might be too long to wait for changes to DHCP options to propagate to DHCP clients.

Long leases can also lead to exhaustion of the pool of addresses that are available for dynamic assignment by the DHCP server. Because the server must wait until the lease expires before reassigning an address to a new client, every device that connects to the network—even a laptop that is connected for only a few hours—is assigned an address that can't be reused for one year. Over time, many addresses may no longer be in use but may not be reused because the leases for those addresses have not expired.

When subnets are assigned new IP addresses, or are *renumbered*, another problem can appear. The laptop and workstation are renumbered according to the new subnet address as soon as they restart. The printer, which is left powered on all the time, is not assigned an IP address on the new network until it extends the lease on its address, which could be as long as six months after the network is renumbered.

LONG LEASE TIMES: BENEFITS AND PROBLEMS

The following are the benefits of long lease times:

- Address assignments are stable.
- No renumbering is needed.
- DHCP packet traffic is low.
- The impact of DHCP server outages is limited (see Chapter 15, "Configuring a DHCP Server").

The following are the problems related to long lease times:

- Leases don't expire, and address allocation pools can be depleted.
- Changes to DHCP option values aren't propagated quickly.
- Networks can't be renumbered automatically.

A Network with One-Minute Leases

On Day 1, several DHCP computers and devices start up in this network. The first, a laser printer that uses DHCP, sends two messages to the DHCP server to obtain an IP address. As in the previous example, this printer never moves and is never powered off. After 30 seconds (half of the lease interval) pass, the client sends the DHCP server a request to renew the lease. This happens every 30 seconds, meaning that in any given day, the DHCP server processes 2,880 lease renewals for this printer. Over the course of one year, the server processes more than one million requests from this printer.

The second device, a DHCP-configured workstation, also starts up on Day 1. It receives its address by sending two messages to the server, and it is fully functional on the network. The workstation also renews its lease every 30 seconds while it is powered up and on the network. Each night, this workstation is turned off. The workstation's lease on the address expires one minute after it is turned off. Every morning, when the workstation is turned back on, it obtains a new address from the server because the lease on its previous address has expired. Because the lease has expired, the server can give to another client the IP address that was previously assigned to the workstation. The workstation might or might not get the same IP address each day.

The third device, a mobile laptop, starts up in one subnet on Day 1 and gets a one-minute lease. While on this subnet, it renews its lease every 30 seconds. The lease expires one minute after the laptop is turned off. The next day, the laptop starts up on a different subnet and gets another one-minute lease. The third day, the laptop starts up on a third subnet and gets yet another one-minute lease. Because the leases are short, they always expire before the laptop connects to a new subnet, and the client has only one lease at any given point. While the client is connected to the network, it renews its lease every 30 seconds, by sending a message to the DHCP server.

Suppose the administrator of this network changes a network parameter. The server passes that new value to DHCP clients in the next packet it sends to each client. Because each client sends a message to the DHCP server to extend the lease on its assigned address every 30 seconds, any change to a network parameter is delivered to the DHCP clients almost immediately.

Also, because the DHCP client's lease expires soon after the client is turned off, leases for unused addresses on each subnet do not become stale. Thus, as more and more devices are added to subnets in this network, the likelihood of running out of IP addresses in a dynamic pool is small.

When subnets are renumbered in this network, the computers are renumbered within 30 seconds if they are connected to the network or the next time they are turned on.

SHORT LEASE TIMES: BENEFITS AND PROBLEMS

The following are the benefits of short lease times:

- Changes propagate to the DHCP clients quickly.
- DHCP dynamic pool depletion is unlikely.
- When the network is renumbered, clients get addresses on their new subnets very quickly.

The following are the problems related to short lease times:

- Clients' IP addresses may change frequently.
- Leases expire overnight.
- Short lease times can cause a heavy load on the DHCP server.
- DHCP service must be highly available because even a short outage terminates network service for all clients.

One Lease per Client

As described in the section "A Network with One-Year Leases," scope depletion in a network is possible when you use long lease times. This happens because the DHCP server grants multiple leases to a single client as it moves across subnets.

You can configure the ISC DHCP server to enable only one assigned address for each DHCP client. When this feature is enabled, the server terminates any existing leases when it assigns a new address to a DHCP client. As a laptop roams across subnets, it has only one lease at a time, preventing the DHCP server from depleting its dynamic pools unnecessarily.

Tradeoffs Between Number of Clients and Lease Length

Generally, it is a bad idea to attach to a network segment more computers than there are available IP addresses. It is possible to share the available addresses with DHCP, as long as only as many computers are turned on at once as you have available addresses. If you have nearly as many computers on a network segment as available addresses, you must keep lease times relatively short to avoid running out of available addresses due to unexpired leases on unused addresses.

USING SHORT-TERM LEASES IN A PUBLIC ACCESS NETWORK

At a recent technical convention, the terminal room had a large table with network connections for attendees to use for their laptop computers. Attendees wandered in and out of the terminal room, connecting and disconnecting their laptop computers. Unfortunately, a limited number of IP addresses were available to assign. The terminal room staff initially chose a lease interval of one hour. The turnover rate of users of the terminal room was quite a bit shorter than this, so the pool of available IP addresses was quickly depleted. When this occurred, new arrivals in the terminal room were unable to get IP addresses when they first connected to the network, but they mysteriously received IP address assignments if they waited long enough. After adjusting the lease interval to 15 minutes, this problem was virtually eliminated.

The Effect of Lease Length on DHCP Server Load

One consideration when deploying DHCP, and when tuning the parameters of the DHCP server, is how a given parameter affects the load on the DHCP server. In the example in section “A Network with One-Minute Leases,” a DHCP server managing 10,000 clients would have to process 300 DHCP messages per second. Although there are DHCP servers that can handle 300 transactions per second, that transaction rate represents a significant load on a DHCP server. In comparison, a lease length of one hour would result in an average load of only five DHCP messages per second, which would place only a negligible load on a DHCP server computer.

The Effect of Lease Length on Reliability

This topic is discussed in detail in Chapter 14, so it’s described briefly here. If you are providing DHCP service on a given network segment using only a single DHCP server, clients cannot renew leases whenever that server is not operating. If the DHCP server might stop operating for an extended period of time (for example, a long weekend while no network support staff are available), it might be wise to choose a lease interval that enables DHCP clients to continue operating if the server fails. Even if the server is very reliable, you may need to shut it down for maintenance from time to time, so the lease interval you choose should be long enough that network users are not inconvenienced by a server outage.

DHCP Leases with DDNS Updates

RFC 2136, "Dynamic DNS (DDNS) Updates," defines a mechanism through which DNS data may be more closely tied to DHCP lease data. Because of RFC 2136, DNS data is itself dynamic and time based. When a new DHCP lease is granted, a DDNS update is performed to change the DNS information for the DHCP client. The details of the process through which a DHCP server can update DNS information are described in more detail in Chapter 11, "DHCP–DDNS Interaction."

If the DHCP client leaves the network and the lease expires, the DHCP server must delete the DNS information for the client. However, if the DHCP client stays on the network, periodically renewing its lease, the information in the DNS servers can remain unchanged. This fact, coupled with the way that changes to DNS are propagated through a network, leads to an obvious recommendation for DHCP lease time in DDNS implementations: The DHCP lease time should be at least twice as long as the longest down time that a DHCP client normally experiences. In most networks, this is some event such as a three- or four-day weekend. A DHCP lease time of eight days ensures that even if the DHCP client shuts down just before it renews its lease (leaving four days and one second on the lease), the lease doesn't expire when the client returns. This prevents unnecessary changes to the DNS namespace.

It might be better, however, to simply provide static IP address allocations for DHCP clients that require a consistent and reliable presence on the network and to configure the DNS records for those clients statically.

DHCP Renew Time

As noted earlier in this chapter, a long lease time on a network leads to a long delay between changes to network parameters and to delivery of the new parameter values to DHCP clients. By default, the DHCP client attempts to renew its DHCP lease after one-half of the duration of the lease expires. However, you can configure the server to send the `T1` option, which overrides the default time the client uses for DHCP lease renewal. Setting this parameter to a low value causes DHCP clients to communicate with the DHCP server more frequently, without running the risk of causing the DHCP lease to expire. The ISC server uses the dhcp-renewal-time statement to configure the value sent in the T1 option.

For instance, suppose you set the DHCP lease time to eight days. Using the default renewal time, a DHCP client renews its lease every four days. If you set the renewal time to 12 hours, all the DHCP clients on the network would renew their leases and get any new parameter values within 12 hours. Of course, this solution doesn't solve the problem of unexpired leases on unused addresses, but it does enable you to reduce the number of unwanted lease expirations by using longer leases while delivering new parameter values to clients quickly.

Customizing Lease Durations

Some DHCP servers enable you to choose different lease intervals for different DHCP clients. If your DHCP server provides this capability, you might want to configure your DHCP server to provide shorter lease lengths to laptop computers than to desktop workstations and still longer lease lengths to devices that are always present and whose IP addresses are unlikely to change. Chapter 16, "Client Identification and Fixed-Address Allocation," describes ways of classifying clients into groups to which you can give different lease lengths based on their particular needs.

Making Adjustments to Lease Durations

As noted previously, if you choose to renumber your network, having a long lease interval can be very inconvenient. One way to resolve this problem without using short leases all the time is to change the DHCP server configuration in anticipation of a network renumbering. If you choose to provide seven-day leases by default, seven days before the day on which you renumber your network, you might shorten the maximum lease duration to six hours. When the network renumbering is complete, you can once again lengthen the lease interval to seven days.

Reaching a Compromise on Lease Length

Using a few simple, rules, you can choose a DHCP lease time and DHCP renew time that best fit your needs:

- Keep the DHCP message rate as low as possible. Although DHCP service doesn't require large amounts of processing and system resources, minimizing DHCP traffic is a good idea.
- If DDNS is deployed, or if you are not providing redundant DHCP service, set the DHCP lease time to a value that is sufficiently large to prevent DHCP leases from expiring over long weekends.
- If you require rapid propagation of changes to DHCP options, use a short DHCP renewal time (`T1` option).

Configuring the Lease Length on the DHCP Server

After you choose a policy for assigning the lease, you must configure the DHCP server to provide appropriate leases to clients. Different DHCP servers handle this task in different ways.

The Microsoft DHCP server enables you to configure a maximum lease length for each scope (that is, each subnet). If a DHCP client doesn't request any particular lease length, it is assigned the maximum. If the DHCP client requests a lease length that is less than the maximum, it is assigned a lease with that duration. If it requests a lease longer than the maximum length, it is assigned a lease of the maximum length.

You can configure the ISC DHCP server in any given scope with minimum, maximum, and default lease lengths. The minimum is provided because many DHCP clients specifically request a particular lease length, and if the administrator uses long lease intervals to increase reliability, this benefit cannot be obtained without requiring such clients to take longer leases. For example, the Microsoft DHCP client always requests a 24-hour lease.

Example 19.1 shows an ISC DHCP server configuration that sets appropriate minimum, default, and maximum lease durations for a network with a single DHCP server that you expect is reliable but that you may need to restart from time to time.

Example 19.1

```
default-lease-time 7200;
max-lease-time 86400;
min-lease-time 7200;
```

Remember that these parameters can be different in every scope. This means that, for example, it is possible to globally declare the maximum lease length to be one day but then provide servers that have static IP address allocations with a maximum lease of one month.

Monitoring the Server

You can use SNMP to monitor the operation of network devices via protocol messages. Any device managed by SNMP has an associated MIB that defines the information on the device that is to be monitored via SNMP.

The DHC working group of the IETF has produced a draft DHCP server MIB, authored by R. B. Hibbs and G. Waters. This section describes the use of SNMP and the DHCP server MIB for gathering operational statistics and server configuration information.

To use the information defined by the DHCP server MIB, you use an SNMP management station to contact the DHCP server and fetch the information about the server operation that is of interest to you. The SNMP management station is configured to poll the DHCP server, gather and correlate information, and, when necessary, generate alarms, email messages, pages, and so on, to inform network administrators of problems with the DHCP server.

Gathering and Using Traffic Statistics

Traffic and usage statistics are useful parameters to monitor. Significant increases or decreases in traffic on a given server generally indicate a change or an event that the system administrator should investigate. The DHCP server MIB defines several objects that you can use to determine traffic statistics. Most notably, you can compute the total number of DHCP and BOOTP packets the server receives and sends by adding the total count numbers for each message type. The MIB also includes information about message response time, and this helps you understand how any load on the server is affecting system performance.

On your network, you might configure the management stations to alert you if the total number of packets you receive is more than 30% higher than the total number of packets sent. You might also choose to be alerted if the total number of packets you receive in a one-hour period exceeds a threshold level. If you know that your DHCP server can handle a packet arrival rate of only 50 packets per second without excessive latency, seeing a packet arrival rate of 1,000 packets per hour indicates that you are likely to have a serious problem with your DHCP service. If the message response time exceeds 3.5 seconds, you might also generate an alarm because the server is nearing the maximum response latency that the protocol enables.

NOTE

The ISC DHCP server generates an entry in its log file for every DHCP message it processes. You can easily extract information about the activity levels in the server by parsing the log file with a program such as a Perl script.

Verifying Server Configuration

It should be obvious that server configuration is *critical* to reliable DHCP service. In a dynamic network, it is especially critical. Changes to subnets and to the DHCP server must occur simultaneously. Any inconsistency between the physical network and the DHCP server's view of the subnets and netmasks on the network results in a failure of DHCP service.

The DHCP server MIB provides a number of server configuration objects that give specific details about the configuration state of the server. The MIB includes information about subnets and subnet masks, secondary (equivalent) subnets, dynamic pool ranges, dynamic pool usage, static addresses, dynamic addresses, and reserved addresses. This information enables the SNMP management station to view the entire server configuration.

In a real-world network, the SNMP management station should either *know* or *discover* information about the subnets on a network. You can then configure the management station to compare what *it* thinks about the network with the DHCP server's list of subnets and masks and to generate an alarm of some sort if they are mismatched. This is much better than the old method of waiting for someone to complain that he or she can't get a lease!

DHCP Address Pool Depletion

The most common error a properly configured DHCP server encounters is DHCP dynamic pool depletion. Imagine the following scenario: Your company is having a training seminar in a large room, with a network hub on Subnet A. One of the organizers of the seminar brings in a couple of Ethernet hubs and daisy-chains two 12-port hubs from one Ethernet port. Then, 20 people bring their laptops into the room, plug them in, and try to get DHCP leases in Subnet A (they all normally work in a different building, hence on a different subnet). Suddenly, 20 more devices than expected are in Subnet A. Unfortunately, only 19 leases are available on that subnet. Someone doesn't get an address and calls the help desk.

The DHCP server MIB enables a method to automatically catch occurrences such as this. As mentioned previously, some MIB objects define the DHCP dynamic pool ranges, as well as the usage of a dynamic pool. You can configure the network management station to generate an alarm if a dynamic pool falls below, for example, 15% availability so that problems such as the one described here are discovered before they escalate.

Monitoring Address Pool Usage

You can check the status of any scope that the Microsoft DHCP server manages through the DHCP Manager application, as described in Chapter 13, "The Microsoft DHCP Server." To determine the number of available addresses in a pool managed by the ISC server, you must have a program that determines the active leases in the pool from the leases file and the number of addresses in the pool from the configuration file. You subtract the number of active leases from the total available addresses to find the number of available addresses. Several Perl scripts are available that perform this task.

Summary

Effective deployment of DHCP service requires careful consideration of the current network architecture and device configurations. You can choose from several alternatives when you move to DHCP. Probably the best choice is to use DHCP for all your networked devices. If that strategy isn't feasible, you can accommodate manually configured devices by careful configuring your DHCP service. You can also benefit from some of the advantages of DHCP in configuring legacy BOOTP devices by integrating your DHCP and BOOTP services.

The choice of address lease lengths and the mobility patterns of your networked devices have a significant effect on the behavior of DHCP clients and on the computing resources that a DHCP server consumes. The essential trade-off is between flexibility and processor load: Shorter leases decrease the delay in reclaiming addresses and getting new configuration parameters to clients, at the expense of increased load on the computer that is providing the DHCP service. There is no single answer to the question “How long should my DHCP leases be?”; DHCP lease lengths should be selected according to the structure of your network and the way in which computers are used on your network. In fact, you might want to choose different lease lengths for different parts of a network.

20

Programmable DHCP Server Customization

IN THIS CHAPTER

- Differentiating Between Clients
- Controlling Address Allocation
- Client Class Options
- Lease Events
- Lease Variables

A few of the preceding chapters discuss how to set up the DHCP server so that it configures all clients with essentially the same parameters. They also show some simple ways to specify whether the server provides addresses to specific clients or to clients the server doesn't know about. To configure computers on the network without DHCP, you must manually configure each computer individually. Although this is a tedious process, it gives you one advantage: You can customize each computer's configuration based on its user's needs rather than configure all systems identically.

This chapter describes how to program the ISC DHCP server so that it can perform many of the same customizations you might perform when configuring clients manually, as well as some automatic customizations that are difficult to perform manually. It also shows how to classify clients more accurately, and it describes the capabilities of the ISC DHCP server class construct. You will learn how to set up different address allocation pools so that you can use different address allocation strategies for different clients. In addition, you will learn how to send different parameters or use different settings for specific clients, different classes of clients, and clients whose addresses come from different pools.

For the most part, this level of customization is specific to the ISC DHCP server version 3.0 and higher. Customizations that are also possible for the Microsoft Windows 2000 DHCP server are noted where appropriate.

Differentiating Between Clients

The DHCP server differentiates between clients based on what each client sends to the server in its messages. Chapter 16, "Client Identification and Fixed-Address Allocation," discusses how the DHCP server identifies a specific client by using either the DHCP `client identifier` option or the client's link-layer address.

The DHCP server can also make decisions about a client based on other options the client sends or on the other contents of the client's packet. Unlike the `client identifier` option and link-layer address, the other contents of a DHCP packet generally aren't guaranteed to be unique. Configuring the server to use them to differentiate between clients enables you to control the server's behavior with respect to classes of clients, rather than just to specific individual clients.

Conditional Statements

The simplest way to differentiate between clients is to use conditional statements. *Conditional statements* work by testing some logical expression; if the result of evaluating that expression is `true`, the statements that are enclosed in the conditional statement are executed (for example, some special configuration parameters or options might be defined). If the result of evaluating an expression is `false` or null, the statements are not executed.

NOTE

You can use many different kinds of expressions to determine what sort of client sent a request. Appendix B, "ISC DHCP Server Configuration File Reference," provides a complete list of the different kinds of expressions that the DHCP server recognizes.

If you provide an `else` clause, the parameters or options within the `else` clause are defined. You can chain together a series of `if` statements by using the `elsif` statement as well.

Consider an installation where IP addresses are in somewhat short supply, so the administrator needs to configure the DHCP server not to allocate leases for longer than they will be used. At this site, all site-owned computers are registered with the DHCP server by using `host` declarations. The site owns a number of printers that support TCP/IP. These printers send the vendor class identifier `acme-printers`. The site administrators don't register these printers because they're easy to identify. The site also provides IP addresses to visiting mobile clients. Example 20.1 shows how to give long leases to the printers, shorter leases to site-owned machines, and very short leases to visiting mobile clients.

Example 20.1

```
if known {
    max-lease-time 18000;
    default-lease-time 18000;
} elsif option vendor-class-identifier = "acme-printers"
    max-lease-time 90000;
    default-lease-time 90000;
} else {
    max-lease-time 300;
    default-lease-time 300;
}
```

In Example 20.1, the `known` expression is used to test whether a client has a `host` declaration. For clients that have `host` declarations, the default lease time is 18,000 seconds; the maximum lease time is also 18,000 seconds. Clients that don't have `host` declarations but send the `vendor class identifier` option `acme-printers` get longer lease times. All other clients receive much shorter lease times.

NOTE

The ISC DHCP server also provides a `switch` statement, which functions in much the same way as the C `switch` statement or the LISP `cond` statement (the syntax is similar to C, but the semantics are more similar to LISP). This is described in Appendix B.

You can also nest conditional statements. Example 20.2 shows a configuration similar to the one mentioned previously, except that there is a special case for Microsoft RAS servers. RAS servers tend to ask for leases they don't need and then forget they got them, so generally it is a good idea not to give addresses to RAS servers. If you must give an address to a RAS server, it should have a very short lease, so that when the RAS server forgets about it, the address can be quickly reclaimed.

Example 20.2

```
if substring (option dhcp-client-identifier, 1, 3) = "RAS" {
    if not known {
        ignore booting;
    } else {
        max-lease-time 300;
        default-lease-time 300;
    }
} elsif known {
    max-lease-time 18000;
    default-lease-time 18000;
```

Example 20.2 Continued

```
} else {
    max-lease-time 300;
    default-lease-time 300;
}
```

Example 20.2 checks to see whether the second through the fourth byte of the DHCP `client identifier` option is the string `"RAS"`. If it is, the client is a Microsoft RAS server. If it is a RAS server and if no `host` declaration for that particular client exists, it is not given an IP address; the message is simply ignored. If the RAS server has a `host` declaration (that is, it is known), it gets a very short lease. If the `client identifier` option wasn't sent or doesn't begin with `RAS`, but the client is known, it gets a long lease. If the client is not known, it gets a very short lease.

NOTE

You can only use conditional statements to conditionalize option and parameter definitions. You cannot conditionally define a new option because new option definition happens when the configuration file is processed. Likewise, it's not possible to conditionally define hosts, subnets, shared networks, groups, pools, address ranges, leases, or permits because all these declarations are processed when the configuration file is parsed, not when a client request comes in.

Client Classing

The ISC DHCP server provides a construct called a *class* that you can use to group clients in a more general manner than you can with a `host` declaration. Like `host` declarations and `subnet` declarations, classes have *scope*. (See Chapter 14, "The ISC DHCP Server.") You can also use a client's membership in a class to control how addresses are allocated to it.

Clients become members of classes either because they match a class's matching rule or because they match a *subclass* of that class. Class-matching rules are evaluated before any conditional statements that may exist in any scope. Currently, a client can be a member of up to five classes. This limit is arbitrary and might be eliminated in the future.

Class-matching rules are evaluated in the order in which the classes are originally declared. For example, if a client matches two classes based on those classes' matching rules, the scope of the class declared first in the configuration file is considered more specific than the scope of the class declared second.

Class-Matching Rules

Two kinds of class-matching rules exist: *explicit rules* and *subclass-matching rules*. In Example 20.3, if a client sends the `vendor class identifier` option `acme-printers` or `united-printers`, it is considered to be in the class `printers`. If it sends a DHCP `client identifier` option that starts with the string `RAS`, it is a member of the `ras-clients` class. As an example of class-based scoping, members of the `printers` class receive long lease times, whereas members of the `ras-clients` class receive very short lease times. Of course, if a more specific scope overrides the class scope's definition of the `domain-name` option, the particular client receives the option from the more specific scope. The only scopes that are more specific than a class scope are the scope of another class that appears earlier in the configuration file and the scope of a `host` declaration.

Example 20.3

```
class "printers" {
    match if option vendor-class-identifier = "acme-printers" ||
             option vendor-class-identifier = "united-printers";
    default-lease-time 18000;
    max-lease-time 18000;
}
class "ras-clients" {
    match if substring (option dhcp-client-identifier, 1, 3) = "RAS";
    default-lease-time 300;
    max-lease-time 300;
}
```

Subclass Matching

As discussed earlier in this chapter, you can use `host` declarations to group clients as either known or unknown. You can also use this feature to differentiate between clients that are registered with the network administrator and clients that are not. However, you might want to divide clients into more than two sets—those that are registered and those that are not.

For example, consider an ISP that sells broadband service. At each customer site, there is a cable modem. Whenever a DHCP request comes in from a client computer that is attached to the cable modem, the cable modem attaches a `relay-agent-information` option to the request. The cable modem vendor would like to be able to provide different levels of service to some clients than to others, depending on what they have paid for.

Master classes and subclasses provide an efficient method of matching many different possible values that might result from evaluating some expression. Instead of specifying a `match if` statement, as in an explicit `class` declaration, a master class has a `match` statement with an expression that extracts some data from the client's DHCP packet. Each `subclass` declaration refers to its master class by name and includes in its declaration a value for the master class's `match` expression. If the value extracted from the client's DHCP packet matches the value specified in the `subclass` declaration, the client matches the master class and the subclass.

When you're determining membership, if a client is a member of a particular subclass, it's also a member of the master class. Indeed, you cannot check that it is a member of the subclass because the subclass has the same name as the master class. Subclass scopes are considered more specific than master-class scopes. Subclasses are not, however, required to have scopes, and not specifying a scope saves memory.

Example 20.4 shows an example of a `class` declaration and several `subclass` declarations, using the DHCP `client identifier` option to differentiate between clients. One subclass has its own scope; the other two do not.

Example 20.4

```
class "engineering" {
    match option dhcp-client-identifier;
}
subclass "engineering" "amee-pc";
subclass "engineering" 1:00:00:ac:42:a9:a6 {
    option root-path "server1.example.com:/var/clients/root-netbsd";
}
subclass "engineering" "nilo1";
```

As you can see from this example, it is possible that some matching data will be in the form of an ASCII string and some will be in the form of a sequence of hexadecimal bytes separated by colons. You might have noticed that `subclass` declarations look very much like `host` declarations. Indeed, in many cases you can use them in exactly the same way. However, you can't specify a fixed address in a `subclass` declaration, and, as Chapter 16 describes, the method a DHCP server uses to determine a client's identity is more complicated than a simple pattern match.

Controlling Address Allocation

You can control address allocation in two ways:

- You can group one or more ranges of addresses into a pool declaration and then control access to those pools by using permit lists. These permit lists can be based on the client's class, known/unknown status, or protocol.

- You can specify a limit to the number of leases that members of a particular class can hold at any one time. You can configure the DHCP server to automatically create new classes and apply lease limits to them.

You can also combine these two mechanisms.

Pool-Based Address Allocation

An *address allocation pool* is a group of addresses that can be allocated on a particular network segment. A pool can contain addresses from more than one subnet, as long as all the subnets are configured on the same network segment.

Allocation Based on Class Membership

Example 20.5 shows clients from different classes grouped onto different subnets that are configured on the same network segment. The first pool has a permit list that enables only clients in the `engineering` class to be assigned addresses. The second pool's permit list enables only clients in the `sales` class, and the third pool prohibits members of the `engineering` and `sales` classes from being assigned addresses, which means all other clients will be assigned addresses from that pool. A pool's permit list is the set of all `allow` and `deny` statements within the `pool` declaration.

Example 20.5

```
shared-network BUILDING-1-LAN {
    subnet 10.227.109.0 netmask 255.255.255.0 {
        option routers 10.227.109.1;
        option domain-name-servers 10.227.109.2;
        pool {
            range 10.227.109.10 10.227.109.254;
            allow members of "engineering";
        }
    }
    subnet 10.227.110.0 netmask 255.255.255.0 {
        option routers 10.227.110.1;
        option domain-name-servers 10.227.110.2;
        pool {
            range 10.227.110.10 10.227.110.254;
            allow members of "sales";
        }
    }
    subnet 10.228.0.0 netmask 255.255.255.0 {
        option routers 10.228.0.1;
```

Example 20.5 Continued

```
        option domain-name-servers 10.228.0.2;
        pool {
            range 10.228.0.10 10.228.0.254;
            deny members of "engineering";
            deny members of "sales";
        }
    }
}
```

Example 20.5 shows pools used to contain address ranges on different subnets. However, it is also possible to have more than one allocation pool on the same subnet. For example, you might want to reserve a small pool on each subnet on a network for unregistered clients and reserve the rest of the IP addresses for registered clients.

Other Allocation Controls

In addition to permitting or denying access to a particular pool based on the class to which a client belongs, you can permit or deny pool access in several other ways. You can permit allocation based on whether a client has a host declaration, by using the `allow known clients` directive or the `allow unknown clients` directive. You can also permit or deny allocation of addresses to BOOTP clients by using the `allow dynamic bootp clients` directive. Finally, it is possible to explicitly permit or deny all access to a pool by using the `allow all clients` directive or the `deny all clients` directive.

Class-Based Address Allocation

Sometimes it is useful to restrict the number of addresses assigned to a class of clients without reserving a specific pool of addresses for the members of that class. Reserving a pool requires either that the maximum number of addresses permitted to each class is reserved at all times or that the pool of available addresses for all permitted classes is shared in an uncontrolled manner.

For example, consider the broadband ISP mentioned earlier in this chapter. This ISP has one large network segment that serves all client sites. (In practice, the network might be segmented, but we'll assume that the network has a single network segment for simplicity.) Each site has a cable modem that accesses the network by exchanging packets with a head-end system at the ISP. Whenever a DHCP client at a client site broadcasts a DHCP request, the cable modem relays that request to the central DHCP server after attaching a `relay agent information` option that contains the `circuit ID` and `remote ID` agent options. The remote ID is always unique to the customer site.

This cable modem provider might want to enable customers to obtain more than one IP address, but it does not want to enable customers to obtain an unlimited number of IP addresses. The provider can do this by declaring a subclass for each client site, based on the contents of the `relay agent information` option, and by specifying a limit on the number of leases that members of that class can obtain. The fact that the `relay agent information` option is appended to every client packet relayed from a particular customer site and is the same for every host at that site restricts the number of leases that DHCP clients at that site can obtain. Example 20.6 shows how this can be done.

Example 20.6

```
class "customer-sites" {
    match option agent.remote-id;
    lease limit 1;
}
subclass "customer-sites" 0:42:77:a9:c6;
subclass "customer-sites" 17:a8:ee:0:0;
subclass "customer-sites" f7:aa:90:1c:2b;
subclass "customer-sites" 27:c9:45:12:a0;
subclass "customer-sites" 99:91:a0:1c:22;
```

In Example 20.6, the lease limit is specified as `1` in the master class. The subclasses inherit the lease limit from the master. This means that only one member of each subclass can hold a lease at any one time. One IP address is fine for a normal home subscriber, but suppose the ISP in our example sells to businesses as well, using the same distribution system. A business site might need more than one globally routable IP address. In this case, the ISP might write a special `subclass` statement for that business, specifying a larger limit, as shown in Example 20.7.

Example 20.7

```
subclass "customer-sites" 99:91:a0:1c:22 {
    lease limit 8;
}
```

NOTE

If a client is a member of a class that limits the number of leases allocated to it but is also a member of a class that doesn't, it can get an IP address only through the class that limits lease allocation. If a client is a member of two classes that limit lease allocation and one is full but the other is not, its lease is allocated from the class that is not full.

NOTE

The `lease limit` feature of the ISC DHCP server depends on the DHCP server having a clear idea of how many leases are allocated to a particular class. If you are using the failover protocol, there is no way for this to happen; depending on which leases are renewed on which servers, it is possible that twice as many leases will be allocated as you intend. It is also possible that exactly the number of leases you intend will be allocated. Therefore, you cannot implement a reliable lease limiting system with failover. It might be possible to modify the ISC DHCP server to do a better job, but at the time of this writing, nobody is doing so.

Automatic Generation of Subclasses

To further automate the process of setting lease limits at customer sites, it is possible for the ISC DHCP server to create subclasses automatically, based on the matching expression in the master class. These classes are then recorded in the lease database so that they persist across restarts of the DHCP server. This is a handy way to get the address allocation limiting described earlier without requiring anybody to, for example, register specific cable modem remote IDs with the DHCP server. Example 20.8 shows an example of a master class that automatically spawns subclasses.

Example 20.8

```
class "customer-sites" {
    spawn with option agent.remote-id;
    lease limit 1;
}
```

In Example 20.8, whenever a request is received from a client that contains the `relay agent information` option and a `remote ID` suboption, the server looks up that suboption in the class's hash table. If that suboption is present, the existing subclass is used. Otherwise, a new subclass is allocated and recorded in the lease database, and that new class is used to account for leases in use at that site.

NOTE

Spawning classes can be combined with predefined subclasses because the `spawn with` and `match with` statements both have the same effect if the class they match already exists. To do this, you use `spawn with` instead of `match with`, as in Example 20.6.

Differentiation Between Similar Sets of Subclasses

In some cases it might be helpful to use spawning classes as described earlier in this chapter but group clients into different sets of subclasses as well. The `spawn with` statement for each class might be identical. When this is case, you can use a `match if` statement along with the `spawn with` statement. For example, suppose the broadband ISP we've been using as an example sells two different kinds of circuits, with different circuit IDs, and the first 4 bytes of the circuit ID are the speed, in kilobits per second. A customer that purchases the 1544Kbps connection is allowed up to eight globally routable IP addresses. A customer that purchases the 256Kbps connection is allowed only one. Example 20.9 shows how this can be configured.

Example 20.9

```
class "slow-sites" {
    match if substring (option agent.circuit-id, 0, 4) = "0256";
    spawn with option agent.remote-id;
    lease limit 1;
}

class "fast-sites" {
    match if substring (option agent.circuit-id, 0, 4) = "1544";
    spawn with option agent.remote-id;
    lease limit 8;
}
```

You can also use the `match if` statement with a `match` statement, when predefined subclasses are being used.

Client Class Options

Two options are defined in DHCP specifically for the purpose of grouping DHCP clients into classes: the `user class` option and the `vendor class identifier` option. The ISC DHCP server doesn't actually treat these options specially, but they have special meanings that relate to client classification. Another class-related option is the `vendor-specific information` option, which is treated specially by the DHCP server.

The `user class` Option

Using the `user class` option, which is defined in RFC 3004, is a very simple way to classify clients based on information that is supplied by the user. The `user class` option contains one or more attributes, each of which can be any arbitrary string of bytes. Because the `user class` option is intended to be specified by the user of the computer, each attribute is normally a human-readable ASCII text string. Most DHCP clients allow the user to specify only a single attribute. DHCP clients can send this string to the DHCP server to specify attributes that apply the user who is using the machine. For example, if you support a sales department, an engineering department, and a support department, you might configure sales machines to report a `user class` attribute of "`sales`", engineering might report "`eng`", and support might report "`support`". You can then use this information to classify the client, as shown in Example 20.10.

Example 20.10

```
class "sales" {
    match if option user-class = "sales";
}
class "engineering" {
    match if option user-class = "engineering";
}
class "support" {
    match if option user-class = "support";
}
```

NOTE

When it was originally defined, the `user class` option allowed the user to specify only a single attribute. Several widely deployed DHCP clients, including the Microsoft and ISC clients, support the old single-attribute `user class` option. Example 20.10 shows the single-attribute `user class` option. Although RFC 3004 is a standard, we can't show you an example of a multi-attribute `user class` option yet because none of the servers we have access to implement the RFC 3004 `user class` option.

For this option to be useful, you must configure it into each DHCP client at your site. The presumption is that every time a new DHCP-enabled workstation is installed, either the network administrator or the user configures that workstation with some kind of class identifier. Some modern DHCP clients, including those supplied by Microsoft and ISC, provide a way for the user to specify a user class.

The `user class` option violates one important precept of DHCP: that the DHCP server, not the user, configures the TCP/IP configuration. However, in practice, getting users to set their `user class` option is not too hard. Unlike IP addresses, `user class` identifiers are normally human-readable text strings that make sense even to novice users.

The `vendor class identifier` Option

The `vendor class identifier` option is intended for use by DHCP client vendors, to identify the vendor that supplied the DHCP client and, possibly, to provide identification information about the hardware on which the DHCP client is running. The vendor defines the format of the `vendor class identifier` option. Currently, no process exists whereby vendors can ensure that the identifier they choose is unique to them. However, vendors usually use trademarked company names, so conflicts are unlikely. The Sun implementation of the `vendor class identifier` option is used here as an example of how this option works.

Sun specifies vendor class identifiers as ASCII text strings, with information about the client encoded as a series of labels separated by periods. The leftmost label is most general, with labels becoming more specific toward the right. For example, the most general label is `SUNW`, indicating that the computer is a Sun workstation, and a Sun SPARC Ultra 5/10 workstation sends a vendor class identifier of `SUNW.Ultra-5_10`. `Ultra-5_10` encodes the specific hardware type. A PC running Solaris identifies itself with the vendor class identifier of `SUNW.i86pc`.

A workstation with a SPARC processor running Solaris might require somewhat different options than a workstation with a processor from the Intel x86 family running Solaris, and you need to configure it accordingly. Example 20.11 shows how to send a different `root-path` option for these two processor architectures.

Example 20.11

```
if option vendor-class-identifier = "SUNW.Ultra-5_10" {
    option root-path "/export/root/sparc";
} else if option vendor-class-identifier = "SUNW.i86pc" {
    option root-path "/export/root/i86pc";
}
```

Example 20.11 is somewhat simplistic; if you support DHCP clients that are supplied by other vendors, you might end up with a long list of `if`, `elsif`, and `else` statements. Also, you might support several different types of Sun SPARC hardware and might want to specify the same behavior for all these types without listing them all explicitly.

Example 20.12 shows a more general case than Example 20.11. It first checks to see whether the `vendor class identifier` option starts with the string `SUNW.`, and then it checks the next five characters in the option to see what kind of Sun workstation sent the request, using the `switch` statement.

Example 20.12

```
if substring (option vendor-class-identifier, 0, 5) = "SUNW." {
    switch (substring (option vendor-class-identifier, 5, 5)) {
        case "Ultra":case "Sparc":        option root-path "/export/root/sparc";
```

Example 20.12 Continued

```
            break;
                case "i86pc":
                    option root-path "/export/root/i86pc";
                    break;
    }
}
```

This is a very straightforward example of using the `vendor class identifier` option. You can also use the `vendor class identifier` option as a key with which to address and encapsulate a series of vendor-specific options.

The `vendor-specific information` Option

Chapter 6, “The Format of DHCP Messages,” describes the format of DHCP messages and, in particular, the format of the DHCP `options` field. The `vendor-specific information` option is actually a DHCP option whose length is variable. It contains either uninterpreted vendor-specific data or a sequence of option tags, lengths, and option data, in the same format as the `options` field itself.

Example 20.13 shows an example of a `vendor-specific information` option that contains vendor-specific option tags. It starts with the code for the `vendor-specific information` option, `43`, followed by the length, and finally the payload. The payload is a sequence of two vendor-specific options, Options 1 and 2. Option 1 is an IP address (10.0.0.1 in Example 20.13), and Option 2 is a 32-bit number (227883991).

Example 20.13

```
43    12    1    4    10    0    0    1    2    4    13    149    59    215
```

The ISC DHCP server enables you to define vendor-specific option spaces and then define options within those spaces. You can then direct the DHCP server to use option spaces you have defined to generate the `vendor-specific information` option. Option spaces are described in Chapter 14.

A HYPOTHETICAL EXAMPLE OF THE `VENDOR-SPECIFIC INFORMATION` OPTION

Suppose you set up a DHCP server for use in a bank branch. The bank purchased a customized turnkey system for operating each branch; each teller gets a workstation, and a central server runs a couple proprietary protocols. The vendor defines a set of DHCP options used to configure the teller workstations. (Let’s call the vendor ExampleSoft and use the vendor’s name for the option space.) The vendor must send two pieces of information to each workstation: the IP address of the central server and a token that the client workstation uses to identify its location to the central server, which might actually support more than one branch. The two options are called `central-server` and `site-token`, respectively. Example 20.14 shows how these options are introduced to the DHCP server.

Example 20.14

```
option space ExampleSoft;
option ExampleSoft.central-server   code 1 = ip-address;
option ExampleSoft.site-token      code 2 = integer 32;
```

The site also has some networked printers that get their IP addresses by using DHCP and do not require the ExampleSoft vendor-specific options. Meanwhile, the branch manager has in her office a workstation running NetBSD that she uses to exchange e-mail with other branch managers and to prepare presentations. Example 20.15 shows the use of the `vendor-option-space` statement to indicate that the ExampleSoft option space should be used to generate the `vendor-specific information` option, but only if the `vendor class identifier` option the client sent consists of the string `ExampleSoft`. This example also shows two subnet declarations for different branches that are served by the DHCP server and by the ExampleSoft branch server. It also shows how vendor-specific options can vary by scope, just like any other options.

Example 20.15

```
if option vendor-class-identifier = "ExampleSoft" {
    vendor-option-space ExampleSoft;
}
option ExampleSoft.central-server 10.0.0.1;
option domain-name-servers 10.0.0.11, 10.0.0.12;
subnet 10.0.10.0 netmask 255.255.255.0 {
    range 10.0.10.10 10.0.10.200;
    option routers 10.0.10.1;
    option domain-name "101-main.photon-bank.com";
    option ExampleSoft.site-token 227883991;
}
subnet 10.0.11.0 netmask 255.255.255.0 {
    range 10.0.11.10 10.0.11.200;
    option routers 10.0.11.1;
    option domain-name "250-university.photon-bank.com";
    option ExampleSoft.site-token 198283471;
}
```

When a client sends a request that includes a `vendor class identifier` option whose value is `ExampleSoft`, the DHCP server iterates through the in-scope definitions of all the options in that class to create a vendor-specific option buffer that contains all the options. It then encapsulates that option buffer as the payload of the `vendor-specific information` option. The sequence of bytes shown in Example 20.16 is the `vendor-specific information` option that is returned to one of the teller workstations that is connected to the 10.0.10.0 subnet.

Lease Events

The ISC DHCP server provides a way to specify actions to take when certain events happen with respect to leases. These events are the `commit` event, the `expiry` event, and the `release` event. These events can be used for a variety of purposes, including for custom DNS update support and event logging. Example 20.16 shows how to cause the DHCP server to log a message when a client's lease expires.

Example 20.16

```
on expiry {
  log (info, concat ("Lease ",
                     binary-to-ascii (10, 8, ".", leased-address),
                      " has expired.")));
}
```

Lease Variables

It is possible to store variables on leases. Each lease has its own variable namespace, so, for example, two leases could both have a variable named `my-mx-name`. The DNS update code uses lease variables to record information about the DNS update. Example 20.17 shows how to add a Mail Exchanger (MX) record to the DNS whenever the DHCP server successfully updates the DNS.

Example 20.17

```
on commit {
  if (exists (ddns-fwd-name) and (not exists (ddns-mx-name))) {
    switch (ns-update (add (IN, MX, ddns-fwd-name,
                             "mail.fugue.com", lease-time / 2))) {
      case NOERROR:
        set ddns-mx-name = ddns-fwd-name;
        on expiry | release {
          eval ns-update (delete (IN, MX, ddns-mx-name));
          unset ddns-mx-name;
        }
    }
  }
}
```

In Example 20.17, the configuration file contains executable code for the `commit` event. The `commit` event happens whenever a lease is committed. When the `commit` event occurs on a lease, this `commit` event code checks to see whether the variable ddns-fwd-name exists. If it does, this indicates that the DHCP server has successfully done a DNS update for this client. The commit code also checks to see whether it has already added an MX record, by checking the value of the ddns-mx-name variable. The commit event happens every time the lease is renewed, not just the first time, so if the commit code didn't check, it would re-add the MX record whenever the lease was renewed.

If the commit code hasn't yet added an MX record, it tries to add it. The code in Example 20.17 tests the result of the update by using the switch statement. If the result is NOERROR, meaning that the update succeeded, the commit code sets the ddns-mx-name variable. This allows it to remember to remove the MX record later and also not to keep trying to add it after it has been added for the first time. The commit code then adds code to execute when the lease expires or is released, using the expiry event and the release event. The expiry and release event code deletes the MX record when the lease expires or is released. This event also unsets the ddns-mx-name variable because if it is not unset, it remains on the lease even though the lease has expired.

Summary

The ISC DHCP server provides a powerful collection of tools you can use to differentiate between clients. You can instruct the server to return different options to different clients based on what the clients send.

Options can be grouped into classes. You can do this either by testing for options that many clients send, by having each client's scope determine its class (for example, with host declarations), by predefining subclasses, or by configuring the server to automatically spawn new subclasses. You can control address allocation policy based on the classes of which clients are members, as well as whether clients have host declarations and whether they use BOOTP or DHCP. You can also limit the number of members of a particular class that can hold a lease at the same time. To automate per-class lease limits, you can configure the server to generate new subclasses as it runs.

Several options are designed specifically for use in classifying clients. Clients can be configured to send the user class option to indicate what class of user is using them, although the user class option is unfortunately not widely supported. Some vendors' clients also use the vendor class identifier option to identify the vendor that sent the option. The vendor class identifier option can also be used to determine the contents of the vendor-specific information option.

You can program the server to take action when various events happen in the lifetime of a lease and to store variables on the lease for use when events occur.

21

DHCP Clients

IN THIS CHAPTER

Previous chapters describe in great detail DHCP and various aspects of how DHCP servers work and are configured. This chapter describes the following:

- How a DHCP client works
- Some specific DHCP clients and what is unusual about them
- Some of the problems you might have when using or providing DHCP service for clients

The Theory of DHCP Client Operation

This chapter uses the ISC DHCP client as a basis for describing the theory of operation of DHCP clients; the ISC DHCP client is the most general of the several open-source DHCP clients that are available. This chapter then goes on to describe the operation of a variety of DHCP clients.

The ISC DHCP client runs as a daemon process; that is, it is started when the computer first starts up, and it continues to run as an independent process until the computer is halted. Other clients may use a different implementation strategy. The DHCP client drives the DHCP protocol; it has an internal state machine that makes transitions as the client moves through the protocol. The DHCP client finite state machine is illustrated in Figure 8.1 and described in more detail in Chapter 8, “DHCP Message Exchanges.”

Getting an IP Address

When the client is first started, it reads in a database of old leases, examines each lease in the database, and discards each lease that has expired. If any leases remain when this process is complete, the client chooses the lease that was acquired most recently and enters the `INIT-REBOOT` state. In this state, the client broadcasts a `DHCPREQUEST` message to ensure that the address in the selected lease is still valid for the network segment to which the host is connected. If the client gets no response, it re-sends the `DHCPREQUEST` message. If the client receives no response after a certain period of time elapses, it tries to use the IP address in the lease.

When they fail to confirm their lease during the `INIT-REBOOT` state, most DHCP clients simply use the IP address they tried to confirm. If the ISC DHCP client fails to confirm its lease by using a broadcast `DHCPREQUEST` message, it tries to confirm that the address is appropriate for the subnet to which it is attached. It does this by configuring its network connection with that address and sending an ICMP echo request to its default router. If it gets a response, it uses the old address; otherwise, it tries to determine whether any of the other unexpired addresses work on the current subnet. If the client can't contact the default router with any of its unexpired addresses, the client reverts to the `TIMEOUT` state.

As DHCP is specified in RFC 2131, a client is expected to start in the `INIT` state if it has no unexpired leases. Many clients implement this behavior and immediately broadcast a `DHCPDISCOVER` message when starting with no unexpired leases. Some implementors have interpreted the text in RFC 2131 to permit a client to first try to renew an expired lease and then revert to the `INIT` state if the renewal fails. This behavior is widely agreed to be incorrect, but some older clients do operate this way.

If the ISC client receives no response to its broadcast message in the `INIT` state, it continues retransmitting `DHCPDISCOVER` messages for some configurable period of time (usually 60 seconds). If it doesn't find a valid lease within that time period, it reverts to the `TIMEOUT` state.

When the Client Fails to Get an Address

The `TIMEOUT` state mentioned in the preceding section is not included in the protocol specification, but many DHCP client implementors added it (or something like it) to their state machines. The purpose of the `TIMEOUT` state is to enable the client to "give up" and notify the user in some way that it failed to acquire an IP address.

When the ISC DHCP client is started, it runs in the foreground until it has either found an IP address or entered the `TIMEOUT` state. This enables daemons that are started after the client is started to assume that the network is configured before they begin, but it prevents the client from holding up the system startup indefinitely. After the purpose of the `TIMEOUT` state is fulfilled, the client waits for a configurable interval and then reverts to the `INIT` state.

If the Microsoft or Apple DHCP client reaches the `TIMEOUT` state, it uses Automated Private IP Addressing (APIPA) to assign an address to the interface that is being configured. When using APIPA, the DHCP client chooses an IP address out of the

65,534 possible IP addresses in the 169.254.0.0/16 subnet. It then sends an ARP request to see whether the address is in use; if the address is in use, the client chooses a different address from the same subnet at random. It continues to do this until it finds an address that isn't in use, and then it configures that address. After the client selects an APIPA address, it periodically attempts to use DHCP to get an IP address. If it finally contacts a DHCP server, it stops using the autoconfigured IP address.

NOTE

IANA has reserved the 169.254.0.0/16 subnet for host autoconfiguration. APIPA is one example of an autoconfiguration mechanism that uses the 169.254.0.0/16 subnet. Autoconfiguration addresses are restricted in scope to a single physical network; routers are never allowed to forward IP datagrams with a source address from the 169.254.0.0/16 subnet to a different network. Autoconfiguration addresses are intended to automate IP address assignment without coordination by a central authority such as a network administrator in a network with no DHCP server.

Using an IP Address After It Is Acquired

After the DHCP client acquires an IP address, it enters the `BOUND` state. When the client initially enters this state, it configures the network interface with the new IP address. If the server provides a default route, the client installs that route. The ISC DHCP client also uses the `domain-name` option (if the server provides one) and the `domain-name-servers` option to create an `/etc/resolv.conf` file. This is a standard configuration file on most Unix and Unix-like systems that is used to configure domain name service.

In addition to configuring the network interface, when the DHCP client enters the `BOUND` state, it also makes a persistent record in a disk file of the lease it acquired. This record includes any DHCP options the server sends; you can use that lease information later if the client is restarted.

Maintaining a Lease on an IP Address

After the DHCP client acquires a lease, it must maintain it. When half of the duration of the lease expires, the client enters the `RENEWING` state. In this state, it periodically unicasts a `DHCPREQUEST` message to the server that assigned it its lease, requesting an extension. If it gets a response to this request that extends its lease, it reenters the `BOUND` state until half of the duration of the new lease once again expires. The client need not reconfigure the network interface when it renews its lease, but it does make use of any changes in the options that the server sends—for example, the default route and the DNS configuration. The new lease is also recorded on disk for later use.

NOTE

Some older DHCP clients do not notice changes to parameters in `DHCPACK` messages when they are in the `RENEWING` state. This is not correct behavior, but if you are having trouble getting a DHCP client to obtain new configuration parameters, it might have this problem. Telling the client to release and then renew its lease often clears up the problem, although in some very old DHCP clients, even that does not work; in some versions of Windows 95, you actually have to reinstall TCP/IP in order to clear out the old parameters.

If seven-eighths of the lease time expires and the client still receives no response, the client enters the `REBINDING` state and begins broadcasting `DHCPREQUEST` messages to locate some other DHCP server that will renew its lease. If it receives a `DHCPACK` message in the `REBINDING` state, the client handles the `DHCPACK` message just as it would in the `RENEWING` state.

When the Lease Expires

If for some reason the client is unable to renew its lease before the entire duration of the lease expires, the client must stop using the IP address it has leased. The client deletes the IP address from its interface configuration and goes back to the `INIT` state to try to get a new address.

Multiple Network Interfaces

The discussion in the preceding sections assumes that the client can operate on only one network interface. This is not far from the truth; if more than one network interface must be configured, the DHCP client program sets up a separate state machine for each interface. These state machines operate independently from one another; one interface can be in the `BOUND` state, while another interface is in the `INIT` state, for example.

As far as IP address configurations go, this state of affairs works perfectly well; each interface is configured with an IP address as soon as a server provides one. Unfortunately, with respect to DHCP options, if two interfaces get different configurations, no standard mechanism exists that says how a client should decide which set of options to use. For options that are specific to an interface, such as a subnet mask, this is not a problem, but for other options, such as the `routers` option and the `domain-name-servers` option, there simply is no way to choose. Most DHCP clients simply use the value they received most recently. MacOS 9 solves this problem by not permitting you to use two interfaces at the same time.

More Than One IP Address Per Interface

Occasionally, users ask for a mechanism for obtaining more than one IP address for a single network interface. The ISC DHCP client included in Version 3.0 of the ISC

DHCP distribution supports this capability. A pseudo-interface can be declared and associated with a real interface, and the DHCP client allocates a separate state machine for that pseudo-interface.

You must give the pseudo-interface a different DHCP client identifier than is sent for the primary interface, or the DHCP server cannot differentiate between the two requests, and it assigns the same IP address to both interfaces. The standard DHCP client does not provide any way to actually use the second IP address.

The Microsoft DHCP Client

Microsoft has included a DHCP client with Windows since it released Windows for Workgroups. DHCP clients are included with Windows 98, Windows 2000, and Windows XP. All these DHCP clients are quite similar to each other. This section describes the Microsoft DHCP client and points out the differences among the various implementations. More information on the Microsoft DHCP client is available in Chapter 4, "Dynamic Host Configuration Protocol," of *The Microsoft Windows 2000 Server TCP/IP Core Networking Guide*, which can be found on the Microsoft Web site, `www.microsoft.com`.

Installing and Enabling the Microsoft DHCP Client

The DHCP client in Microsoft Windows is part of the IP protocol suite. Windows 98 and Windows 2000 may not automatically install the IP protocol suite. Depending on your system configuration, you might need to install it yourself. In some versions of Windows, you do not need to install the IP protocol suite; it is installed automatically.

To use the DHCP client in Windows 98, you must install IP networking. You can do this by right-clicking the Network Neighborhood icon on the Windows desktop and selecting Properties. Windows then displays the Network dialog box. If you select the Configuration tab, you see a window within the dialog box that lists a series of protocols and the network interfaces to which these protocols are bound. If the TCP/IP protocol is already bound to the interface you intend to use, you don't need to install it again. If it is not, click Add.

Windows 98 then displays the Select Network Component Type dialog box. Click the line that reads Protocol, and then click the Add button. This opens the Select Network Protocol dialog box, which contains two windows: The one on the left displays a list of manufacturers, and the one on the right displays a list of protocols. The manufacturer for the TCP/IP protocol is Microsoft; if you click Microsoft in the left window, you should see TCP/IP as one of the choices in the right window. You can click TCP/IP, and then click OK to install it. Windows might prompt you to insert a floppy disk or CD-ROM from which it can install the TCP/IP software.

Figure 21.1 shows the Network dialog box, the Select Network Component Type dialog box, and the Select Network Protocol dialog box, as they look if you follow the instructions in this section to the point just before you click OK.

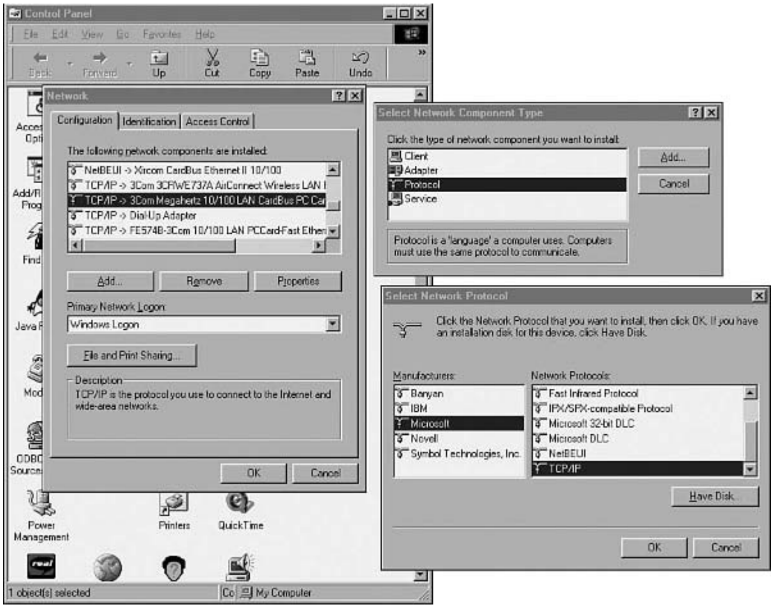

FIGURE 21.1 Dialog boxes used when installing TCP/IP on Windows 98.

After you install TCP/IP protocol support, you can configure it by selecting a TCP/IP binding from the Network dialog box; you should select the binding for the network card on which you want to enable DHCP. Then click Properties. Windows will then display the TCP/IP Properties dialog box. Select the IP Address tab, as shown in Figure 21.2, and click the radio button Obtain an IP Address Automatically to enable DHCP.

You can configure the Windows 98 DHCP client to obtain its IP address from DHCP but still use a statically configured DNS configuration. If you do not want it to do this, you should go back to the TCP/IP Properties dialog box and select the DNS Configuration tab. Then click the Disable DNS radio button. If DNS is enabled, even if you didn't provide configuration information, Windows 98 does not use the DNS configuration that the DHCP server provides.

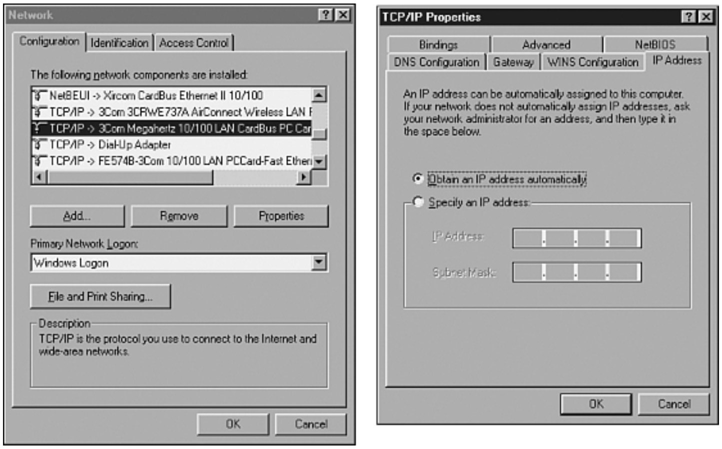

FIGURE 21.2 The TCP/IP Properties dialog box.

The Microsoft DHCP Client on Windows 2000

To install TCP/IP in Windows 2000, first open the connection configuration dialog box. Select Start and then click Settings. Then select Network and Dial-up Connections. Select the connection (usually Local Area Network) from the Network and Dial-up Connections entry. Click the Properties button to display the current protocols installed on the interface, and then click Install to install TCP/IP (see Figure 21.3).

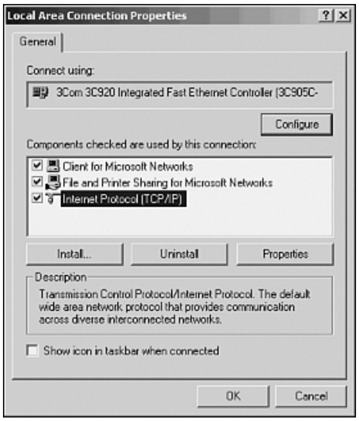

FIGURE 21.3 The Windows 2000 TCP/IP Installation dialog box.

After you install the TCP/IP software, you can select the Internet Protocol (TCP/IP) entry from the Properties dialog box and click Properties. To enable DHCP, click Obtain an IP Address Automatically, as shown in Figure 21.4.

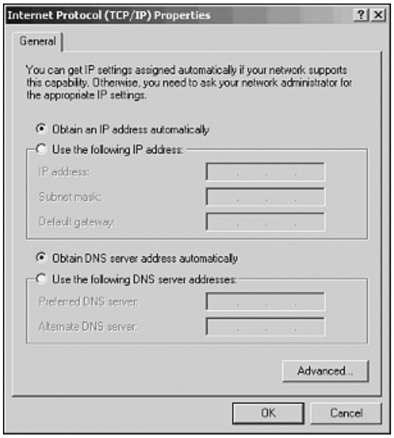

FIGURE 21.4 The Windows 2000 TCP/IP Properties dialog box.

The Windows DHCP Client User Interface

The Microsoft Windows DHCP client for Windows 95, Windows 98, and Windows Me has a user interface program called `winipcfg`. To use `winipcfg`, choose Start, click Run, and then type `winipcfg` in the Open dialog box and click `OK`.

`winipcfg` then pops up its own dialog box, displaying the status of the DHCP client. To see the complete status, click the More Info button. Figure 21.4 shows `winipcfg` after More Info has been selected.

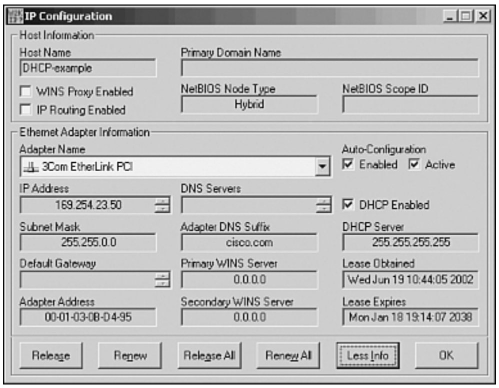

FIGURE 21.5 The `winipcfg` dialog box.

You can use the winipcfg dialog box to see what options the DHCP server sends, to see what IP address the server provides, and to display various other details of the DHCP client's state. If the client is configuring more than one network adapter, you can change the adapter whose state is being displayed or affected by selecting the network adapter from the drop-down list box below the Ethernet Adapter Information label.

You can also use winipcfg to release and renew the DHCP client's lease; to release the lease, click the Release button or the Release All button. To renew the lease, click the Renew button or the Renew All button. Release and Renew affect only the configuration for the network interface whose state is currently displayed, whereas Release All and Renew All affect the state of all network adapters.

Behavior Specific to the Microsoft DHCP Client

This section describes some of the unique behavior of the Microsoft DHCP client. Although this behavior is, by and large, unique to the Microsoft DHCP client, these differences are due to interpretation of the protocol specification. Nothing that the Microsoft DHCP client does that is documented here is specifically forbidden by the protocol specification.

Use of the `host-name` Option

The DHCP protocol standard, and indeed the documentation for many DHCP servers, leads you to believe that if you define a host-name option for a particular DHCP client, that client will use the defined host-name option. Unfortunately, the Microsoft DHCP client does not use the host-name option in this way. Indeed, it ignores the host-name option in any DHCP packets it receives.

Instead, it uses the host-name option to tell the DHCP server what it thinks its name is. It does this by sending its NBNS name in the host-name option whenever it sends a DHCPDISCOVER or DHCPREQUEST message. Some DHCP servers take advantage of this by using the NBNS name to do a DNS update. Also, as mentioned in Chapter 16, "Client Identification and Fixed-Address Allocation," you can configure some DHCP servers to use the NBNS_name as a client identifier. Some problems occur with this, but to understand them, you must know a bit about NBNS.

Microsoft historically has used NBNS, also known as WINS, to name Windows network clients. NBNS is a distributed naming protocol that can function over IPX or IP and can operate whether or not a centralized NBNS server is present. If no centralized NBNS server is present, the protocol operates by broadcasting NBNS queries to the network and waiting for responses. NBNS is in the same class of protocol as Sun's Network Information Services protocol; it is easy to manage, and therefore advantageous, on a small network, but it does not scale for use throughout the Internet.

When Microsoft Windows is first installed, one of the questions it asks the person installing it is the name of the system. This question is mandatory; you can't finish the installation without answering it. People tend to answer the question in a variety of ways; if an individual is installing a single Windows workstation, he or she might give it a name such as TEDSPC or SNEECH. A systems administrator who is installing large numbers of Windows machines might have a standard way of choosing the name of the PC—either the name of the person to whom it is delivered, or by using some counting scheme, such as PC001, PC002, and so on. You can determine the NBNS name of a Windows machine by right-clicking the Network Neighborhood icon, selecting Properties from the menu, and then selecting the Identification tab, as shown in Figure 21.5.

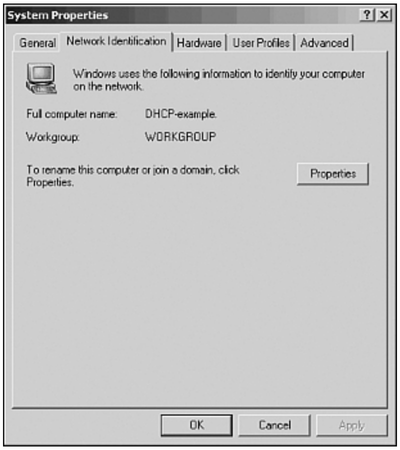

FIGURE 21.6 Finding the NBNS name of a Windows computer.

When the Windows machine first becomes active on the network, it uses the NBNS protocol to ensure that its name is unique within its workgroup. If an NBNS server is serving the administrative domain in which the workstation is installed, this process succeeds; if the user chooses a name that is in use by some other user in the same workgroup, Windows chooses a different name. If no NBNS server exists, you cannot ensure that the name is unique. Note also that NBNS only ensures that a name is unique within a workgroup, so if you have more than one active workgroup on your network, the NBNS name alone is not guaranteed to be unique.

NOTE

You can control whether a client broadcasts WINS requests by configuring the DHCP server to send a `netbios-node-type` option to indicate whether Windows should broadcast or contact a WINS server. You can also configure the DHCP server to send the `netbios-name-servers` option, which indicates which NBNS servers the DHCP client machine should use. Because a response from the DHCP server can affect the conflict resolution process, the DHCP server cannot assume that NBNS has checked the hostname for uniqueness before the DHCP client starts.

As a consequence, although it would be handy to be able to count on the uniqueness of the name that the Microsoft DHCP client sends in the `host-name` option, it is not possible to do so. If a DHCP server is going to use this name as a unique identifier, this may result in an identifier conflict, as described in Chapter 14. If a DHCP server wants to update the DNS server by using the `host-name` option sent by the Microsoft client, it must perform its own conflict resolution process prior to doing so, and it must ensure that the name is not already claimed by some other client.

The Client Identifier Option

The Microsoft DHCP client does not provide a user interface with which the user can choose the client identifier that it sends. Instead, it uses the format suggested in RFC 2132: The client identifier is a series of bytes beginning with a single byte containing the network hardware type, followed by the link-layer address of the network interface that the DHCP client is configuring.

The `DHCPINFORM` Message

Some Windows applications may need information from the DHCP server that is not requested or sent when the DHCP client acquires its lease. In that case, the application can send a `DHCPINFORM` message, requesting further information. For this reason, recent versions of Windows frequently generate `DHCPINFORM` requests, and it's helpful to have a DHCP server that supports `DHCPINFORM`.

The dhcpcd DHCP Client

Yoichi Hariguchi of Fore Systems and Sergei Viznyuk of PhysTech have developed the dhcpcd DHCP client for Unix and Unix-like operating systems. dhcpcd is a general-purpose DHCP client that implements all the messages in RFC 2131 and many of the options in RFC 2131.

dhcpcd passes configuration information through a file in the dhcpcd directory that can be processed by a shell interpreter to set environment variables. System configuration programs and other applications can look at the values of the environment variables to get the configuration information obtained by dhcpcd.

The dhcpcd client discovers and configures a device's interfaces. The client's behavior can be controlled by command-line options. Here is a partial list of the functions that can be specified to dhcpcd:

- Ignore the DNS servers supplied by the server.
- Set the domain name and the hostname of the device as supplied by the server; the default behavior is to ignore that information from the server.
- Specify the client identifier to be sent to the server from this client.
- Specify the vendor class to be sent to the server.
- Send a `DHCPINFORM` message to the server.

In normal operation, dhcpcd spawns a child process that runs in the background as a daemon. When the `dhcpcd` daemon is running, the `dhcpcd` application can be used to send a `SIGHUP` signal to the daemon to cause it to send a `DHCPRELEASE` message to the server, or it can be used to send a `SIGALRM` signal to cause the daemon to send a `DHCPRENEW` message.

The pump DHCP Client

Red Hat has developed the pump DHCP client as part of the Red Hat Linux distribution. It is the default DHCP client installed with recent versions of Red Hat Linux. pump is a general-purpose client that can act as either a DHCP client or a BOOTP client. It implements all the DHCP messages in RFC 2131 and a subset of the options defined in RFC 2132 and later DHCP RFCs.

The pump client can configure the IP address and subnet mask for an interface, the default gateway for the host, a list of DNS servers, and the domain search list in `/etc/resolve.conf`. It can send a client identifier similar to that sent by Windows clients, in which the client's link-layer address is sent as the client identifier.

pump can be controlled either by command-line options or through a configuration file. The command-line options can control which interface to configure, what lease-length and hostname to request, whether to act like a Windows client, and whether to configure parameters such as DNS servers and default gateway. After pump has completed the configuration process, it writes a configuration script file, by default `/etc/sysconfig/network-scripts/ifcfg-eth0`, which is subsequently executed as part of the normal interface initialization. Configuration parameters obtained from the DHCP server are used to set shell variables in this script. The shell variables, in turn, control the interface initialization.

In normal operation, pump runs in the background by spawning a child process that runs as a daemon. When the pump daemon is running, it can accept commands from the pump application to perform actions including immediate renewal of an address, release of an address, or printing of the current status information about the interface managed by the pump daemon.

The ISC DHCP Client

The ISC DHCP client is a general-purpose client intended for use on computers running Unix and Unix-like operating systems. The DHCP client protocol engine is implemented by using the same underlying network code that's used to implement the ISC DHCP server. On top of that is a DHCP client protocol engine. When an address is acquired, the DHCP client invokes a shell script that issues system-specific commands to configure the network interfaces and routing table, and to install the DNS configuration.

ISC DHCP Client Installation

The ISC DHCP client is included in the ISC DHCP distribution. The instructions for installing the DHCP server, included in Chapter 14, "The ISC DHCP Server," and in Appendix F, "DHCP Server and Operating System Versions," also apply to the client. The client program is called `dhclient`, and it is installed automatically when you type `make install` after building the distribution, as shown in Chapter 14.

Because the underlying network interface code is the same for the ISC DHCP client and server, the system-specific caveats mentioned in Appendix F all apply to the client as well as the server. Like the server, the client should be installed in the system startup script so that it automatically configures network interfaces on system startup.

ISC DHCP Client Operation

As mentioned in the section "The Theory of DHCP Client Operation," earlier in this chapter, the ISC DHCP client normally starts up and runs in the foreground until it acquires an IP address. After it acquires an IP address, the foreground process exits, and the client continues running in the background as a daemon.

You can take advantage of this behavior by starting the DHCP client in the system startup script before any daemons that depend on the network being configured are started. Because the client does not exit until the network is configured (or until it can no longer successfully be configured), daemons that are started after the DHCP client's foreground process exits can reasonably depend on having network connectivity when they are started.

If, for some reason, it is not considered desirable to have the DHCP client go into the background on startup, you can specify the `-d` switch. You can specify the `-q` switch to make the client startup process completely silent. You can obtain a complete listing of DHCP client switches in the `dhclient` manual page by typing `man dhclient` at the shell prompt.

You can control which broadcast interfaces the DHCP client should configure by providing a list of those interfaces on the command line. The names of the interfaces should be the same names shown when you type `netstat -i`.

ISC DHCP Client Configuration

The ISC DHCP client does not require any special configuration. However, it is possible to provide a configuration to change the default behavior of the DHCP client. You can widely adjust the DHCP client's protocol timing through the configuration file. You can also configure the client to send different values to the DHCP server.

Requesting Additional Options

The DHCP client can be configured to request additional options that can be used by a custom DHCP client shell script. You can do this by using the `request` statement, as shown in Example 21.1 (note that if you set up a parameter request list, you must explicitly request the `subnet mask`, `routers`, `domain-name`, and `domain-name-servers` options, or the client does not receive those options).

Example 21.1

```
request subnet-mask, routers, domain-name, domain-name-servers,
        ntp-servers;
```

Rejecting Unacceptable Offers

You can instruct the DHCP client to reject DHCP offers that do not provide options that the client needs. For example, you might say that at a minimum, the client needs `subnet-mask` and `routers` options, as shown in Example 21.2.

Example 21.2

```
require subnet-mask, routers;
```

Overriding Options Sent by the DHCP Server

You can configure the DHCP client to override or augment some values that the DHCP server returns to it. For instance, the DHCP server might return a domain name of `fugue.com` when you want to have `/etc/resolv.conf` specify a domain search path of `manhattan.fugue.com fugue.com`. You can do this by using the `prepend` statement. If the computer on which you are running the DHCP server has a local name server, you might want to use only your own name server. You can accomplish this by using the `supersede` statement.

Example 21.3 shows a sample configuration that performs these two customizations.

Example 21.3

```
prepend domain-name "manhattan.fugue.com";
supersede domain-name-servers 127.0.0.1;
```

Doing Interface-Specific Customizations

You can specify different customizations for different network interfaces. One useful customization is simply to specify a list of network interfaces to configure in the configuration file, rather than having to specify them on the command line. Only the interfaces that are mentioned in the configuration file (or on the command line) are configured.

Example 21.4 shows an example of the `interface` statement being used to send different `client identifier` options for two different network interfaces. If the `send` statement is specified outside an `interface` declaration, it causes the same value to be sent for all interfaces.

Example 21.4

```
interface "ln0" {
  send dhcp-client-identifier 1:8:0:2b:4c:a9:ad;
}

interface "fpa0" {
  send dhcp-client-identifier "snorg-fddi";
}
```

Microsoft DHCP Client Emulation

To emulate the behavior of a Microsoft Windows DHCP client, it is sometimes useful to send a `host-name` option and a standard client identifier. Example 21.5 shows how to do this.

Example 21.5

```
send dhcp-client-identifier 1:0:0:ad:a9:22:10;
send host-name "ABACUS\0";
```

Microsoft Windows DHCP clients send a zero byte at the end of the `host-name` option. Microsoft clients also send a client identifier that is based on the network interface card's link-layer address. By sending the same sort of options that the Microsoft server does, the ISC server can be made to convince a DHCP server that only works with Microsoft DHCP clients to work with the ISC DHCP client as well.

The example assumes that the client's network interface is an Ethernet card with a link-layer address of 0:0:ad:a9:22:10.

ISC DHCP Client Network Setup Script Customization

Because the ISC DHCP client does its network setup through a shell script, it's possible to customize the network setup process quite extensively. You are not supposed to modify the shell script directly; instead, the standard client script invokes two scripts: one before it takes any action and one when it exits. The first script is called `/etc/dhclient-enter-hooks`, and the second is called `/etc/dhclient-exit-hooks`. The manual page for the DHCP client script `dhclient-script` describes how you can program these customization scripts.

ISC DHCP Client Debugging

If the DHCP client is having trouble getting an IP address, it can be useful to observe its startup messages to see what is happening. The progress of the client through the phases described in the section "The Theory of DHCP Client Operation," earlier in this chapter, displays up to the time at which the client gets an IP address. After the client has an IP address, you can track its progress by looking in the system log; like the ISC DHCP server, the ISC DHCP client logs all its activity by using the `LOG_DAEMON` facility.

A normal client startup log should look something like the one in Example 21.6.

Example 21.6

```
grosse# dhclient
Internet Software Consortium DHCP Client V3.0.1
Copyright 1995-2002
The Internet Software Consortium.
All rights reserved.
Please contribute if you find this software useful.
For info, please visit http://www.isc.org/dhcp-contrib.html
Listening on BPF/ep0/00:10:4b:ec:93:61
Sending on BPF/ep0/00:10:4b:ec:93:61
Sending on Socket/fallback/fallback-net
DHCPDISCOVER on ep0 to 255.255.255.255 port 67 interval 1
DHCPOFFER from 10.0.0.3
DHCPREQUEST on ep0 to 255.255.255.255 port 67
DHCPACK from 10.0.0.3
New Network Number: 10.0.0.0
New Broadcast Address: 10.0.0.255
bound to 10.0.0.2 -- renewal in 1800 seconds.
grosse#
```

If no DHCP server is on the network segment to which the client is connected, you simply see a series of `DHCPDISCOVER` messages, and finally, after about one minute, the client announces that it's giving up and goes into the background. This can also happen if the network is very busy and the DHCP server or the network is dropping packets, or if the DHCP server has no IP addresses available to offer to the client.

If two DHCP servers are competing, and one is incorrectly configured, you see a slightly different sequence, as shown in Example 21.7.

Example 21.7

```
 [...]
DHCPDISCOVER on ep0 to 255.255.255.255 port 67 interval 1
DHCPOFFER from 10.0.0.3
DHCPOFFER from 10.0.0.7
DHCPREQUEST on ep0 to 255.255.255.255 port 67
DHCPNAK from 10.0.0.7
DHCPDISCOVER on ep0 to 255.255.255.255 port 67 interval 2
[...]
```

In this case, the client fails to get an IP address because when it selects an address from the DHCP server at 10.0.0.3, the DHCP server at 10.0.0.7 sends a `DHCPNAK` message, indicating that the address is incorrect and causing the client to start over. If you are not the administrator for this site and you have no immediate way to fix the DHCP server at 10.0.0.7, you can instruct the DHCP client to ignore responses from that server, as shown in Example 21.8.

Example 21.8

```
reject 10.0.0.7;
```

If the DHCP client already has an IP address, it starts up in the `INIT-REBOOT` state and initially broadcasts a `DHCPREQUEST` message, rather than a `DHCPDISCOVER` message. When the network configuration it's requesting is correct and the DHCP server is available, this looks exactly like what's shown in Example 21.8, minus the `DHCPDIS-COVER` and `DHCPOFFER` messages. If the address is incorrect, the transaction (if it goes well) looks like the one shown in Example 21.9.

Example 21.9

```
grosse# dhclient
[...]
DHCPREQUEST on ep0 to 255.255.255.255 port 67
DHCPNAK from 10.0.0.3
DHCPDISCOVER on ep0 to 255.255.255.255 port 67 interval 1
```

Example 21.9 Continued

```
DHCPOFFER from 10.0.0.3
DHCPREQUEST on ep0 to 255.255.255.255 port 67
DHCPACK from 10.0.0.3
New Network Number: 10.0.0.0
New Broadcast Address: 10.0.0.255
bound to 10.0.0.2 - renewal in 1800 seconds.
grosse#
```

If the DHCP server can't provide an IP address to a DHCP client that starts in the `INIT-REBOOT` state, the log shows the `DHCPREQUEST/DHCPNAK` exchange, and then the client sends `DHCPDISCOVER` messages until it gives up because the server isn't responding.

Controlling the ISC DHCP Client

You can control the ISC DHCP client by using the OMAPI interface. Using OMAPI, you can tell the DHCP client to release its lease and exit, to stop renewing its lease, or to immediately renew its lease. This can be useful when you're shutting down the system, and also when the system is hibernating and waking up.

In order to control the DHCP client with OMAPI, you must configure the client to accept OMAPI connections. To do this, you add an `omapi port` statement, a `key` statement, and an `omapi key` statement to your `dhclient.conf` file, as shown in Example 21.10.

Example 21.10

```
key omkey {
  algorithm HMAC-MD5.SIG-ALG.REG.INT;
  secret MBZMjgiuubIC7IqAU0v7k8wtAJY2axufNyJVUYlyro0=;
}

omapi port 923;
omapi key omkey;
```

The secret key is required because the OMAPI protocol is a TCP/IP-based protocol, and you do not want anybody who can connect to the OMAPI port to be able to control your DHCP client. Do not use the key shown in the example; generate one by using the `dnssec-keygen` command, which comes with the ISC BIND version 9 distribution. Example 21.11 shows how to do this.

Example 21.11

```
tongpanyi% dnssec-keygen -a HMAC-MD5 -b 256 -n USER omkey
Komkey.+157+00560
tongpanyi% cat Komkey.+157+00560.private
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: MBZMjgiuubIC7IqAU0v7k8wtAJY2axufNyJVUYlyro0=
tongpanyi%
```

The `dnssec-keygen` command requires random data to generate a key, and it either asks you to type on the keyboard or use your operating system's built-in secure random number generator. If your system does not have a very good source of entropy, `dnssec-keygen` appears to hang. If this happens, you can interrupt it and then restart it with the `-r keyboard` flag, which tells it to just use your typing on the keyboard to generate randomness.

OMAPI is an object-oriented API for accessing and changing objects in remote servers. To control the client, you use the `omshell` command to change the value of the `state` attribute of the `control` object on the DHCP client. The three values that are useful are shown in Table 21.1.

TABLE 21.1 DHCP Client Control States for OMAPI

Value	Action	Description
2	Shutdown	The DHCP client deletes its DNS registration, sends a `DHCPRELEASE` message to the DHCP server, and exits.
3	Hibernate	The DHCP client stops sending packets on the network interfaces it controls. This should be done before putting a portable computer into any sort of hibernation.
4	Awaken	The DHCP client immediately resumes the DHCP protocol by going into the INIT-REBOOT state, which means it broadcasts a `DHCPREQUEST` message to confirm its IP address. If its lease expired while it was asleep, it goes into the `INIT` state and broadcasts a `DHCPDISCOVER` message. This should be done whenever a portable computer awakens from hibernation, and you might also need to do it if you unplug a computer from one network and plug it into another.

Example 21.12 shows the `omshell` command being used to set the client into the `shutdown` state.

Example 21.12

```
bdagmed# omshell
> server 127.0.0.1
> port 854
> key omkey MBZMjgiuubIC7IqAU0v7k8wtAJY2axufNyJVUYlyro0=
> connect
obj: <null>
> new control
obj: control
> open
obj: control
state = 00:00:00:00
> set state = 4
obj: control
state = 4
> update
obj: control
state = 4
> ^C
bdagmed#
```

The Apple MacOS X DHCP Client

Apple's MacOS X operating system comes with a DHCP client that combines many of the good features of the DHCP clients described previously. This client can be configured with a GUI, and it can be controlled at the command line as well. To access the DHCP GUI, click the Apple icon in the upper-left corner of the screen and select System Preferences, as shown in Figure 21.7.

When you select System Preferences, the `System Preferences` application opens, as shown in Figure 21.8.

To configure DHCP, click the icon labeled `Network`. The `Network` dialog box, shown in Figure 21.9, appears.

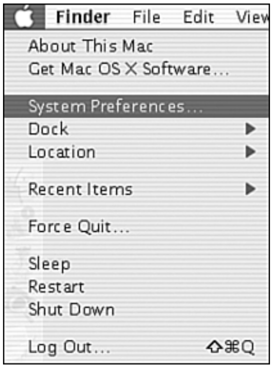

FIGURE 21.7 Getting to System Preferences.

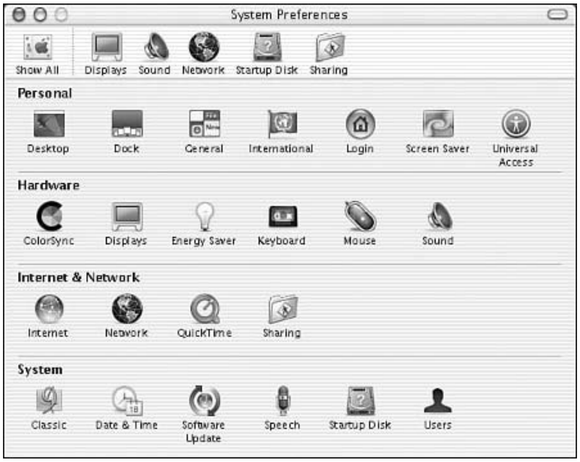

FIGURE 21.8 The System Preferences Main screen.

The Network dialog box allows you to select the network interface you are configuring. In Figure 21.9, we are configuring the Ethernet interface. You can select a different interface to configure by clicking on the selector menu labeled `Show: Ethernet Interface`. The Network dialog box allows you to select a variety of different ways of configuring the network interface. We have chosen DHCP, using the selector menu that in Figure 21.9 is labeled Configure: Using DHCP.

FIGURE 21.9 The Network dialog box.

The Network dialog box lets you set a `client identifier` to send to the DHCP server; just type a string of text into the text box labeled `Client Identifier (optional)`. This is equivalent to Example 21.4. You can also customize the list of domain name servers to use and enter a list of domain names to search, by using the `Domain Name Servers` and `Search Domains` text boxes. This is equivalent to Example 21.3.

Controlling the MacOS X DHCP Client from the Command Line

You can make temporary changes to the DHCP configuration on MacOS X and also get the results of the most recent DHCP exchange by using the `ipconfig` command. This can be very helpful because there is no way to use the GUI to release a lease or renew a lease. The `ipconfig` command operates on a single network interface, and you have to know the name of the network interface on which you want to operate. Ethernet and wireless Ethernet interfaces on MacOS X always have a name that starts with `en` and ends with a number. If you have only one Ethernet interface in your computer, it is `en0`. If you have one Ethernet interface and one wireless Ethernet interface, one is `en0` and the other is `en1`. You can tell which is which by getting a list of all the interfaces by using the `ifconfig` command, as shown in Example 21.13.

Example 21.13

```
dechen% ifconfig -a
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet 127.0.0.1 netmask 0xff000000
en0: flags=8863<UP,BROADCAST,b6,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 00:03:93:17:66:3c
        media: autoselect (none) status: inactive
```

Example 21.13 Continued

```
        supported media: none
                         autoselect
                         10baseT/UTP <half-duplex>
                         10baseT/UTP <half-duplex,hw-loopback>
                         10baseT/UTP <full-duplex>
                         10baseT/UTP <full-duplex,hw-loopback>
                         100baseTX <half-duplex>
                         100baseTX <half-duplex,hw-loopback>
                         100baseTX <full-duplex>
                         100baseTX <full-duplex,hw-loopback>
en1: flags=8863<UP,BROADCAST,b6,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 10.0.0.102 netmask 0xff000000 broadcast 10.255.255.255
        ether 00:30:65:01:e7:88
        media: autoselect status: active
        supported media: autoselect
dechen%
```

In Example 21.13 you can see three network interfaces: `lo0`, `en0`, and `en1`. The loopback interface, `lo0`, is never configured with DHCP. The other two can be configured with DHCP. You can tell that `en0` is the twisted-pair Ethernet interface and `en1` is the wireless interface because `en0` provides so many supported media options, and with wireless, there is no choice of media type.

In Example 21.13 `en1` has the IP address 10.0.102. To release the lease on this IP address, you use the `ipconfig` command as shown in Example 21.14.

Example 21.14

```
dechen% ipconfig set en0 NONE
```

After you have released the lease in this way, the DHCP client does not attempt to get another IP address until you reenable it. You can reenable it by using the `ipconfig` command as shown in Example 21.15. You can also trigger a lease renewal at any time by using the command shown in Example 21.15, even if the client already has a valid lease.

Example 21.15

```
dechen% ipconfig set en0 DHCP
```

You can also find out what options the DHCP server sent the DHCP client on a particular interface the last time the DHCP client communicated with the server, also by using the `ipconfig` command, as shown in Example 21.16.

Example 21.16

```
dechen% ipconfig getpacket en1
op = BOOTREPLY
htype = 1
dp_flags = 0
hlen = 6
hops = 0
xid = 856380767
secs = 0
ciaddr = 10.0.0.102
yiaddr = 10.0.0.102
siaddr = 0.0.0.0
giaddr = 0.0.0.0
chaddr = 0:30:65:1:e7:88
sname =
file =
options:
Options count is 11
server_identifier (ip): 10.0.0.1
dhcp_message_type (uint8): ACK 0x5
lease_time (uint32): 0xe10
renewal_t1_time_value (uint32): 0x708
rebinding_t2_time_value (uint32): 0xbb8
domain_name (string): fugue.com
router (ip_mult): {10.0.0.1}
domain_name_server (ip_mult): {10.0.0.1}
subnet_mask (ip): 255.0.0.0
broadcast_address (ip): 10.255.255.255
end (none):
dechen%
```

This shows everything in the packet—the options and the BOOTP packet header. A shell script can use this information to extract interesting options from the packet. Unfortunately, at the time of this writing there is no way to get the MacOS X DHCP client to request different options or to send different options.

Summary

This chapter describes the theory of operation of a DHCP client, and then describes several popular DHCP clients, how to install them, how to configure them, and some problems that you might see with the clients in the event of a server misconfiguration.

The DHCP client drives the DHCP protocol; it contains a state machine, handles message timeouts, configures the network interface according to the responses it gets from the DHCP server, and unconfigures the interface if it loses contact with the DHCP server for such a long time that its lease expires.

All DHCP clients follow pretty much the same process in acquiring IP addresses, but each DHCP client behaves slightly differently in some cases, depending on the design goals of the implementor.

22

Setting Up DHCP in a Small Office

IN THIS CHAPTER

- Small Office Network Architectures
- IP Address Translation
- Running a DHCP Server and Client on the Same Computer
- Running the DHCP Server on Your Firewall
- Problems with DSL Routers
- Configuring an Integrated Router/Server
- Configuring a WAN Port
- Configuring a LAN Port and Small Network Services

Most of this book is targeted at professional network administrators who are setting up DHCP service for large sites and who have a great deal of control over how their networks are configured. This chapter describes how to set up DHCP service for a smaller network, which may be connected to the Internet using an analog phone line, ISDN, DSL, or a cable modem. Seamless integration of DHCP service from an ISP and DHCP service on a local network can pose some problems. This chapter addresses these problems and includes some discussion about firewalls that might also be of interest to network administrators at large sites.

The networks discussed in this chapter are in a small office of some kind—perhaps an ophthalmologist's office, a home office, or a network that enables you to share an Internet connection transparently with your family. Setting up a DHCP server for such an application isn't much different from setting one up for a large site; just about everything you have learned in previous chapters applies equally well to a small office environment. However, some of the things you might do to work around the limitations of a small networking environment could interact badly with DHCP service. This chapter informs you of some of these problems and provides ways of working around them.

You can use the information in this chapter if any of the following is true:

- You are connected to an ISP through a bridge, not a router (for example, with DSL or cable modem service).

- You are running DHCP server on the same computer that you are using to do IP address translation between your office network and the Internet.
- You are running a DHCP server on the same computer that you are using as a router or firewall between your network and the Internet.
- You are using one of the integrated, or "appliance," Internet connection devices that act as routers and provide DHCP, DNS, and NAT services to the small office or home network.

Small Office Network Architectures

A very common configuration in small office environments is a computer running Linux or NetBSD that is connected to the Internet on one interface and to the local office LAN on a second interface; this computer acts as a router *and* as a DHCP server. Often, the Internet connection provides only one IP address, so the router must be configured to provide NAT (also known as IP masquerading). Even when the ISP provides more than one IP address, the router is often configured to act as a firewall. In some cases, the router might need to act as a DHCP client on the interface that is connected to the Internet but as a DHCP server on the other interface. In this situation, you might use one of the three topologies shown in Figure 22.1.

In Topology A, the router/server is connected directly to the ISP through some kind of point-to-point link: an analog or ISDN modem, a leased line, or something similar.

In Topology B, the ISP provides a device that is connected to the ISP's network on one side and to the local network on the other. This device is a bridge, forwarding packets from one network to the other. It probably also filters what it sends so that it doesn't forward packets on one side of the bridge that aren't intended for computers on the other side of the bridge.

Topology C also includes a device provided by the ISP, but instead of connecting your side of the device to the local network, you connect it to a separate network that only your router/server is connected to.

BRIDGING DEVICES

If your ISP provides you with a bridging device to connect your local office network to the ISP's network, you should use Topology C if possible. Topology B might inappropriately allow packets from your ISP's customers or your ISP onto your network, and it might inappropriately allow your packets onto your ISP's customers' networks. Because your ISP is providing the interface, you must trust your ISP to configure the bridge so as not to forward packets containing private information from your network to the ISP's network. Even if the ISP doesn't intend to do this, you might not want to assume that the ISP knows how to configure the bridge so that this won't happen. In addition to the obvious potential security problems, your DHCP service might interfere with the ISP's DHCP service, and vice versa.

Ralph Droms tried to use Topology B to set up a home network when he could not locate a router/server computer to use that had two network interfaces. He found that his DSL link failed whenever he tried to connect a second computer to the hub in his home. After a few calls to the ISP, Ralph learned that the DSL link functioned with only one device; as soon as the DSL equipment in the ISP's office detected a second computer on the hub, it shut down. Ralph tracked down a second network interface for his router/server and successfully connected his home network by using Topology C.

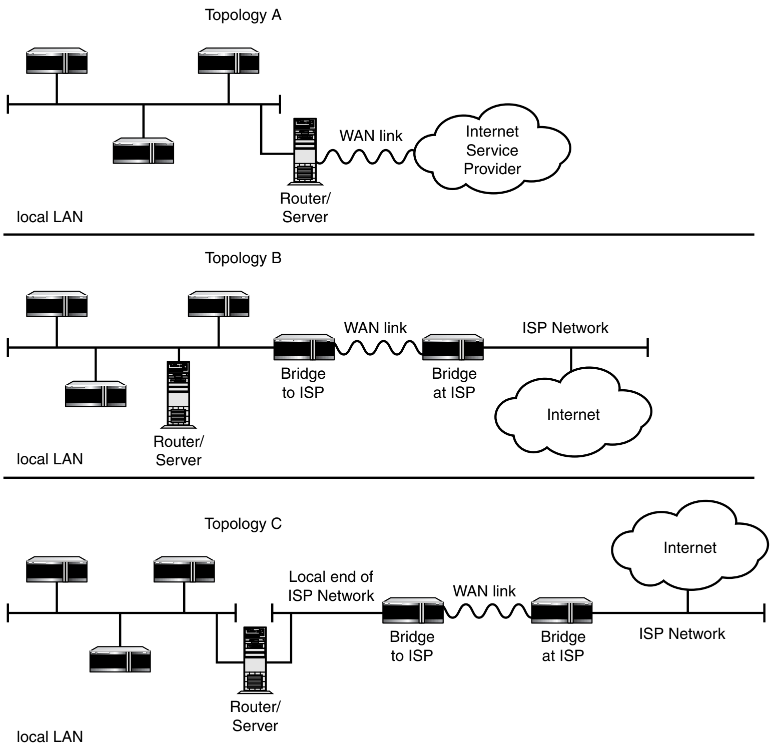

FIGURE 22.1 Three topologies for a small office network.

IP Address Translation

If you are connecting through an Internet connection that provides you with only one IP address but you have more than one computer in your office that needs to use the Internet, you need IP address translation. An IP address translation router operates by rewriting packet headers so that packets sent from your office network to the Internet appear to be coming from the router (which has a valid IP address on the Internet). Packets sent from the Internet to your router are rewritten and forwarded to the correct machine on your local network.

If you are using private IP addresses and NAT on a small office network, you can use DHCP to manage the private IP addresses. Configuring the ISC DHCP server for a small network, using private IP addresses, is discussed in more detail later in this chapter.

You can configure an IP address translation router by using any of the topologies shown in Figure 22.1. If the router also acts as a firewall, keep in mind that Topology B provides less security than Topology A or C because packets from outside the firewall actually run on the same network segment as packets running inside the firewall.

Running a DHCP Server and Client on the Same Computer

If you are connecting to the Internet by using DSL or a cable modem, your ISP most likely expects you to use DHCP to get an IP address to communicate on the Internet. In that case, you must set up both a DHCP client and a DHCP server on your router/server machine.

Running the DHCP Server and Client on Different Interfaces

If you set up a network by using Topology B, you must make sure that the DHCP server running on your server/router doesn't try to provide the DHCP client running on your server/router with an IP address. You can easily exclude the DHCP client from receiving DHCP service by including a host declaration in the server's configuration file (see Example 22.1).

Example 22.1

```
host router-server {
  hardware ethernet 08:00:2b:5a:19:22;
  deny booting;
}
```

To run the client and server on different interfaces, you *must* use an operating system on which the DHCP distribution supports the use of multiple network interfaces. You can find a list of these operating systems in Appendix H, "DHCP Server and Operating System Versions."

You must also make sure that your DHCP client and server use the correct interfaces; that is, the DHCP client needs to configure the interface that is connected to your ISP, and the DHCP server needs to listen for DHCP messages on the interface that is connected to your small office network. For both the client and server, you can specifically identify the interfaces on the command line for each, as shown in Example 22.2.

Example 22.2

```
dhclient if0
dhcpd if1
```

Configuring the Client

The DHCP client normally configures all broadcast interfaces that it finds. To prevent this, you can either specify on the command line the name of the interface that it should configure when you invoke it or you can write an interface declaration for that interface in the `/etc/dhclient.conf` file. The declaration can be as simple as the one shown in Example 22.3 or as complex as you need to make it.

Example 22.3

```
interface "ep0" {
}
```

Configuring the Server

You can either invoke the DHCP server with the name of the interface connected to your local office network or write a shared-network declaration for the network connected to the subnet on which your Internet connection is terminated. Example 22.4 shows how you can do this.

Example 22.4

```
not authoritative;


shared-network ISP {
  subnet 192.168.0.0 netmask 255.255.255.0 {
  }
}


subnet 10.0.0.0 netmask 255.255.255.0 {
  authoritative;
}
```

In this example, the configuration file includes the `not authoritative` statement, which sets the default to be that the server does not send `DHCPNAK messages`. The ISP subnet declaration identifies the ISP subnet to ensure that the server does not send `DHCPNAK messages` to DHCP messages received from the ISP's subnet. The subnet declaration for the local office network then states that the server *is* authoritative for the local office network, meaning that it sends `DHCPNAK messages` in response to DHCP messages received from the local office subnet. In addition, the subnet declaration for the local office network also needs some configuration parameters and some IP addresses to assign.

Running the DHCP Client and Server on the Same Interface

If you are using Topology B, it's rather difficult to guarantee that the DHCP server and client don't interfere with each other because they are both using the same interface. The DHCP client normally initializes the interface on startup into a state where it has no IP addresses, which means it discards any addresses from the small office network that were previously configured for that interface. If the DHCP client on the operating system you are using uses the BSD socket API, you cannot prevent this from happening; the BSD socket API doesn't provide a mechanism for working around it.

You can tell which API the client is using by starting it up with `dhclient -d`.

At the end of the client's output are two lines that start with `Listening on` and `Sending on`, similar to the output shown in Example 22.5. After that is the name of the API being used; in this case it is `Socket`, meaning the BSD socket API.

Example 22.5

```
Listening on Socket/ep0
Sending on Socket/ep0
```

Avoiding the `PREINIT` Problem

The `PREINIT` phase of the client script normally starts the network interface. If the DHCP client is not using the BSD socket API, you can prevent it from initializing the network interface by installing the `/etc/dhclient-enter-hooks` script shown in Example 22.6.

Example 22.6

```
if [ x$reason = xMEDIUM ] || [ x$reason = xPREINIT ]; then
  exit 0
fi
```

The script in the example disables the PREINIT phase. This means the network must already be started before the client runs. To complete this workaround, you must arrange to configure the interface with the IP address on your office network *before* you start the DHCP client.

Configuring an Alias in the DHCP Client

When the DHCP client acquires a new IP address for a given interface, it normally gets rid of the old network configuration for that interface and installs the new one. To have the DHCP client reinstall the IP alias as it configures the network interface, you must write an alias declaration in the /etc/dhclient.conf file. The alias declaration tells the DHCP client that a second IP address needs to be configured on the interface, in addition to the one it got from the DHCP server. Example 22.7 shows an alias declaration for the DHCP client whose startup output is shown in Example 22.6.

Example 22.7

```
alias {
  interface "ep0";
  fixed-address 10.0.0.1;
  option subnet-mask 255.255.255.0;
}
```

This example assumes that the local network has a network number of 10.0.0.0 and a subnet mask of 255.255.255.0 and that the NAT router's IP address on that subnet is 10.0.0.1. You need the interface declaration within the alias declaration to tell the client that the IP alias should be configured on interface ep0, along with whatever address is received from the ISP's DHCP server.

Limiting the DHCP Server's Authority

As mentioned previously, if you're using Topology B, your local DHCP service and other DHCP services can be connected to the ISP to interfere with each other. In particular, your DHCP server might receive DHCP messages from clients on other networks, and it might interpret those messages as coming from a client that has an invalid IP address. Therefore, it is very important that you configure your DHCP server so that it does not send DHCPNAK messages in response to packets it receives from other networks. If you skip this step, you risk causing your ISP a great deal of trouble.

Fortunately, it is easy to configure the ISC server so that it does not send DHCPNAK messages. You simply include `not authoritative;` at the top of the /etc/dhcpd.conf file.

Running the DHCP Server on Your Firewall

A *firewall* is a computer that sits between one network and another, forwarding some packets while not forwarding others. You generally place a firewall between your "trusted" internal network—in this case, your office network—and the Internet, which you probably don't trust. The firewall is configured to prevent packets from being sent from the Internet that might compromise the security of machines on your internal network. This book does not describe the detailed workings of firewalls. If you don't know how they work and are curious, you might want to read *Firewalls & Internet Security: Repelling the Wily Hacker* by Steven M. Bellovin and William R. Cheswick.

Ideally, no network services should run on your firewall. Your firewall should just be filtering packets between your network and the outside world and preventing bad packets from getting in. If you install a DHCP server on a firewall, you risk the possibility that a bug in the DHCP server might enable an attacker outside your firewall to subvert the firewall and penetrate your network.

However, people who are setting up small office environments with only a few machines generally can't afford a separate machine to use as a firewall. They also tend not to be as concerned about active attacks from the Internet as they might be if they were involved in E-commerce or had a serious Web presence. If you are involved in either of these applications, you're probably already paying more every month for your Internet connectivity than a dedicated firewall machine costs.

You are likely to run into two problems when running the DHCP server on your firewall: filtering rules that prevent DHCP from working and the use of a server identifier that all clients might not be able to reach.

Filtering Rules

DHCP requires that clients be able to send packets from a source address of 0.0.0.0 to a destination address of 255.255.255.255.

A firewall is normally configured to allow only packets from or to your local subnet, and it might reject packets from 0.0.0.0 or to 255.255.255.255. Some firewall implementations examine packets when they receive them, rather than when the routing decision is made. If you have such a firewall, you must configure it to accept packets from 0.0.0.0 or to 255.255.255.255. It must also be able to send replies from its own IP address to 255.255.255.255. If you omit the first rule, the DHCP server never sees any DHCP packets. If you omit the second rule, it sees them and tries to respond, but the client never sees its response.

Server Identifiers

The DHCP protocol requires clients that are in the `RENEWING` state to send unicast messages of their renewal requests (`DHCPREQUEST` packets) to the IP address that the DHCP server provides in the `dhcp-server-identifier` option. The ISC DHCP server normally sends the primary IP address of the interface on which it receives the client's request in the `dhcp-server-identifier` option.

Firewall rules might prevent clients on the local network from sending packets to that address. This is most likely to occur with Topology B, in which the firewall has one interface configured with two IP addresses: one on your office subnet and one on the ISP's subnet. In that case, the DHCP server might choose to use the IP address on the ISP's subnet for the server identifier. If your router is not configured to pass DHCP packets, your DHCP clients might not be able to send unicast renewals to that IP address. To solve this problem, you simply write a `server-identifier` statement that uses the server's IP address on the office subnet.

Problems with DSL Routers

Some DSL routers are known to have problems. The routers in question are marketed as complete solutions; they have built-in firewall support, built-in IP address translation support, and built-in DHCP service. Although you should use the firewall support and IP address translation support in these routers, the DHCP support can be a problem if you aren't aware of its existence.

For instance, consider this scenario. A DSL router comes preconfigured with DHCP service enabled, but the DHCP server setup is completely incorrect. The router is connected to the same subnet on which a legitimate DHCP server is operating. Whenever a DHCP client tries to get an IP address, it sends a `DHCPDISCOVER message`, gets a `DHCPOFFER message`, sends a `DHCPREQUEST message`, and gets a `DHCPACK message` from the legitimate server. Unfortunately, before the `DHCPACK message` from the legitimate server arrives, a `DHCPNAK message` from the DSL router arrives, causing the client to report that it is not permitted to obtain an IP address.

The solution to this problem is to reconfigure the router to disable DHCP service. After this is done, the DHCP clients on the network can be configured with no trouble at all.

Configuring an Integrated Router/Server

One popular way to connect a small network to an ISP is with a dedicated device that acts as a router, performs NAT, and provides DHCP service. Such a device would take the place of the router/server in Topology C of Figure 22.1. Typically, the device would have two Ethernet ports, one of which connects to the DSL or cable modem and one that is used for the small network. Some of these devices also include built-in hubs and can provide four or eight ports for the local network.

The dedicated router/server requires much the same configuration as the Linux- or NetBSD-based system described earlier in this chapter. Because such devices are designed to provide router, NAT, and DHCP service, they usually come with Web-based user interfaces that are tailored to the specific services provided by the device. These devices come with initial configurations that are appropriate for many small networks, so that you may be able to simply plug the device into your network and use it immediately.

Configuring a WAN Port

Integrated router/server devices usually have separate WAN ports that must be configured to connect with the ISP service through a device such as a DSL modem or a cable modem. You need to configure this port according to the requirements of your ISP. If your ISP provides you a static IP address, subnet mask, default router, and DNS server, you should configure the router/server with those values. If your ISP uses DHCP, you need to configure the router/server to obtain its configuration from the DHCP service provided by your ISP.

The examples in this section are based on the Etherfast Cable/DSL Router manufactured by Linksys. This device includes a WAN port and either one or four LAN ports, depending on the model. The device is configured through a Web-based user interface, which is shown in the figures.

Figure 22.2 shows the main configuration page for the Etherfast Cable/DSL Router. This page is used to set up the WAN port and some other functions. In this example, the selection Obtain an IP Address Automatically configures the device to use DHCP from the ISP. If Specify an IP Address is selected, you enter the IP address provided by the ISP in the field next to the selection, and you enter the remainder of the information provided by the ISP in the fields labeled Subnet Mask, Default Gateway Address, and DNS (Required).

This example allows the possibility of using Point-to-Point Protocol over Ethernet (PPPoE). Some ISPs, especially those providing DSL service, use PPPoE to connect the customer’s modem device to the ISP network. If your ISP gives you configuration information for PPPoE, you should enter it in the appropriate fields.

Configuring a LAN Port and Small Network Services

There are a couple choices to make when you’re configuring a LAN port and the services delivered on the local network. The first of these is whether the local network can use globally routable IP addresses or private addresses. If you have arranged for a globally routable network address, you need to assign one of those addresses to the LAN interface of your router/server. Otherwise, you use a network address from the nonroutable or private address space and assign one of the addresses to the LAN interface. The Etherfast Cable/DSL Router uses the private network address 192.168.1.0 by default and assigns the IP address 192.168.1.1 to the LAN interface, as shown in Figure 22.2.

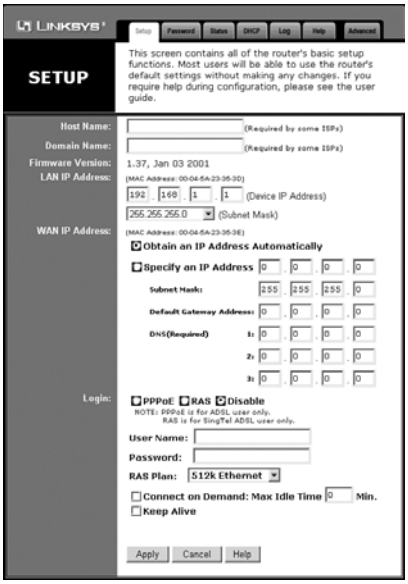

FIGURE 22.2 The setup page for the Linksys cable/DSL router.

Next, you choose whether to provide DHCP service on your local network with the router/server. The Linksys device comes configured to provide DHCP service; you should disable the service if you have another DHCP server on your local network. You also need to select the range of IP addresses to be assigned by the router/server. Be sure to choose addresses that are appropriate for the network address assigned to your local network, and don't include the address assigned to the router/server LAN interface as one of the available addresses. The Linksys device assigns addresses in the range 192.168.1.100 to 192.168.1.149 by default, as shown in Figure 22.3.

Finally, you configure the DHCP server with other parameters for the DHCP clients on your local network. Typically, the DHCP server provides a subnet mask, a default route, and a list of DNS servers. Often, the DHCP server automatically determines the subnet mask from the subnet mask of the local network interface, and it uses the router/server's IP address as the default route. If the router/server obtained its WAN interface IP address through DHCP from your ISP, the router/server DHCP server might automatically forward the list of DNS servers from the ISP to your local DHCP clients. Otherwise, you have to manually configure the DHCP server with a list of DNS servers. The Etherfast Cable/DSL Router automatically provides the subnet mask, default route, and DNS servers, so those parameters need not be manually configured for the DHCP server.

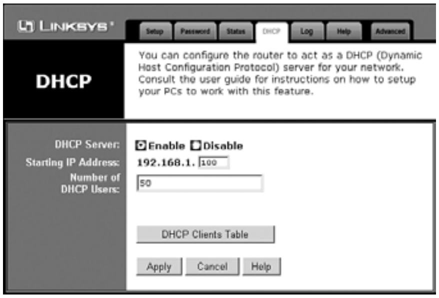

FIGURE 22.3 The DHCP service configuration page for the Etherfast Cable/DSL Router.

Summary

DHCP service can be useful in a small office environment. Although for the most part setting up DHCP service in such an environment is the same as setting it up in any other environment, you might encounter some special difficulties. For instance, you should be aware of the possible interference that a firewall can cause to a DHCP server running on it. You should also know about the problems that DHCP clients and servers can encounter when running on the same computer and the problems that a poorly configured DHCP server can cause when connected to a network segment that is bridged to an ISP's network. You can solve these problems in a number of ways, as described in this chapter.

Dedicated devices or appliances can provide DHCP service, NAT to allow multiple computers to share a single global IP address, and act as a firewall. These devices are simple to install and configure. They come with default configurations that are appropriate for many local network configurations and offer plug-and-play connection of a local network with multiple computers through a analog, modem, or DSL modem to an ISP.

23

Updating DNS with DHCP

IN THIS CHAPTER

- Overview of Updating DNS with DHCP
- The Motivation for Doing DNS Updates from DHCP
- The Domain Name Update Policy
- Name Clashes
- DNS Update Security
- Configuring the Servers
- DNS Record Removal
- Debugging Problems with DNS Updates

This chapter explains how to operate a DHCP service that performs DNS updates. It discusses both client- and server-originated updates, site update policies, security policies, and the mechanics of configuring DHCP clients and servers to update DNS.

You can make use of most of the information in this chapter without reading and understanding Chapter 11, "DHCP–DNS Interaction," but reading Chapter 11 will help you understand the subtleties, and more importantly, to be able to understand what's going on when the protocol doesn't work.

In order to illustrate the process of configuring the DHCP and DNS servers as described in this chapter, we use the ISC DHCP server and ISC BIND version 9 in our examples. These examples should carry over to other DHCP and DNS servers as well.

Overview of Updating DNS with DHCP

In order for a host to be properly identified in DNS, it must have an `A` record, or a forward mapping. This is a mapping from a fully qualified domain name (FQDN) to an IP address. For example, if a host is called `dechen` and is in the domain `example.com` and has an IP address of 10.0.0.1, it needs an `A` record that maps the name `dechen.example.com` to the IP address 10.0.0.1.

It is also useful to be able to determine a host's FQDN from its IP address. This is called a *reverse mapping*, and it is accomplished with a `PTR` record. The `PTR` record has a domain name that is constructed in a special way based on the IP address, and its value is the host's domain name.

In the example in the preceding paragraph, the `PTR` record's name would be `1.0.0.10.in-addr.arpa`, and its value would be `dechen.example.com`.

NOTE

We say "updating DNS" instead of "updating a DNS server" because DNS as a whole is a database, and DNS servers serve portions of that database. The `PTR` record and the `A` record for a client are in completely different parts of the database, and although it is likely in some cases that the same DNS server will serve both of these parts, this is not a requirement. In some of the configurations that this chapter describes, it is very unlikely that the `A` and `PTR` records for a client will be on the same DNS server.

The Motivation for Doing DNS Updates from DHCP

A DHCP client can act as a network client without ever having an `A` record or a `PTR` record in DNS. However, there are two problems with this. First, many network servers require that the client's IP address, `PTR` record, and `A` record all match. Second, sometimes two DHCP clients need to exchange information, and the easiest way for their users to rendezvous is by exchanging domain names.

One additional reason for updating DNS is that it makes logging easier, particularly if client IP addresses change a lot. It's much easier to find a client's hostname in a log than it is to figure out the client's IP address and look that up.

Many network services on the Internet check that the client's IP address, `A` record, and `PTR` record all match. They do this by first constructing a domain name from the IP address and using that domain name to look up the `PTR` record. They then use the domain name specified in the `PTR` record to look up the `A` record. If the client's IP address appears in the `A` record, the client is allowed to access the network service; otherwise, it is not.

Although this provides no authentication, it provides a small amount of accountability; if the name server for the `PTR` record and the name server for the `A` record agree about the identity of the client, there's a good chance that if the client causes a problem, you can find the culprit by contacting the administrator of one or both of the name servers.

The second reason for publishing the client's name is that users of computers generally understand domain names but are less likely to understand IP addresses. Also, with DHCP, the client's IP address is not guaranteed to be stable. If the DHCP server and client cooperate, however, the client's name should be stable.

For clients that roam to different administrative domains, this stability requires that the client be able to update its own `A` record and that the DHCP server in the administrative domain to which the client has connected use the domain name supplied by the client to update the client's `PTR` record.

The Domain Name Update Policy

When setting up a DHCP server to do DNS updates, one issue that you must consider is who should update the A record. Both the DHCP server and the DHCP client can, in theory, update the A record. Even if you do not allow clients at your site to update the name server at your site, your DHCP server may encounter DHCP clients from other sites that want to update their own A records at their sites.

In order for a client to update its own A record, it must know its FQDN and it must have permission to update the A record for its FQDN. Most clients are not configured with their FQDN, or they do not have permission to update their own A record. So even at a site that allows clients to update their own A records, you usually need to configure your DHCP server to update the A records of clients that cannot perform their own updates.

The DHCP server can advise a client not to update its own A record, but it can't force the client not to update its A record.

Because clients can specify arbitrary FQDNs, there is no way to configure the DHCP server to update DNS with the FQDN that the client provides; if the DHCP server is going to update an A record, the A record has to be in a domain that is served by a DNS server that your DHCP server has permission to update.

So when deciding how to configure your site, you need to decide the following:

- What to do when the client wants to update its own A record.
- What to do when the client provides a hostname that it wants you to use in the update.
- What to do when the client doesn't provide a hostname.
- Into what domain you want to place client A records when no domain is specified.

The Threat Model for PTR Record Updates

In general, if a client wants to update its own A record, your DHCP server has to store the FQDN the client provides in the client's PTR record. This means that the DHCP server must store a client-supplied string in DNS. Therefore, it is worth understanding whether this is a problem.

The first question to ask is whether there is any sort of risk to your site security in storing a client-supplied domain name on your DNS server? It's also worth considering the positive aspects of honoring the client's request.

There are perhaps three types of risks that you might be taking in accepting a client-supplied domain name and installing it in that client's PTR record:

- The client might use this as an attack on the name server.
- The client might use the presence of the name in DNS as a way to bypass some DNS-based security mechanism.
- The client might use the PTR record to misrepresent its domain name while doing something nefarious.

It is theoretically possible that if your DNS server has a bug, a DNS update of a PTR record with a client-supplied name might compromise the DNS server. Although this is not a threat to be completely discounted, it is certainly a difficult attack to mount. Attacks of this sort generally involve some sort of buffer overflow. But the client is sending the DHCP server a domain name, and the DHCP server is going to validate it. So an attacker has to construct a domain name that is malformed in such a way that it will compromise the DNS server but will get past the DHCP server. This seems unlikely; in reality, the DHCP server will probably be unable to process such a domain name. It is more likely that an attack on the DHCP server itself would succeed than that an attack *through* the DHCP server to the DNS server would succeed.

The second possibility is that the client could insert a name into its PTR record and then use that PTR record to bypass some sort of authentication mechanism. But this is a very unlikely attack as well. The problem with this attack is that sites that use domain-name–based security mechanisms don't trust the PTR record—they check the A record as well. But if the client can update its A record, it is not lying about the name it provided to the DHCP server, so there is no problem.

The third possibility is that the client might put some bogus name in its PTR record before doing something improper. If the improper behavior were detected, the blame for it might be deflected on the rightful owner of the domain name. This is probably the most likely of the three threats described here. It seems unlikely that it would present a problem, however, because any sensible system administrator is going to check the A record as well as the PTR record when trying to figure out where an attack originated. However, there's no guarantee that this will happen—it is certainly possible that people will jump to the wrong conclusions and bad things will happen as a result.

Client-Supplied Hostnames

Microsoft Windows clients and Apple Macintosh clients always provide their own hostnames. Microsoft clients are generally not willing to accept hostnames provided by the DHCP server. So in practice, if you want users of DHCP clients configured by your server to be able to communicate with each other conveniently, you must configure your DHCP server to do domain name updates using the name supplied by the client.

No Supplied Hostname

Sometimes the client doesn't supply a hostname. In most cases, this is because the client is some kind of device that doesn't know its name—for example, a printer. Also, DHCP clients on some operating systems, such as Linux, do not require the user to set a hostname, and if no hostname is set, they do not send a hostname.

In the case of a printer, you need to configure the printer with a hostname so that users can find the printer. In the case of Linux clients, if the user wants his or her client to have a name, he or she will give it one. However, users who are not very knowledgeable might never need to name their clients but might want to be able to use network services that do domain name checking. For such clients, you might want to configure your DHCP server to make up a name when the client doesn't supply one.

Name Clashes

It is quite possible for two users of two different computers to give their computers the same name. The DHCP server checks to see if the name the client provides is in use before it installs the name in the DNS server. If the name exists and belongs to some other client or was not installed by a DHCP server, the client requesting the name doesn't get it. If the DHCP server can, it informs the client that it cannot have the name it wants. The mechanism whereby this is done is described in Chapter 11, in the section "DHCP Client Name Collision."

Dual Booting

In some cases, a computer with one name may run two different operating systems, with two different DHCP clients, and the DHCP clients may provide different identification, but use the same hostname. (Client identifiers are explained in Chapter 16, "Client Identification and Fixed-Address Allocation") In this case, the two DHCP clients get different IP addresses, and the name is associated with only one of those IP addresses. To resolve this problem, you should make sure both DHCP clients send the same identification information or that they send different hostnames.

DNS Update Security

In order to protect your DNS server from unauthorized updates while permitting the DHCP server and perhaps DHCP clients to do updates, you must establish a security policy on the DNS server and configure the DHCP server and clients to use whatever authentication mechanism that policy requires. There are two basic authentication mechanisms you can use: IP address–based authentication and cryptographic authentication.

It is very easy to make cryptographic authentication work, and in our experience it is quite difficult to get IP address–based authentication to work, so we don't recommend that you waste any time on IP address–based authentication; not only is it less secure, but it's actually more difficult to set up.

There are four different ways of using cryptographic authentication in DHCP updates:

- TSIG-based shared secret
- SIG(0)-based public key
- TKEY shared secret
- GSS-TSIG

The simplest of these is TSIG. With TSIG, the client that does the update is configured with a private key, and the DNS server is configured with that same key. The update client signs its update with the key, and the DNS server checks the signature to verify that the update client has the key.

Although TSIG is very simple, it has the classic key distribution problem: You have to arrange to install the private key on the DNS server and on every update client. If the update client is a single DHCP server or a pair of DHCP servers, this is very easy. It is much more difficult to install a different TSIG key on every DHCP client. Also, when the key changes, you have to update every client and server that is using the key.

SIG(0) provides a solution to the key distribution problem by using DNS to store the public part of a key. The update client keeps the private part of the key. The update client needs to register the public key in DNS to establish trust, but this is not too difficult because the key is not secret. Therefore, SIG(0) is a much better choice than TSIG for DHCP clients.

Unfortunately, SIG(0) suffers from the fact that public key cryptography is expensive. The TKEY protocol provides a somewhat complicated solution to this problem: A SIG(0) key is used to establish a shared secret key, and the shared secret key is used to sign updates as with TSIG. The TKEY key is not permanent; if the DNS server forgets the key, the client has to reestablish it, which it can do with the SIG(0) key.

The Microsoft DHCP server, client, and DNS servers all support a keying system called GSS-TSIG. GSS-TSIG uses Kerberos to distribute private keys, which solves the key management problem. Unfortunately, the protocol is complicated, and different versions of Windows implement it differently, so there aren't many non-Microsoft DHCP or DNS products that work with GSS-TSIG. If you are running a Microsoft-only shop, GSS-TSIG is your only choice, but if you are not, it might not be possible for you to use it.

It is beyond the scope of this chapter to describe in detail how all these keying systems work. The ISC DHCP server supports only TSIG updates, so we mostly talk about TSIG here.

NOTE

One important feature that all these four mechanisms share is that they use the time that the update was generated to detect a replay attack. In order for this to work, the update client and the DNS server have to agree on what time it is. If their clocks differ by more than about five minutes, the update fails. So you must synchronize the clocks on the update client and DNS server, using NTP or some similar mechanism.

Generating a TSIG Key

To generate a TSIG key with ISC BIND version 9, you use the `dnssec-keygen` command, as shown in Example 23.1.

Example 23.1

```
dnssec-keygen -a HMAC-MD5 -b 256 -n USER mykey.example.com
```

This generates two output files, both of which contain a key whose name is `mykey.example.com`. The name is very important: It has to be the same on the DNS server and the update client. The TSIG key's name doesn't have to be an FQDN, and it does not appear anywhere in the DNS server (in fact, it can't because it's a secret).

Unfortunately, `dnssec-keygen` does not produce the key in the format that the DNS and DHCP servers want, so you have to extract the secret and put it into the right format. The output files are named `kmykey.example.com.`*`fingerprint`*`.key` and `kmykey.example.com.`*`fingerprint`*`.private`. *`fingerprint`* is the key's fingerprint, which is two numbers, separated by a period. You don't need to use the fingerprint, but you see it in the filename that `dnssec-keygen` uses. The contents of the key file look something like Example 23.2.

Example 23.2

```
mykey.example.com IN KEY 0 2 157 \
   rxOEG1ZXiIRv1vXSvaHFpyxkFq1GUtn4P3uxovceRfQ=
```

The part that looks like a bunch of garbled text followed by an = is the secret key that the two servers must share. To make it into a TSIG key declaration that the ISC DHCP server and ISC BIND can understand, rewrite it as shown in Example 23.3.

Example 23.3

```
key mykey.example.com. {
 algorithm HMAC-MD5.SIG-ALG.REG.INT;
 secret rxOEG1ZXiIRv1vXSvaHFpyxkFq1GUtn4P3uxovceRfQ=;
};
```

This key declaration can be shared between the DNS server and the DHCP server if they are running on the same computer—just put it in a file that both servers can reach and include that file in both servers' configuration files.

NOTE

It is *very* important that the file containing the secret keys for DNS update be readable only to root. If it can be read by other users, those users can learn the secret, and they can use the key to do DNS updates.

Configuring the Servers

You need to configure at least two servers: the DHCP server and the DNS server. You might need to configure more than one DNS server; if the `A` record and the `PTR` record are not on the same DNS server, you have to configure both DNS servers to accept updates. If your DHCP server will update `A` records in more than one domain and those domains are served by separate DNS servers, you have to configure each DNS server to accept updates from the DHCP server. If you will be doing DNS updates from a DHCP client, you also have to configure the DHCP client.

Configuring the DHCP Server to Do Updates

In order for a DHCP server (or a DHCP client, for that matter) to update DNS, it needs to know what DNS server to contact, and it needs to know how to identify itself to the DNS server.

NOTE

In some cases, when both DHCP and DNS are being managed by the same software product, it might not be necessary to explicitly configure the DHCP server or the DNS server with DNS update information. However, in the case of the ISC DHCP and DNS servers, you must configure them explicitly.

DNS is a hierarchy of DNS zones. A *zone* is a portion of the hierarchy that is managed as a unit; a single DNS server is the *master* for any given zone, and each zone has a *start of authority* (`SOA`) record that describes it.

Every domain name is contained in a zone. It is not possible to tell which zone contains a given domain name by looking at the domain name—for example, the name `oxytocin.sci.example.com` might be in the zone `sci.example.com` or the zone `example.com`, or it might be the top of the zone `oxytocin.sci.example.com`.

This is an important problem because in order to update a domain name, the DHCP server must know which zone it's in, and it must know the IP address of the master domain name server for that zone. There are two ways the DHCP server can get this information: You can configure it, or the DHCP server can look it up in DNS.

The ISC DHCP server can do either—if you do not configure it with zone information, it looks up that information in DNS. This is a little bit of extra work, but the DHCP server caches the zone information, so there isn't much performance impact. The main reason to use zone declarations is that you can only specify the DNS update key in a zone declaration.

Example 23.4 shows how to configure the DHCP server with zone information and update keys for a forward zone and a reverse zone. In this example, we assume that the key we generated earlier is stored in a file called `/etc/update-keys`.

Example 23.4

```
include "/etc/update-keys";
zone "sci.example.com" {
  master 10.0.100.17;
  key mykey.example.com.;
}

zone "0.0.10.in-addr.arpa" {
  master 10.0.100.17;
  key mykey.example.com.;
}
```

You must be certain that the zone declaration matches an actual zone in DNS. A zone in DNS is a domain name that has an SOA record. So if the name specified in the zone is not a name that has an SOA record, the zone declaration does not work, and the update fails, probably with a very cryptic error message.

You can check to see if the name you are using is actually a zone by using the `dig` command, as shown in Example 23.5.

Example 23.5

```
dechen% dig sci.example.com. soa
...
;; ANSWER SECTION:
sci.example.com. 3600 IN SOA ns.sci.example.com. \
                             postmaster.example.com. \
                             1999032501 3600 1800 \
                             604800 3600
...
dechen%
```

If you don't see an SOA record in the answer section, the name you have chosen is not a zone, and if you use it in a zone declaration, updates fail.

Configuring the DNS Server to Allow Updates

You must configure the DNS server to allow updates from the DHCP server. This involves configuring the DNS server with the same key you used with the DHCP server and configuring the DNS server to allow updates using that key. Example 23.6 shows an example of a DNS server configuration that corresponds to the DHCP server configuration shown in Example 23.4.

Example 23.6

```
include "/etc/update-keys";
zone "sci.example.org." {
  type master;
  file "sci.db";
  allow-update { key mykey.example.com; };
};

zone "0.0.10.in-addr.arpa." {
  type master;
  file "10.0.0.db";
  allow-update { key mykey.example.com; };
};
```

Completing the DHCP Server Configuration

The ISC DHCP server does not do DNS updates by default. You have to enable them. You can do this with the `ddns-update-style` statement. The ISC DHCP server supports three styles: `none`, `ad-hoc` and `interim`. The `ad-hoc` style is not recommended, so if you want DNS updates, use the `interim` style, as shown in Example 23.7.

Example 23.7

```
ddns-update-style interim;
```

By default, the ISC DHCP server allows clients to do their own DNS updates. To make the DHCP server do updates on behalf of the client, even if the client wants to do its own update, you use the `deny client-updates;` statement.

To specify the domain name in which the DHCP server should install `A` records for clients, you use the `ddns-domainname` statement, as shown in Example 23.8.

Example 23.8

```
ddns-domainname "sci.example.com";
```

To specify a hostname for a particular client, you put a `ddns-hostname` statement in the `host` declaration for that client as shown in Example 23.9.

Example 23.9

```
host foo {
  hardware ethernet 10:02:22:3c:d2:aa;
  ddns-hostname "foo";
}
```

To generate a unique hostname for all clients, you can write an expression that derives the hostname. For example, to generate the hostname based on the client's IP address, you could write a statement like the one in Example 23.10.

Example 23.10

```
ddns-hostname =
  concat ("dhcp-",
     binary-to-ascii (10, 8, "-", leased-address));
```

Finally, to generate a unique hostname for clients that don't specify their hostname but use the client's hostname if the client specifies it, you could write a statement like the one in Example 23.11.

Example 23.11

```
ddns-hostname =
  pick (option fqdn.hostname, option host-name,
     concat ("dhcp-",
       binary-to-ascii (10, 8, "-", leased-address)));
```

The ISC DHCP server has other flags that control DNS updates, but they are generally fairly obscure; they are listed in Appendix B, "ISC DHCP Server Configuration File Reference."

Configuring the DHCP Client to Do Updates

To configure a DHCP client to update its own `A` record, you must do the following:

- Configure the client with its FQDN
- Configure the client to cooperate with the DHCP server
- Configure the client with a key so that it can update its `A` record
- Configure the client to contact the DNS server
- Configure the DNS server to accept the update

In order for a DHCP client to cooperate with a DHCP server in updating the DNS server, it must use the `fqdn` option, which is described in Chapter 11. At the time of this writing, only two DHCP clients support the `fqdn` option: the ISC DHCP client and the Microsoft DHCP client.

It is possible to configure any DHCP client that supports scripting to update its `A` record: Simply configure the client to call a script that updates the FQDN. However, a client that cannot send an `fqdn` option cannot get the DHCP server to update its `PTR` record. The following sections show an example of how to update a client's `A` record from a script.

Configuring the ISC DHCP Client to Update Its A Record

To configure the ISC DHCP client to update its own `A` record, you simply configure it to send the server an `fqdn` option that contains a FQDN and indicates that the client will be doing the update. The ISC DHCP client then assumes that you actually want it to do the update. You can configure this as shown in Example 23.12.

Example 23.12

```
send fqdn.fqdn "dechen.sci.example.com.";
send fqdn.encoded on;
send fqdn.server-update false;
```

Note that the FQDN has a period at the end. This is *absolutely* required; without it, the domain name is not fully qualified, and the DHCP server appends the local domain to it. The client currently requires that you set the `fqdn.encoded` option, although this might be set automatically in the future. The `fqdn.server-update` option tells the server whether the client expects the server to do the update; by setting it to `false`, you indicate that the client intends to do the update itself.

You must also configure the client with a key declaration and a zone declaration, as with the DHCP server. The DHCP client itself only supports TSIG keys, although you can do a SIG(0) update from an external script if you prefer. The DHCP client key configuration is exactly the same as the DHCP server configuration, although you should not use the same key for the client that you use for the server. The code in Example 23.13 shows a key declaration and a zone declaration for a DHCP client that will be updating its name in the `sci.example.com` zone.

Example 23.13

```
key dechen.keys.example.com. {
 algorithm HMAC-MD5.SIG-ALG.REG.INT;
 secret UAEydKm+sEkbG8CpPS4k3oPBW6Az6LybYKdBYHkl6u0=;
};

zone sci.example.com {
  key dechen.keys.example.com.;
  primary 10.0.100.17;
}
```

Configuring the DNS Server for Client Updates

Finally, you must configure the DNS server to allow the client to update its own name. In the DHCP server example, we gave the DHCP server permission to update the *entire* forward and reverse zone. You probably do not want to give any client access to the entire zone in which its name resides. BIND version 9 provides an `update-policy` directive that allows finer-grained control. The configuration code in Example 23.14 gives update clients that have the key `dechen.keys.example.com` permission to update records on the name `dechen.sci.example.com`.

Example 23.14

```
include "/etc/update-keys";

key dechen.keys.example.com. {
 algorithm HMAC-MD5.SIG-ALG.REG.INT;
 secret UAEydKm+sEkbG8CpPS4k3oPBW6Az6LybYKdBYHkl6u0=;
};

zone "sci.example.com" {
  type master;
  file "sci.db";

  update-policy {
    grant mykey.example.com subdomain sci.example.com;
```

Example 23.14 Continued

```
    grant dechen.keys.example.com. name \
           dechen.sci.example.com.;
  };
};
```

In Example 23.14, we have eliminated the `allow-updates` statement that is shown in Example 23.6. The DHCP server is still allowed to update the zone by using the `mykey.example.com` key because of the first `grant` statement in the update policy. The second `grant` statement gives the client permission to update its own name.

Updating the Client's A Record from a Script

If your DHCP client does not directly support DNS updates but does support scripts, or if you want to use SIG(0) with the ISC DHCP client, you must use a shell script. The shell script can use the `nsupdate` command, which comes with BIND version 9.

The ISC DHCP client normally assumes that if you configure it to tell the server it will be doing the update, it should do the update. You can prevent it from doing this by adding the statement in Example 23.15 to your `dhclient.conf` file:

Example 23.15

```
do-forward-updates off;
```

The ISC DHCP client can be made to invoke a shell script whenever it acquires or renews a lease. This script is called `dhclient-enter-hooks`, and it is stored in the `/etc` directory. The script file must be executable. Example 23.16 shows a `dhclient-enter-hooks` script that installs an `A` record for the client:

Example 23.16

```
TTL=120
SERVER=10.0.100.17
ZONE=sci.example.org
HOSTNAME=dechen
KF=/etc/dnskeys/Kdechen.keys.example.com.+157+62188.private
if [ x$reason = xBOUND ] || [ x$reason = xREBIND ]
 then
  nsupdate -v -k $KF > /dev/null << EOF
    server $SERVER
    zone $ZONE
    update delete $HOSTNAME A
    update add $HOSTNAME $TTL A $new_ip_address
    send
```

Example 23.16 Continued

```
  EOF
elif [ x$reason = xEXPIRE ] || [ x$reason = xRELEASE ]
 then
  nsupdate -v -k $KF > /dev/null << EOF
    server $SERVER
    zone $ZONE
    update delete $HOSTNAME A
    send
  EOF
fi
```

This script works with both SIG(0) and TSIG update keys, and it operates in almost exactly the same way as the configuration shown in Examples 23.12 and 23.13. The one difference is that instead of using a `key` statement in the `dhclient.conf` file, the `nsupdate` command uses the private key file generated by `dnssec-keygen`, which, in the example, is placed in the `/etc/dnskeys` directory.

This script does not run, unchanged, with some other DHCP client. In order to make it work with a different DHCP client, you must figure out how the DHCP client signals to the script that a new address has been acquired or that the client is releasing its old address. Also, the ISC DHCP client places the newly acquired IP address in the `new_ip_address` shell variable. Other DHCP clients may pass the IP address in some other way. When you understand how the client does these things, modify the script accordingly and configure the client to invoke the script when it acquires a lease, when it releases a lease, and when a lease expires.

DNS Record Removal

One important problem to consider with DNS updates is what happens when the client's lease expires and the client's IP address is reassigned to a new client. If the client's `A` record is being used by any network services to locate and contact the client, then when the client goes away, if its name remains, those services continue to attempt to contact it at its old IP address.

Not all DHCP servers and clients follow the same policies for DNS record removal. Some DHCP servers simply leave the `A` record in place. The ISC DHCP server automatically deletes the `A` record from the DNS server when the lease expires or when the client releases the lease.

If the client is updating its own `A` record, the situation is more difficult because DHCP clients frequently do not know in advance that they are being disconnected from the network. If the user of the client instructs the client to send a `DHCPRELEASE` message, the client may delete the name before sending the `DHCPRELEASE` message (the ISC client does this). However, if the client computer is unplugged from the network, the DHCP client doesn't know to remove its A record until the connection is gone, and then it's too late.

There is no real solution to this problem, so mobile DHCP clients must not arrange for network services that use protocols that are not cryptographically secure to connect to them.

The Threat Model for Dangling A Records

The threat model for dangling A records is not an entirely academic subject. A sophisticated user might set up network services on a DHCP client machine. A malicious person might set up a computer that watches for the DHCP client machine to be disconnected from the network. When the client is disconnected, the attack computer could take over its IP address and start listening for service connections.

For example, suppose you set up an SMTP listener on your DHCP client. After you disconnect from the network, the attacker could accept incoming SMTP connections on your IP address and intercept your e-mail until your client updates its A record or the lease expires on the server and the server deletes the A record.

There could be another, more subtle, attack: Suppose that the DHCP client is advertising a Web server or and FTP server on its A record. In that case, a malicious person could set up a fake Web server or FTP server with content similar to the DHCP client's content, but with perverted files intended to compromise the security of any machine that uses them. Because it would be listening on the IP address that the client's A record advertises, people who have come to trust that name would connect, assuming they were talking to the server they trust, and would not be as careful in checking the files they downloaded.

So a mobile DHCP client that advertises network services must be careful to use only network services that provide a mechanism whereby the server can prove its identity to clients that connect and where no nonsecured path is available that would appear the same to a naive client. For example, secure HTTP provides a mechanism by which the client can verify the server's identity, but if the user types `http` instead of `https`, the Web browser never checks the identity of the server. SMTP provides an authorization mechanism, but doesn't require that SMTP clients use authentication, so it is unlikely that a bogus SMTP server would be detected.

Time to Live on Client A Records

The Time to Live (TTL) of a DNS record tells a DNS cache how long it can continue to use a DNS record it has retrieved before it has to check to see whether it has changed. The TTL is used to prevent a DNS cache from holding on to stale data that may have changed. The TTL isn't an expiration date on the record—DNS records do not expire. Rather, it is a guess as to the longest time interval during which it is not likely that the record will change.

The TTL on a DNS record should be chosen considering three things: minimizing redundant lookups, minimizing out-of-date answers, and minimizing lookup failures due to unreachable name servers.

DNS caching reduces the load on authoritative name servers. A DNS record with a TTL of zero can't be cached, so it must be looked up on an authoritative name server every time it is used. Imagine how many hits the name servers for the `.com` domain would take if the SOA record for `.com` had a TTL of zero. Every time anybody or any server anywhere on the internet looked up a name in the `.com` domain (for example, `example.com`), the request would have to go to a name server for the `.com` domain. This would amount to millions of queries per second.

On the other hand, a DHCP client is not going to be that popular. It is more likely to receive a query every couple minutes. One other aspect of caching is that it only works if a lot of queries for the same name go through the same cache. So if a DNS client at `example.org` and another DNS client at `dhcp-handbook.com` both try to look up `dechen.sci.example.com`, they each have to consult an authoritative name server because they do not use the same DNS cache.

However, as the user of a particular DHCP client surfs the Internet, the client machine is likely to fetch quite a few Web pages from the same server before moving on to the next server. So it is likely that there is *some* benefit to caching—just not for very long.

On the other hand, as mentioned earlier in the chapter, having the `A` record dangling is not a very good idea. If the client cleanly releases its lease before disconnecting, the client or the DHCP server removes the client's `A` record from the DNS server. The latest time that a record for that client could still be cached is the time the `A` record is deleted plus the TTL on the record. Therefore, you would like to keep the TTL fairly short.

It is not possible to say exactly what the right TTL is; it depends on your site and the kind of users that are using it. Clients that are not likely to move, such as cable modem and DSL customer computers, can safely use quite long TTLs. Clients that move around a lot, such as laptop computers, need short TTLs.

DHCP doesn't provide the DHCP client with a way to give the DHCP server hints about how long a TTL to use, so you have to guess based on your site's particular usage patterns. At a minimum, a TTL of between 10 seconds and 2 minutes should work pretty well for a mobile client, and if your DNS server is not seeing an excessive load, you might want to just assume this TTL value for every client. At a cable modem or DSL ISP, you might need to use a longer TTL to reduce load on your DNS server, and it is probably safe to do so.

To set the TTL to use on DNS updates in the ISC DHCP server, you use the `ddns-ttl` parameter, as shown in Example 23.17.

Example 23.17

```
ddns-ttl 120;
```

The ISC DHCP client currently does not provide a way to set the TTL, although if you use a script to update DNS, you can specify the TTL to the script.

Debugging Problems with DNS Updates

This section lists a series of error messages that the DHCP server or client may print and explains what is the likely cause of each message.

If you look at the domain name in the error message in Example 23.18, you can see that the `sci.example.com` domain name is repeated twice.

Example 23.18

```
Jan 23 16:20:39 server dhcpd: Unable to add forward map from
dechen.sci.example.com.sci.example.com to 10.0.0.1: timed out
```

What has happened in Example 23.18 is that the DHCP client was configured to send its FQDN, but the domain name that you specified was not fully qualified—it did not have a trailing period. To fix this problem, add the trailing period in the DHCP client's configuration, as shown in Example 23.19.

Example 23.19

```
send fqdn.fqdn "dechen.sci.example.com.";
```

You might get a timeout message like the one shown in Example 23.18, but the domain name being updated looks correct. In this case, the update message was dropped by the DNS server. The DNS server usually ignores updates because the update isn't signed or because the signature isn't valid (that is, the secret on the DNS server is different than the secret on the DHCP server or client). So make sure that you've written a correct zone declaration with a correct key reference and that the secret in the key on the DHCP server or client matches the secret in the key on the DNS server. Also make sure that the key name is the same in both places; the key name is what the DNS server uses to figure out which key to use to check the signature.

The message in Example 23.20 indicates that the DNS server was able to confirm that you have the key you intended to use but that key does not authorize you to update the DNS server. This means that you've set up the `update-policy` or `allow-updates` statement on the DNS server incorrectly, so that the key you think should allow to make the update isn't actually configured to allow you to make the update.

Example 23.20

```
Jan 23 16:20:39 server dhcpd: Unable to add forward
map from dechen.sci.example.com.sci.example.com to
10.0.0.1: invalid TSIG key.
```

The message in Example 23.21 indicates that the system clock on the DNS server and the system clock on the DHCP server or client are more than five minutes apart. You must synchronize these clocks in order for DNS updates to succeed. You can do this by using NTP or a similar protocol or by simply synchronizing the servers by hand when they get out of sync.

Example 23.21

```
Jan 23 16:20:39 server dhcpd: Unable to add forward
map from dechen.sci.example.com.sci.example.com to
10.0.0.1: clock skew too great.
```

The message in Example 23.22 indicates that you have written a zone declaration that does not refer to a zone. If you want to update `dechen.sci.example.com` and there is an SOA record for `example.com` but no SOA record for `sci.example.com`, you must write a zone statement for `example.com`, not `sci.example.com`.

Example 23.22

```
Jan 23 16:20:39 server dhcpd: Unable to add forward
map from dechen.sci.example.com.sci.example.com to
10.0.0.1: not a zone
```

Summary

DHCP clients can benefit from having forward and reverse mappings in DNS. Either the DHCP client or the DHCP server can update the client's forward mapping; only the DHCP server can update the reverse mapping. You must have a site policy to determine whether your DHCP server will cooperate with clients that want to update their own forward mapping.

When the DHCP server is updating the forward mapping, it can generate a domain to update, using a hostname or FQDN provided by the DHCP client, and it can also be configured to use a name that is configured by the server administrator. The

DHCP–DNS update protocol resolves naming conflicts in favor of the client that has an active lease with a certain name when another client with the same name tries to get a lease.

Updates can be secured with TSIG, SIG(0), or GSS-TSIG, and TKEY and SIG(0) can be used together to establish TSIG keys in a relatively manageable fashion. You must generate a key and install it on the DNS server and the DHCP server or client.

24

Debugging Problems with DHCP

IN THIS CHAPTER

- The Debugging Process
- Connectivity Problems
- When the Server Does Not Respond
- Server DHCPNAK Message Behavior
- Incorrect Option Values
- The Uniqueness of Client Identifiers
- Dual-Boot Client Systems
- Duplicate IP Addresses
- When a Client Fails to Get a Reserved IP Address

Previous chapters discuss how DHCP works and how to configure DHCP servers and clients. This chapter discusses problems that can occur when DHCP servers and clients are configured incorrectly. It also provides an overview of the process of debugging DHCP clients and servers when they aren't working correctly and describes some specific problems that occur rather frequently. This chapter is strictly about debugging DHCP itself; debugging of the DHCP failover protocol and DHCP–DNS interactions are covered in Chapters 18, "Failover Configuration," and 23, "Configuring DHCP–DDNS Interactions," respectively.

The Debugging Process

The debugging process includes three basic parts:

- Discovering that you have a problem
- Determining what the problem is
- Solving the problem

These parts might seem obvious, but each of these parts can be fairly subtle. Therefore, the following sections describe them in detail.

Discovering That You Have a Problem

A computer that uses DHCP is likely to encounter two fundamental problems:

- The DHCP client may fail to acquire or renew a lease.
- If the DHCP client acquires a lease, the information the server provides may be incorrect.

Depending on which version of the DHCP client you are using and on the nature of the problem, it may be difficult to tell which of these two problems has occurred.

Failure to Acquire or Renew a Lease

Some older DHCP clients, such as the one included in Windows 95, notify the user if they fail to acquire or renew a lease. Unfortunately, most modern DHCP clients assume that if they are unable to get an IP address from a DHCP server, they should use an Automated Private IP Addressing (APIPA) as described in Chapter 21 in the section titled, "When a Client Fails to Get an Address." The clients included in Windows 98, 2000, Me, and XP, and the clients included in Mac OS 9 and later versions all do APIPA autoconfiguration if they can't contact a DHCP server or acquire an IP address.

These clients do not display a dialog box if they fail to acquire an IP address. Instead, they choose an IP address on the 169.254.0.0/16 subnet. The only way to tell that the client chose an autoconfiguration address is to find out what IP address the client has, if any. If the client has an IP address that starts with 169.254, the DHCP client was not able to acquire an IP address from the DHCP server.

To check the IP address that a client acquired on Windows 98, you use the `winipcfg` command. When you do, a dialog box appears, allowing you to see what IP address is assigned to each network interface. In Windows NT, 2000, and XP you use the `ipconfig` command to display a list of interfaces and their IP addresses. In Windows Me you can use either of these commands. On Unix and Linux systems, you can type `netstat -in` to get a list of the IP addresses associated with each interface. On Mac OS 9, the TCP/IP Control Panel sometimes shows the IP address that is configured to a particular network interface, if you select that interface and click on the `Info` button, but it does not always display this information. On Mac OS X, you can use the Network Preferences dialog box to show the IP address of an interface, and you can also use the `netstat` command.

NOTE

DHCP clients on multiuser systems, such as Unix, Linux, NetBSD, and Windows NT, do not necessarily display dialog boxes indicating that a problem exists in acquiring an IP address. Instead, the DHCP client reports the problems through the system error log. To determine that these systems failed to acquire an IP address, you can either look in the error log or check the IP address assigned to each network interface that the DHCP client is instructed to configure. If the network interface doesn't have an IP address, the DHCP client failed to acquire one.

On Unix-like systems, the `syslog` daemon, which is configured through the `/etc/syslog.conf` file, writes the system error log. If you don't know where DHCP client errors are logged, consulting this file and the documentation for the DHCP client should help. On Windows NT, you can run the Event Log program to examine the event log for errors.

Incorrect Information from the DHCP Server

If the DHCP server is providing a lease but is providing incorrect information about network services with that lease, the DHCP client generally cannot tell that this is the case, and it does not display a dialog box or log a message indicating that an error occurred. The only evidence that something is wrong is that the client cannot access some or all network services.

For example, if the DHCP server supplies an incorrect value for the Domain Name System (DNS) server or the default route, the DHCP client cannot resolve DNS names and contact other computers on the network, even though it has a valid IP address. A DHCP client for a diskless workstation (for example, a network computer) that is provided either with incorrect information about the name of its bootfile or with an incorrect IP address for the server that provides that file may display an error message such as "File not found" on startup.

If you suspect that something of this nature is occurring, you should examine the client's configuration in detail to discover the problem. The Windows 95/98 `winipcfg` command displays the IP address of the domain name servers that it receives, as well as the default route, so it is possible to verify that these configuration parameters are correct. To get `winipcfg` to display more information, you can click the `More` button in the `winipcfg` dialog box. DHCP clients for Unix systems generally write the IP addresses of the DNS servers into the `/etc/resolv.conf` file, and you can use the `netstat -r` command to display the routing table, including the default route provided by the DHCP server. On Windows systems that support the `ipconfig` command, you can use `ipconfig/all` to display all the parameters received from the DHCP server that Windows knows how to use. On Mac OS X, the Network Preferences dialog box displays the client's IP address, subnet mask, and default route, but it does not display the domain name server address or the domain name. The file `/var/etc/resolv.conf` may contain the domain name server address and the domain name.

Determining What the Problem Is

After you discover that you have a problem, you must figure out why you have the problem. The client may fail to acquire or renew a lease for four general reasons:

- The client might be unable to get DHCP packets to the server.
- The server might be receiving DHCP packets but unable to get its responses to the client.

- The server might have no IP addresses to allocate to the client or it might be configured not to allocate an IP address for that client.
- More than one DHCP server might exist, and the servers might be configured in such a way that they interfere with each other.

If the information the client receives is incorrect, either the DHCP server is incorrectly configured or more than one DHCP server exists and the DHCP server from which the client acquired its lease is not the right DHCP server for that client.

To determine what the problem is, you need one or more tools. You should have documentation for your DHCP server and have a reasonable understanding of it. You must be able to access the server log files for your DHCP server, you must know the identity of the client that is having difficulty. You might also need a network analyzer to monitor the DHCP traffic between the client and the server.

NOTE

A *network analyzer* is a tool that reads and interprets packets on a network segment. A network analyzer is a program that runs on your computer, such as `tcpdump`, `etherfind`, or `snoop`. You can use a network analyzer to examine the contents of DHCP messages and determine the nature of a problem.

Although you need a network analyzer to debug some problems, you can debug many DHCP problems without one. If you don't have a network analyzer, you can often figure out what is going wrong by looking at the output of the DHCP server and the DHCP client.

Solving the Problem

After you identify a problem, solving it might be easy. For example, if the problem is network connectivity, you need to fix the connectivity problem. If you are running out of addresses, you need to figure out some way to allocate more addresses. However, if the problem is more complex—for example, two DHCP servers are interacting badly—you might need to learn more about the protocol or call in your server vendor for help. If you ask for help from the vendor, you should provide your vendor with as much information as possible; you should not leave out details you think are irrelevant. If you can't figure out the problem, you might be focusing on the wrong details.

The rest of this chapter discusses actual problems you might run into while operating DHCP and how to solve them.

Connectivity Problems

To acquire a lease, a DHCP client must first be able to communicate with a DHCP server. This might seem obvious, but in many cases a DHCP client can't acquire a lease for an IP address simply because it is unable to communicate with the DHCP server.

Most DHCP servers log informational messages when they receive packets from clients. Therefore, the first place to look to see whether the client and server are communicating is the DHCP server log.

The ISC DHCP server logs communication information by using the `syslog` daemon. Where the `syslog` daemon stores the log messages varies on different versions of Unix and Unix-like operating systems. You can find out where your `syslog` daemon stores these messages by looking in the `/etc/syslog.conf` file for a line that indicates where messages of the class `daemon` are logged. You can then use the `grep` command to search that file for entries that include the string `dhcp`; this shows you what is logged. The ISC DHCP server logs routine events at the `info` level, so your `syslog.conf` file must specify that messages at that level should be logged.

The Microsoft DHCP server logs events to one of seven files—one for each day of the week. Each file's name is `DhcpSrvLog.`, followed by the first three letters of the English name for the day of the week, the first letter of which is capitalized. For example, the log for Monday is `DhcpSrvLog.Mon`. Each log file contains a message at the top that explains the format of the entries in the file.

To find an event associated with a particular client, and thus prove that the DHCP server is receiving messages from that client, you must know the information that the server uses in its log message to identify the client. This is generally the client's link-layer address or `client identifier` option. (See Chapter 16, "Client Identification and Fixed-Address Allocation," for details about client identification.)

If you can find a record of the client's request in the server log file, you know that the server is receiving requests from the client. You can then try to figure out why the client isn't receiving a response.

If you do not see some record of the client's request in the server log file, you have found at least one problem. To discover why the server didn't receive the client's request, you must consider the path that a client packet should follow from the client to the server.

Local Connectivity

The first step in the path that a client packet should follow from the client to the server is the client's network interface card and the wiring between the network interface card and the rest of the network segment to which it is connected. If the network interface card is not working, the client cannot communicate on the network. Likewise, if the network wiring isn't working, the client cannot communicate.

NOTE

Remember if the client has been connected to a network segment other than the one to which you think it's connected you may see unexpected results. You should be sure to correctly identify the network segment to which the client is connected.

The first step in determining whether a network card is working is to manually configure the network interface with an IP address and see whether the client can send packets to the local router.

On Unix systems, Unix-like operating systems, and Windows systems you can use the `ping` command for this purpose. Just type `ping` *`xxx.xxx.xxx.xxx`*, where `xxx.xxx.xxx.xxx` is the IP address of the router. On Unix systems, use `ping -n` to avoid a delay in accessing the name server. If you don't get a response, either the network card isn't working or the wiring between the network card and the router isn't working. To determine which of these two is the problem, try connecting the network interface to a network port that's already working for some other computer. If that enables the client to `ping` the router, you have a wiring problem, a bad port on a network switching device, or some other network hardware configuration problem.

If the client can't `ping` the router, even with a network port that you know is working, you should make sure that the cable you're using to connect the network interface card to the network is working by trying a different cable. If none of these tests enables you to `ping` the local router, you might have a bad network card or a bad driver for the network card. If this is the case, contact your operating system vendor or your network interface card vendor. If the problem is a network card, you should try swapping in a different network card of the same type to see whether it works. Again, if this is the problem, you might have to go to your network interface card vendor for help.

If you don't find a problem with the client's connection to its local network segment, the next step is to figure out why. If the DHCP client and server are on the same network segment and the client can, when manually configured with an IP address, `ping` its router, a problem exists with the DHCP server machine or the DHCP server itself. If the DHCP client and server are on different network segments, you must have a relay agent to convey the client's request to the server.

NOTE

One failure mode that you might run into looks a little bit like a network failure but is actually an artifact of a certain kind of Ethernet transport: switched Ethernet. With switched Ethernet, when you configure a client manually, you can `ping` the router and the DHCP server, but when you try to use DHCP to configure the client, the DHCP server never receives a packet from the DHCP client. This can happen with some brands of network switches. The switch simply drops the first packet a node sends, as well as any other packets it sends in the first 30 seconds. Some DHCP clients give up trying to contact the DHCP server before 30 seconds have passed.

Therefore, if you have just installed a new model of switch or if you are using DHCP for the first time on a switched network, you might see these symptoms. If you do see these symptoms, keep in mind that most switches have a way to disable this behavior—or at least shorten the time delay. We can't tell you specifically how to do this with your particular brand of switch, but you will probably find information about this in your switch's documentation; it usually has something to do with fast port startup or spanning tree computation.

Relay Agent Connectivity

When the DHCP client and DHCP server are not connected to the same network segment, the DHCP server does not see a packet broadcast by the DHCP client unless a relay agent is on the network segment to which the client is connected. In this case, the most frequent problem is that the relay agent on the network segment to which the DHCP client is connected either is not configured or is configured incorrectly. Chapter 15, "Configuring a DHCP Server," describes how you can configure DHCP relay agents.

If other DHCP clients on the same network segment are being configured correctly, you do not have to check the relay agent. However, if no other clients are configured on the network segment, or no clients on the network segment are able to acquire IP addresses, it's worth checking to make sure the relay agent is correctly configured.

Server Connectivity

If you can verify the following, then you need to look to the server:

- That the client's connection to its local network segment is working
- That the client and server are on the same network segment, or that the relay agent between the client and the server is configured correctly

If you have a network analyzer you can run on the network to which the DHCP server is connected, it might be worthwhile to make sure that the packet is actually arriving on that network segment.

Determining Whether the DHCP Server Is Receiving Client Messages

Although it might sound obvious, the first thing you should do if you are setting up DHCP service for the first time is to make sure that the DHCP server is actually running. You should check to make sure a DHCP server process exists. If you find that no DHCP clients are getting DHCP service, you might try restarting the DHCP server, even if you see a DHCP server process running.

The DHCP server might not see a DHCP packet that is sent to the correct network and is actually arriving on that network because the DHCP server node is dropping the packet. The node might drop the packet for two common reasons:

- The node is configured with firewall rules that block receipt of DHCP packets.
- There is a bug in the network protocol stack that is running on the DHCP server node.

If you run firewall filtering on a DHCP server node, you should refer to Chapter 22, "Setting Up DHCP in a Small Office," for details about setting this up correctly. Otherwise, you should probably contact your DHCP server vendor to figure out why the server is not receiving packets. Excessive network traffic can also temporarily prevent a client from obtaining an address, but unless your network is badly overutilized, this is unlikely to result in a persistent outage.

After you establish that the DHCP server is receiving the DHCP client's requests, you should see whether the DHCP server is responding. If you see in the log that the server is sending a positive response to the client, but you know that the client is failing to acquire the lease the server is offering, either the server is unable to get its responses to the client or some other server is interfering.

Determining Whether the Client Is Receiving Responses

If the server is sending responses to the client but the client isn't succeeding in acquiring or renewing its lease, the responses may not be reaching the client. This might be happening for a variety of reasons.

The most common reason is that the server is sending the responses to the wrong IP address. DHCP requires that the DHCP server or relay agent send responses to one of two IP addresses: either the IP address that is assigned to the client or to 255.255.255.255. The client may specifically request that the server or relay agent broadcast the response. It does this by setting the `BROADCAST` bit in the `flags` field of the DHCP message. If the client doesn't specifically request a broadcast response, the server or relay agent unicasts the response to the client if it is able to do so. Otherwise, it broadcasts the response.

NOTE

When DHCP was first defined, the protocol specification required that any responses be sent to the newly assigned IP address, avoiding the use of the IP broadcast address. Experimentation with existing TCP/IP implementations showed that some implementations do not accept UDP datagrams with unicast addresses before an IP address is configured. The `BROADCAST` bit was created to enable implementations to request the use of IP broadcast if necessary.

Some DHCP server implementations also broadcast instead of unicast because of limitations in the IP stack of the operating system on which they are running. The ISC DHCP server broadcasts to local clients if it is using the BSD socket API because this API does not specify a standard way of unicasting to clients that can't respond to ARP messages. Otherwise, the server unicasts unless the client specifies otherwise or unless the DHCP relay agent on the network segment to which the client is attached can't unicast.

The IP implementations on some operating systems do not work correctly when a network server tries to send a datagram to the IP address 255.255.255.255. Instead of sending the packet to 255.255.255.255, these operating systems send the datagram to the subnet broadcast address. For example, if the DHCP client is on a subnet numbered 10.117.221.0, with a subnet mask of 255.255.255.0, a server or relay agent with this problem sends the response to 10.117.221.255 instead of to 255.255.255.255. You can see whether this is happening by using a network analyzer connected to the same network segment as the client to examine the response from the server. Appendix F, "DHCP Server and Operating System Versions," lists some of the operating systems on which this can be a problem and describes some workarounds.

This is a problem because when the server sends a message to a client on the subnet broadcast address, the client has no way to tell that the message is a broadcast message. Clients that accept unicast responses accept the message anyway, but clients that can handle only broadcast responses do not receive messages sent to the subnet broadcast address. So you might see that some DHCP clients have no trouble getting addresses but others are never able to get IP addresses.

If the server is on a different network segment than the client, the relay agent must relay the packet back to the client. The relay agent puts the IP address of the interface on which it received the DHCP message from the client in the `giaddr` field. If the IP address in the `giaddr` field is wrong, either the DHCP server is not able to get its response to the relay agent or the relay agent is not able to deliver the message to the client.

NOTE

Relay agents are sometimes configured as firewalls or as network address translators. If the DHCP server and client are on opposite sides of a firewall or network address translator, it is possible that the DHCP server will not be able to send packets to the IP address specified in the `giaddr` field because the `giaddr` field shows the IP address of the network interface on the client's side. It is also possible for the client to have trouble unicasting to the server after it has acquired an IP address and for the server to have trouble unicasting to the client. If you are running a configuration like this, you must provide some means for the client and server to communicate and for the server to communicate with the relay agent.

If the relay agent fails to honor the `BROADCAST` bit or sends the response to the wrong IP destination address, the client will not receive it. You can use a network analyzer to see exactly what the relay agent is sending to the server and to the client.

Some clients can accept only broadcast replies from the DHCP server prior to receiving an IP address, but they do not set the `BROADCAST` bit in the `flags` field of the packet. This does not comply with the DHCP protocol specification.

If a network analyzer indicates that the `BROADCAST` bit is not set in a `DHCPDISCOVER` packet—and the DHCP client doesn't receive an offer, yet it appears as though the DHCP server sent one— the client might need a broadcast response, even though it is not setting the `BROADCAST` bit. If you can, try to configure the DHCP server to set the `BROADCAST` bit in outgoing messages, despite the setting of the `BROADCAST` bit in incoming message. If you do this, and the client can then receive `DHCPOFFER` and `DHCPACK` messages from the server, the client probably has this bug.

Some clients accept only a unicast packet and do not accept broadcast packets. These clients typically signal their intentions by not setting the `BROADCAST` bit. However, if your server doesn't support unicast on the platform on which it is operating, it cannot honor the client's setting of the `BROADCAST` bit.

NOTE

Unicast support in DHCP servers is very operating-system specific and is not possible on some operating systems. RFC 2131 specifies that a server should unicast when the broadcast flag is not set but requires that a client must be able to handle a broadcast, even if it did not set the `BROADCAST` bit. However, some clients do not fully conform to the DHCP specification on this point.

When the Server Does Not Respond

If a server is receiving requests from a client but is not responding to them, you should look in the server's log file to see why. Most DHCP servers log some kind of error message when a client's DHCP request is received but no IP address is ultimately offered to the client. If you find a useful log message that describes the problem, you need only fix it. If you cannot locate the problem from the log entries, you can check for a couple potential obstacles, as described in the following sections.

No Available IP Addresses

In order for the DHCP server to assign an IP address to the client, it must have an address that is available for dynamic assignment or a static assignment for the client that is valid on the network to which the client is connected. The DHCP specification requires that a server not respond to a client if the server has no addresses left to assign. If the server does not respond when the client requests an IP address, it may be because no address is available to assign to the client.

Server Not Configured for Client's Network Segment

Another potential problem is that the server does not have a configuration entry for the network segment to which the client is attached. The DHCP specification requires that a server simply ignore requests for addresses from network segments for which it is not configured. If you don't include the client's network segment in the server's configuration, the server does not respond to the client's request.

BOOTP Clients and DHCP Servers

Most DHCP servers respond to BOOTP messages, but they must be explicitly configured to offer such support. If a device uses BOOTP and is unable to get an IP address from a DHCP server, you should first ensure that the DHCP server is configured to offer services to BOOTP devices. Then you should ensure that either a static IP address that is valid on the network segment to which the client is attached is configured for the BOOTP client or that the server is configured to support dynamic address allocation for BOOTP clients.

Dynamic BOOTP enables the DHCP server to assign an available address to a device by using BOOTP the first time that server sees a request from that device. For compatibility with the BOOTP specification, the server gives that device an unlimited lease on the IP address. From then on, whenever the DHCP server sees a BOOTP request from that device, as long as the device is still connected to the same network segment, it sends it the same IP address.

Server `DHCPNAK` Message Behavior

DHCP specifies that if a DHCP server receives a `DHCPREQUEST` message from a DHCP client for an IP address that it knows to be incorrect, it must send a `DHCPNAK` message to the client. This causes the client to stop using that IP address and go back into the

INIT state, after which it should acquire a different IP address. If a DHCP server sends a DHCPNAK message when it shouldn't or doesn't send one when it should, the client may not be able to acquire or renew a lease.

The Server Sends DHCPNAK Message When Inappropriate

If two DHCP servers are providing DHCP service for a single network segment, both DHCP servers must agree on the subnet configuration and on any static IP address assignments they have for that network segment. If they are performing dynamic IP address allocation, they must not be allocating from the same set of IP addresses unless they have some way to communicate with one another about which addresses they assign (for example, the DHCP failover protocol).

If two DHCP servers do not agree on the configuration of a given network segment, it is likely that each server is preventing clients from completing the DHCP configuration process with the other server. If you have access to the logs of both DHCP servers, you can compare them to see whether this is happening. If a client sends a DHCPDISCOVER message, gets a DHCPOFFER message from one server, sends a DHCPREQUEST message for that IP address, and gets a DHCPNAK message from the other server, one of the two servers is not configured correctly; the DHCPNAK message prevents the DHCP client from acquiring an IP address from the other server.

Rogue DHCP Servers

If you think you have only one DHCP server configured to support a given network segment, you might be wrong; perhaps some user of that network segment tried to configure his or her own DHCP server and got the configuration wrong. You might be able to determine that this happened by running a network analyzer on the network and watching the DHCP packets that are exchanged with a client. If you see DHCP packets coming from some IP address on which you aren't aware that a DHCP server exists, you have a rogue DHCP server on your network.

Configuration Drift Between Cooperating DHCP Servers

A common configuration error can occur when two servers are providing service on the same network segment. As the DHCP server begins to run low on addresses for a network segment, a subnet is added on one server to make more addresses available, but the administrator forgets to add that subnet to the other DHCP server. The new subnet is configured on the same network segment as the old subnet. Thus, one server's idea of what IP subnets are configured for the network segment is different from the other server's idea.

This configuration error does not show up until some DHCP client is offered an address on the newly allocated subnet. At that time, the DHCP client broadcasts a

DHCPREQUEST message for the offered IP address. The server that offered the IP address responds with a DHCPACK message. Because the other server does not have the new subnet in its configuration, it decides that the IP address the DHCP client is requesting is invalid for the network segment to which it is attached. The protocol requires DHCP servers to send DHCPNAK messages to clients whose configurations are incorrect for the network to which they are attached, so the second DHCP server sends a DHCPNAK message to the client.

This can be very difficult to detect because most DHCP clients accept the first response they receive. Sometimes the client receives the DHCPACK message first and sometimes the client receives the DHCPNAK message first. If the client receives the DHCPACK message first, it uses the newly assigned address; if the DHCPNAK message arrives first, the client starts its initialization process again. If you think you have a problem with conflicting server configurations, you can use a network analyzer to determine which server is sending a DHCPACK message and which is sending a DHCPNAK message. You can also figure this out by comparing the server log messages for that client.

The solution to this problem is to configure the second server to be aware of the existence of the newly allocated subnet on the first server. If you do this, the server knows that addresses on the new subnet are valid for the network segment, and it remains silent when it sees a DHCPDISCOVER message from the DHCP client that received an IP address on the new subnet from the first DHCP server. To avoid this problem, you might want to generate server configuration files from a common source so that when a change is made to one server, it's automatically propagated to the other server.

Server Fails to Send DHCPNAK Messages When Appropriate

DHCP servers are expected to validate IP addresses that are requested by clients. If a client requests an IP address that is not valid for the network segment to which it is attached, the DHCP server is expected to send a DHCPNAK message in response to the client's request. This DHCPNAK message causes the DHCP client to move to the INIT state. The client then broadcasts a DHCPDISCOVER message so that the local DHCP server can offer it a valid IP address.

If a DHCP server fails to send a DHCPNAK message when it should, the DHCP client may continue to use the IP address if time remains on the lease. If the IP address is not valid for the network segment to which the server is connected, the client cannot use the network.

NOTE

To protect against misconfiguration by naive users, the ISC DHCP server does not send `DHCPNAK` messages to DHCP clients that send `DHCPREQUEST` messages for IP addresses on the wrong network unless it is positively configured to do so, using the `authoritative` option. The Microsoft DHCP server should always send a `DHCPNAK` message in this situation, so this isn't a problem for installations in which the Microsoft server is used. If you are using the ISC DHCP server, you should make sure to include `authoritative;` at the top of the configuration file.

Incorrect Option Values

Sometimes a client receives incorrect option values from the server when it gets a lease for an IP address. Most frequently, this is the result of a configuration error in the server and the association of the various option values with the different subnets and DHCP clients.

If the server doesn't provide some way to report the option values that it sends, you can use a network analyzer to display the option values in the DHCP messages and diagnose the problem. The DHCP client may also send a `parameter request list` option, specifying the parameters it is expecting. The treatment of this list varies from server to server, and the server and client may not always agree on what this list means. In such a situation, the server might not send the client all the options that it needs in order to operate.

If you use a network analyzer and determine that the server is not sending all the options that the client is requesting, you might have to override the DHCP `parameter-request-list` option on the server.

The Uniqueness of Client Identifiers

For DHCP to operate correctly, DHCP client identifiers must be unique within each administrative domain. If two clients choose the same client identifier, the server cannot differentiate between the two clients. In many cases, the user does not have control over the identifier the DHCP client chooses to send. In many of those cases, the client identifier is based in some way on the client's link-layer address, which should guarantee uniqueness.

However, for cases in which the user can configure the client identifier, it is important that he or she be very sure that it is unique because it is the identity, or key, that the DHCP server uses to tell one client from another. This problem is discussed in detail in Chapter 16.

Dual-Boot Client Systems

A *dual-boot client system* is a computer on which a user is switching between two different operating systems. For example, you might have an Intel-based system with a partitioned disk that has Windows loaded in one partition and Linux loaded in another partition. You can choose which operating system to use when the system first starts up. Because the two operating systems are completely independent of one another, they use two separate DHCP clients, each with its own state.

In some cases, those two different operating systems generate an identical client identifier (typically from the link-layer address); the DHCP server is then unable to distinguish one operating system from the other. Each operating system may, however, have a different expiration time for the lease, even though they share the same IP address.

Sometimes the operating systems on dual-boot systems have different DHCP client identifiers. Most commonly, one operating system generates a client identifier from the link-layer address and the other does not use a client identifier at all. In this case, many DHCP servers see these two operating systems as fundamentally different DHCP clients and give each of them a different IP address. Treating a dual-boot system as two different DHCP clients may be an advantage if, for example, one of the operating systems provides services that the other does not provide. However, because each operating system has its own IP address, each dual-boot system consumes two addresses, which might exhaust the pool of available addresses.

Chapter 23, "Updating the DNS with DHCP," describes problems with dual-boot machines that take advantage of DHCP–DNS dynamic updates. Briefly, if you are using dual boot with dynamic DNS updates, you should be sure that either the two operating systems use different client identifiers and different hostnames, or that they use the same client identifier and the same hostname.

Duplicate IP Addresses

When two network interfaces are configured with the same IP address, problems result. A DHCP server does not give one client a lease on an IP address while some other client still holds a valid lease on that same address. However, a network user could misappropriate an IP address without using DHCP by knowing the network number and subnet mask and testing each valid IP address on the subnet until he or she found one that is not in use. If the DHCP server subsequently assigned that IP address to a DHCP client, both the client and the manually configured computer would be using the same IP address simultaneously.

Some DHCP servers try to avoid allocating duplicate IP addresses by sending an ICMP echo message to the address before allocating it to a client. If the server receives a response, the address is already in use. The server marks the address as unusable and tries another address for the client. This check works only if the other computer using that IP address is operational at the time that the DHCP server checks the address. If the rogue user's machine is active when the server checks, the server notifies the system administrator through a log message that a conflict exists. If the machine is not active, the user of the DHCP client notices the problem when the rogue machine is again powered on.

DHCP clients are expected to use ARP to check a newly assigned IP address prior to using it. If a client finds some other system using the address it was assigned, it sends the DHCP server a `DHCPDECLINE` message and then goes back to the `INIT` state to acquire a different IP address. If a client doesn't support `DHCPDECLINE` messages, you might need to manually mark the IP address as unusable through the server configuration and then restart the client that experienced difficulty.

Duplicate-address situations can result from a variety of reasons with DHCP, and they are, in general, completely preventable. Keep in mind the following:

- Do not configure a DHCP server to dynamically allocate an IP address that has been manually assigned to a computer on the network.
- When using redundant DHCP servers (as described in Chapter 17), do not supply the same IP address to two different servers to be dynamically allocated.
- When using the DHCP failover protocol, make sure that any address pools that you configure with the same addresses on two servers are actually configured as shared pools.
- Do not lose your DHCP server's database of active leases.

If you keep these things in mind, you should have problems with duplicate IP address allocation only if you have users who refuse to follow your address management policy.

When a Client Fails to Get a Reserved IP Address

Most DHCP servers enable you to reserve a particular IP address for a particular client. The DHCP server does not offer that IP address to any other client, and if the client appears on the network segment where the reservation exists, it should get that IP address.

For cases in which redundant DHCP servers are used, as described in Chapter 17, both DHCP servers must be aware of the static allocation. Otherwise, the DHCP server that is not configured with the static allocation might make a dynamic IP address assignment to the client. This can be a problem if the client depends on having a consistent IP address because the dynamically allocated address is almost certainly different. Therefore, when you're configuring redundant DHCP servers, the dynamic IP addresses must be different on the two DHCP servers, and the static IP address allocations must be the same.

Summary

Successful DHCP client configuration requires that several parts of a network work correctly. Specific characteristics of clients and servers, failure or incorrect configuration of network components, and even uncooperative users may all cause DHCP to fail. If you suspect a problem with DHCP, you should first examine your server's log files to see whether you can find some clues about the problem there.

When working on problems with DHCP, you should first ensure that the DHCP client and DHCP server can communicate with each other. If the client and server are on different network segments, you must ensure that a correctly configured DHCP relay agent is supporting the client's network. You can use a network analyzer on the client's network segment to confirm that the client is sending DHCP messages and that the server's responses are reaching the client. You can also check whether the replies from the server are being delivered to the correct IP address.

Configuration problems on DHCP servers can often cause difficulties. A DHCP server must be configured with information about the network segments to which clients may be attached. The server must also have addresses available to assign to clients on those network segments.

Redundant DHCP servers are especially prone to configuration problems. You should be very careful to ensure that all servers are configured with the same list of subnets, that the dynamic address pools are separate, and that the reservations for IP addresses are identical. Only one server should support dynamic BOOTP for any subnet.

Although no one can anticipate every failure scenario possible with a DHCP service, this chapter describes several common problems. If you follow the strategies described here, you should be able to diagnose and fix any DHCP problem.

25

DHCP for IPv6

IN THIS CHAPTER

- An Introduction to IPv6
- The Motivations for DHCPv6
- The Design of DHCPv6

The IETF has developed a new version of IP, which is intended to replace the current version of IP, version 4, with as few changes as possible to the rest of the TCP/IP suite. This new version, called *IPv6*, is named after the version number stored in the `version` field of the IP header. Version 4 of IP (defined in RFC 791), is sometimes referred to as *IPv4* to differentiate it from IPv6. Early versions of IPv6 were called *IP next generation* (IPng) after the popular television show *Star Trek: The Next Generation*.

NOTE

IP version numbers 0, 1, 2, and 3 are unassigned. IP version number 5 was assigned to Internet Stream Protocol Version 2 (ST II; see RFC 1819) before IPng was developed. Thus, when the IETF settled on the preliminary design of its new version of IP, IANA assigned it version number 6.

Several motivations exist for the development of IPv6. One of these motivations—to improve several aspects of IP addressing—is directly related to DHCP. The changes to IPv6 are significant enough to warrant the development of a new version of DHCP for IPv6, called *DHCPv6*.

NOTE

For clarity, this chapter explicitly refers to IP version 4 as IPv4 and to DHCP for IPv4 as DHCPv4. Elsewhere in the book, these protocols are referred to simply as IP and DHCP.

This chapter discusses the differences between IPv4 and IPv6 that affect DHCP, as well as the details of DHCPv6.

NOTE

As this book was being written, the DHCPv6 specification was documented in an Internet Draft. The details of the protocol might change before they are accepted as an Internet Standard. Check `www.dhcp.org` for the latest details about DHCPv6.

An Introduction to IPv6

This section presents a brief introduction to IPv6, including features of IPv6 that are directly connected to DHCP functions. The IPv6 documents define several terms that describe components of an IPv6 internet. Some of those definitions are included here because they are used in some of the IPv6 mechanisms discussed in this chapter:

- **Node**—Any device that implements IPv6
- **Router**—A node that forwards IPv6 datagrams
- **Host**—A node that is not a router
- **Link**—A mechanism (for example, Ethernet) through which nodes can exchange datagrams at the link layer
- **Interface**—A node's connection to a link
- **Address**—An IPv6 identifier for an interface
- **Prefix**—The initial bits of an IPv6 address that identify a link; equivalent to a network number in IPv4

IPv6 Addressing

Each IPv6 address contains 128 bits and, unlike the Class A, B, and C addresses in IPv4, an IPv6 address has no inherent structure. Instead, IPv6 addresses are organized into types by *prefix*. The prefix is identified, when necessary, by a 1-byte *prefix length*, which gives the number of bits in the prefix.

At present, most of the IPv6 address space is reserved for future use. Prefixes are allocated for unicast addresses, addresses with limited scope (described in more detail in the section "Address Types and Scoping"), and multicast addresses. Also, a compatibility mode exists in which IPv4 addresses can be represented in the IPv6 address space.

Because of their length, IPv6 addresses use a different format than IPv4 for textual representation. IPv6 addresses are written in hexadecimal notation, in 16-bit groups separated by colons. This notation is known as *colon hexadecimal,* or *colon hex*. The colon hex format includes two additional notational shortcuts: Any leading zeros from the hexadecimal representation of each 16-bit group can be dropped, and a pair of colons can indicate a string of groups that contain zeros (`::`). The following is an example of an IPv6 address:

```
2000:0:0:122:C34:3F:54CA:B
```

Several addressing mechanisms in IPv4 are formalized in IPv6. In IPv4, the address 0.0.0.0 can be used as a source address during initialization when the computer is not assigned an address, and the address 127.0.0.1 is used as the loopback address. IPv6 also defines an address that consists of all zeros, 0:0:0:0:0:0:0:0, for use during initialization, but in IPv6 the loopback address is 0:0:0:0:0:0:0:1. In IPv6, an interface can have multiple addresses. Finally, the private addressing scheme for IPv4 that was introduced in RFC 1918 is extended in RFC 2373 for IPv6 to define scoped addresses that are restricted to a node's local link or site.

NOTE

RFC 1918 reserves several IP addresses for use in private internets. Organizations can use these addresses for any internal networks, and computers attached to these networks can use IP addresses from the IP addresses without coordinating with any number assignment authority, such as IANA. The addresses reserved in RFC 1918 include one Class A IP address, 10.0.0.0; 16 Class B addresses, 172.16.0.0–172.31.0.0; and 256 Class C addresses, 192.168.0.0–192.168.255.0.

Use of these private addresses postpones the imminent shortage of unique IP addresses by allowing organizations to reuse parts of the IP address space on their internal networks. The primary drawback is that a computer using a private address cannot exchange packets with a computer outside its own internal network.

Address Types and Scoping

In IPv6, an address has three possible scopes:

- **Global**—A global address is unique among all addresses on the Internet and is deliverable (that is, can be routed) to any Internet destination.
- **Site-local**—A site-local address is unique among all addresses within a site and is deliverable to any destination within the site. A *site* is defined as an internet that has a boundary defined by routers that will not forward IPv6 datagrams to site-local addresses.
- **Link-local**—A link-local address is unique among all addresses on the link (that is, the local network) and is deliverable only to destinations on the same link.

IPv6 also introduces the *interface identifier* (RFC 2373), a 64-bit value that can be generated from an interface link-layer address. For example, to construct an interface identifier for Ethernet hardware, you insert $fffe_{16}$ (16 one bits) between the first 3 bytes and the last 3 bytes of a 48-bit Ethernet address to form an EUI-64 identifier. Then, you form the interface identifier by complementing the universal/local bit, which is the next-to-lowest order bit of the first octet of the EUI-64 identifier. Interface identifiers are then used with a specific prefix to generate a 128-bit link-local address. For example, the Ethernet address 0:80:3e:6d:31:ea:

```
00000000 10000000 00111100 01101101 00110001 11001010
```

is translated to the interface identifier 02:80:3e:ff:fe:6d:31:ea:

```
00000020 10000000 00111100 11111111 11111110 01101101 00110001 11001010
```

NOTE

The technique for creating the interface identifier for an interface is defined in the document that specifies the techniques for transmission of IPv6 datagrams over the particular interface hardware (for example, RFC 2464, "Transmission of IPv6 Packets over Ethernet Networks," in the case of Ethernet hardware).

RFC 2464 describes the use of the EUI-64 identifier in creating an interface identifier for an Ethernet interface and the way in which an EUI-64 identifier is constructed from a 48-bit IEEE 802 address.

Scoping and Multicast Addresses

Scoping applies to multicast addresses as well as to unicast addresses. A 4-bit `scope` field defines several classes of increasing scope, including link-local and site-local addresses. Thus, it is possible to define a multicast address that can be used at different sites without conflict. For specific services or technologies using multicast, a single multicast address can be defined for use at all sites.

IPv6 does not include a broadcast address like the 255.255.255.255 address in IPv4. Instead, the address FF01:0:0:0:0:0:0:1 is reserved as a link-scoped multicast address for all nodes on a link, and FF02:0:0:0:0:0:0:1 is reserved for all nodes within a site.

IPv6 Auto-Configuration

One of IPv6's design goals is to enable the automated configuration of IPv6 devices. Several specific mechanisms—some of which are formalizations of IPv4 practices and others of which are new to IPv6—enable computers that use IPv6 to determine or obtain an IP address without manual configuration or intervention by a network server.

Link-Local Addresses

In simple network configurations with no routers, an IPv6 node can use a link-local address with no knowledge of a local IP address. The node selects an address that is not in use by other nodes on its network.

NOTE

The scenario of a single network with no connections to other networks is often called the "dentist's office" network. This describes an ad hoc network created by nonexpert users plugging some computers into a hub. Why dentists were chosen as the canonical nonexpert users is unclear, but the name has stuck.

The standard technique for a computer to generate an IPv6 link-local address is to combine the link-local address prefix with the interface identifier. A dentist's office network need not be configured with a network number, and computers on that network can use a link-local address to contact other computers on the network.

Of course, this automatic process does not configure DNS servers, DNS names, or other services. It does, however, guarantee that computers on an isolated network can communicate with each other and support other manually or automatically configured services.

Although the link-local address generated according to these rules is quite likely to be unique, a small chance exists that some other device on the same network is already using the same address. To check the use of its generated link-local address, a device using IPv6 transmits a request on its local network, asking whether other devices are already using that address. If the device receives no responses to that query, it can safely begin to use the new address.

Prefix Advertisements

Link-local addresses are useful only in communicating with other computers on the same link. To exchange data with computers on other networks, a computer must have a site-local address or a global address. A computer that uses IPv6 can pick a site-local address or a global address for itself by combining the network number or prefix assigned to the local network with the computer's own interface identifier. However, without other information, a computer does not know the prefix for its local network. IPv6 *router advertisement* messages, transmitted by routers to announce their availability on a link, carry prefix numbers for a link. Computers that receive these messages can find out the prefix for a link and then generate an appropriate address.

Fragmentation and Path MTU Discovery

One last difference between IPv4 and IPv6 is that routers do not perform fragmentation on IPv6 datagrams. Only the source of an IPv6 datagram can fragment that datagram. Thus, a computer must know the smallest MTU of all the links along the path to the destination, which it can determine by using path MTU discovery. Conceptually similar to path MTU for TCP in IPv4, an IPv6 node must use path MTU discovery for all destinations, including UDP and TCP, to avoid losing packets on links with small MTUs. To minimize the need for fragmentation, hardware technologies must support an MTU of at least 1,280 bytes for use with IPv6.

The Motivations for DHCPv6

Some difference of opinion exists in the IPv6 community about the need for DHCPv6. A rich set of auto-configuration mechanisms, along with the capability to control network prefixes through router advertisements, means it is possible to have auto-configured operation of IPv6 devices without a centralized address administration service.

Even though formal auto-configuration mechanisms exist in IPv6, some installations still require centralized management and control of IPv6 address allocation and protocol parameter configuration. Other sites might want to provide hosts with configuration parameters such as a list of DNS servers through DHCP. Therefore, the Dynamic Host Configuration Working Group (DHCWG) has developed a new version of DHCP.

The Design of DHCPv6

DHCPv6 retains the client/server architecture of DHCPv4, and many of its server functions are similar to those in DHCPv4. At the same time, some key differences exist between DHCPv6 and DHCPv4.

Differences Between DHCPv6 and DHCPv4

The authors of DHCPv6 took advantage of the opportunity to start with a clean design slate for DHCPv6 by adding new features to the DHCP service, using the new IPv6 addressing features, and dropping backward compatibility with BOOTP.

Using Multiple Addresses for an Interface

One key difference between the two versions of DHCP is that DHCPv6 manages multiple addresses for each interface. When DHCPv4 was designed, few (if any) TCP/IP stacks could be configured with more than one IP address. As a simplifying assumption, DHCPv4 was designed with the constraint that each managed interface could be assigned only one IP address.

NOTE

The DHCPv4 specification allows only a single address for each separately identified interface. Because clients can use distinct values in the `client-id` option, it is theoretically possible to allocate more than one IP address to a single physical interface by using DHCPv4. In practice, the DHCP specification does not define how to assign multiple IP addresses to a physical interface, and most DHCPv4 clients cannot support more than a single DHCP address on an interface.

IPv6 allows for multiple addresses on each interface. This feature enables the use of link-local, site-local, and global addresses through the same interface as well as *virtual hosting*, which provides for multiple copies of a service—for example, multiple Web servers—through a single physical interface.

DHCPv6 enables a client to request multiple addresses for an interface. The address requests might come as separate DHCPv6 messages, asking for the addresses as required by individual application servers on the DHCPv6 client. The DHCPv6 server implementation manages the addresses in sets called *identity associations* (IAs).

Using Link-Local Addresses

In DHCPv4, using UDP and IP before the client is assigned an address is accomplished by using the all-zeros source address and link-level broadcast. The IPv6 link-local address auto-configuration mechanism provides a valid IPv6 address to the client, which the client and server or relay agent use to exchange messages on the link they share.

Managing Lifetime Addresses

Another difference between DHCPv4 and DHCPv6 is the mechanism through which the assignment duration of addresses is managed. In IPv6, an address has an associated *valid lifetime*, which defines the time through which the address can be used. This lifetime is related to the DHCPv4 lease, and the network administrator uses it to control the length of time over which a client can use an address.

Using Fewer Protocol Configuration Parameters

At present, the DHCPv6 specification defines far fewer options than DHCPv4. IPv6 has fewer configurable parameters than IPv4, and SLP for IPv6 provides many of the application service addresses defined as options in DHCPv4.

Using Multicast Addresses in DHCPv6

DHCPv6 defines two multicast addresses that are reserved for the use of clients, servers, and relay agents:

Name	Use
`All-DHCP-Agents`	Clients to send messages to servers and relay agents
`All-DHCP-Servers`	Relay agents to forward messages to servers

Changes in Message Formats and Client Identification

DHCPv6 uses a new format for protocol messages. The new format has a fixed-format header that is smaller than the DHCPv4 message header. DHCPv6 uses options in which the option code and length fields are 16 bits long, which allows for many more options and which accommodates more data in each option than in DHCPv4.

A client is identified in DHCPv6 by a *DHCP unique identifier* (DUID). The client's DUID remains unchanged through the lifetime of the device (for example, even if the interface hardware in the device is changed), so a DHCPv6 server can reliably identify a DHCPv6 client. The DHCPv6 specification defines several techniques through which a device can generate a DUID that guarantees that each DHCPv6 client has a unique DUID.

Client/Server Transactions in DHCPv6

This section describes the specific message exchanges that take place between DHCPv6 clients and servers. These transactions are similar but not identical to the transactions in DHCPv4.

Initial Assignment of Addresses

A DHCPv6 client obtains an initial address by exchanging messages that are similar to the initial messages in DHCPv4. The client uses multicast to send a message that locates available DHCP servers. This message corresponds to the `DHCPDISCOVER` message that is broadcast by a DHCPv4 client to the local broadcast address. A DHCPv6 client locates only servers with the initial message. Any address assignment is performed in the second transaction between the client and server.

The client begins by sending a `Solicit` message to DHCPv6 servers or relay agents on its local link, looking for an available server. This message is sent to the `All-DHCP-Agents` multicast address. Any servers that choose to respond to the initial discovery message reply with an `Advertise` message to the client, through a relay agent if necessary. The client then selects a server and sends a `Request` message, asking for any addresses and other configuration information. Finally, the chosen server responds with a `Reply` message that contains the requested parameters.

When it does not have a valid site-local or global address, the DHCPv6 client uses its link-local address as the source address. The DHCPv6 server or relay agent on the same link as the client uses that link-local address to unicast reply messages to the client.

Obtaining Additional Addresses

DHCPv6 defines the IA as a mechanism through which clients and servers manage IPv6 addresses. An IA contains one or more IPv6 addresses that are managed as a group. Individual addresses can be added to the IA over time, and an address is deleted from the IA when the valid lifetime for the address expires.

A client can subsequently contact a server directly with requests for additional IPv6 addresses to be assigned to a particular IA. The server keeps a list of all the addresses assigned to an IA. Each of these addresses has an independent valid lifetime.

Extending Leases

Similarly to IPv4, the client can send `Renew` messages to the server to extend the lifetime of addresses in an IA. And, as in DHCPv4, if the server fails to respond within some time, the client may send `Rebind` messages to all servers to extend the lifetimes. DHCPv6 defines T1 and T2 in the same way as DHCPv4 (see the section "Extending a Lease" in Chapter 8, "DHCP Message Exchanges"), so the server can control the times at which the client contacts the server to extend address lifetimes.

Duplicate Addresses

When a DHCPv6 client receives an address assigned by a server, the client uses the IPv6 *duplicate address detection mechanism,* which is part of the Neighbor Discovery Protocol (RFC 2461), to confirm that the address is not already in use by another node on the same link. If the client does find that the address is already in use, it sends a `Decline` message to the server and discards the address.

Releasing Addresses

A DHCPv6 client can return to a server addresses that it no longer needs. The client can choose to return all addresses at once or it can return some addresses and retain others. In DHCPv6, the server acknowledges returned addresses to the client, using the client's link-local address either directly or through the relay agent on the client's link.

Configuring Clients Without Address Assignment

Because of stateless address auto-configuration, many IPv6 hosts do not require explicit address assignment from a DHCP server. Those hosts need other information such as the addresses of DNS servers. A DHCP client can obtain configuration information without being assigned an address by sending an Information-request message. A server responds with a Reply message that contains the configuration information requested by the client.

Reconfiguring Clients

The DHCPv6 reconfiguration mechanism allows a server to force the client to contact the server for new configuration information, including addresses, lifetimes for addresses, and other parameters. The server sends a Reconfigure message to the client, to which the client responds with either a Renew or an Information-request message. The server then sends the client a Reply message that contains the new information.

Interaction with IPv6 Auto-Configuration

DHCPv6 address configuration complements IPv6 auto-configuration, and IPv6 nodes use one or both forms of configuration or the other. The IPv6 router advertisement mechanism provides administrative control over that choice through flag bits that indicate whether IPv6 nodes should use auto-configuration or DHCPv6 to determine their IPv6 addresses. The flag bits also control whether the clients use DHCPv6 to obtain other configuration parameters when using auto-configuration for addresses.

Summary

The IETF is developing a new version of IP, called *IPv6*. IPv6 includes several features for address management that either are not included in IPv4 or represent formalization of current practice in IPv4. IPv6 includes scoped addresses, which are forwarded only across a single network (link) or only within a single site. With IPv6, computers can obtain information about network numbering and an auto-configuration mechanism through which they can generate addresses based on the network numbering information.

The DHCWG has developed a specification of DHCP for IPv6, called *DHCPv6*. DHCPv6 explicitly allows for multiple addresses on each interface and allows clients to request addresses in more than one transaction. DHCPv6 allows servers to initiate interactions with clients so that servers can push new configuration information to clients on demand. Network administrators can control whether IPv6 devices use auto-configuration or DHCPv6 through the router advertisement mechanism.

DHCPv6 is published as an Internet Draft, and the details might change before the specification is accepted as an Internet Standard. To obtain information about the latest version of the DHCPv6 specification, refer to the DHCWG charter page, `www.ietf.org/html.charters/dhc-charter.html`.

PART IV
Appendixes

IN THIS PART

A

Microsoft DHCP Server Examples

The main text of this book provides examples that use the configuration file language of the ISC DHCP server. This appendix shows you how to use the Microsoft DHCP server supplied with Windows 2000 for every example in the main text that the Microsoft server supports.

This appendix is intended for use as a reference and should not be read sequentially. As a result, some examples are redundant, and in many cases the text illustrating one example also refers to another example.

To use these examples, you must have the Microsoft DHCP server installed, as described in Chapter 13, "The Microsoft DHCP Server." Before using the examples in this appendix, you should read Chapter 13, which explains how to make the Microsoft DHCP server work.

After you install the Microsoft DHCP server, you must start the DHCP Manager and connect it to the DHCP server you want to modify. Figure A.1 shows the DHCP Manager window.

Examples 3.1, 3.2, 3.3, and 3.4

Subnets are called *scopes* in the Microsoft DHCP server. To add a scope, select the server you want to update (generally `Local Machine`) in the DHCP Manager window. The server is then listed in the DHCP Manager window, as shown in Figure A.1. Select Action, New Scope. The DHCP Manager runs the New Scope Wizard, which prompts you for information about the new scope. At a minimum, you need to configure the addresses available for assignment in the scope and the maximum lease time for the scope.

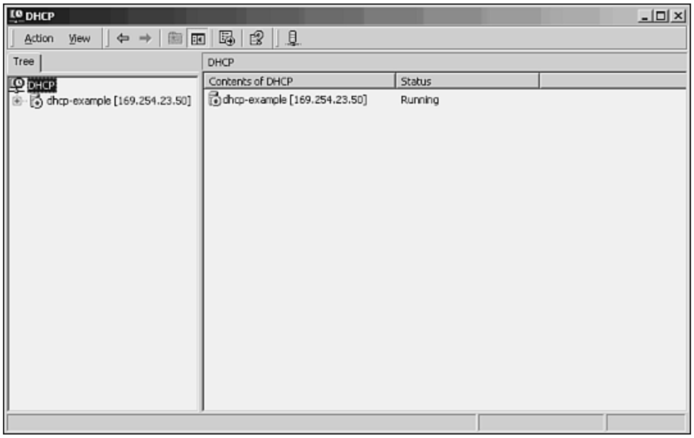

FIGURE A.1 The DHCP Manager window.

Example 3.2, in Chapter 3, "Configuring the DHCP Server," shows examples of five new subnets. Because Microsoft treats scopes as a concept independent of subnets, it is impossible to show an example of an empty scope declaration, as in Example 3.2. Instead, Examples 3.1, 3.2, 3.3, and 3.4 are combined. Figure A.2 shows the full definition of the first of the GSI subnets.

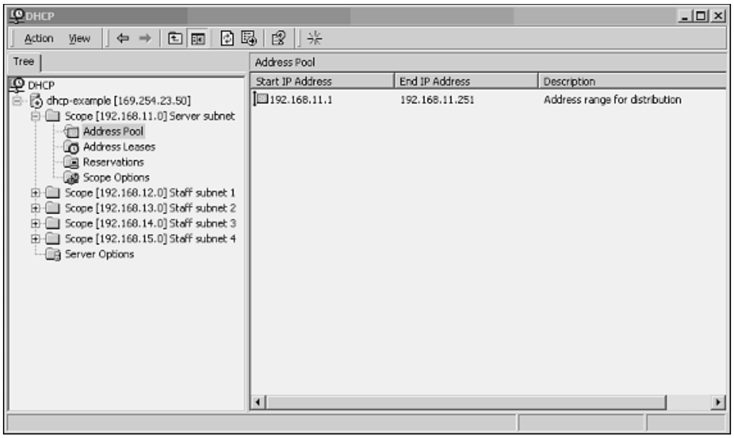

FIGURE A.2 The DHCP Manager window, showing subnets for the GSI network.

Examples 3.5 and 3.6

Example 3.5 shows the format of an option declaration, and Example 3.6 shows a simple subnet-specific option declaration. The Microsoft DHCP server enables you to declare options that are specific to the scope, very much like the `routers` option shown in Example 3.6. To define an option that is local to a scope, highlight the scope in the DHCP Manager window and select the Scope Options item. Select Action, Configure Options and a dialog box for option configuration appears. Figure A.3 shows the `routers` option being defined.

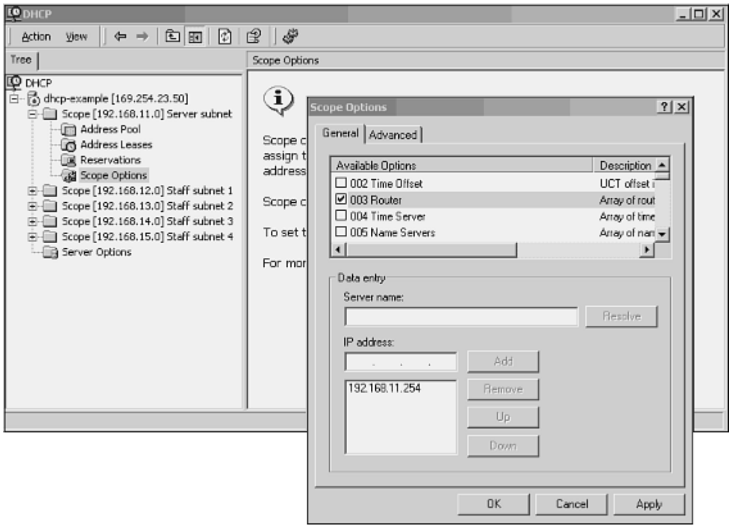

FIGURE A.3 Entering a router's IP address in the Array Editor dialog box.

You might notice that the names of the options are slightly different in the Microsoft DHCP server than in the ISC DHCP server; for example, the ISC `routers` option is equivalent to the Microsoft Router option. The `subnet-mask` option is set up in the Create Scope dialog box, and it can't be configured in the Scope Options dialog box. The ISC `domain-name-servers` option corresponds to the Microsoft `DNS Servers` option, and it is defined in the same way as the `Router` option. Be careful not to mistake the Microsoft `Name Servers` option for the DNS Servers option—the two are completely different. The Name Servers option refers to IEN116 name service, which is now obsolete. If you define IEN116 name servers and don't define domain name servers, your DHCP clients can't do DNS name resolution.

Example 3.7

Example 3.7 configures all five subnets, which involves going through the scope configuration process, described in the preceding eight examples, once for each subnet. This appendix doesn't walk you through that entire process; you should just repeat the instructions given in the preceding examples once for each subnet to fully implement the configuration shown in Example 3.7.

Examples 3.8 and 3.9

The Microsoft DHCP server enables you to set a per-scope maximum lease time, but it does not enable you to set a different default lease time. If a client does not specify a lease time, it receives the maximum lease time.

The maximum lease time is set in the New Scope Wizard or the Scope Properties dialog box. The Microsoft DHCP server requires you to set the maximum lease duration for a scope by specifying the number of days, hours, and minutes, rather than just adding it all up and specifying a number of seconds. The 10,368,000-second `max-lease-time` value in Example 3.9 represents a 120-day lease. The maximum lease time is initially set in the Set Lease window in the New Scope Wizard. To change the maximum lease time for a scope, highlight the scope in the DHCP Manager window and select Action, Properties. Enter the new lease time in the dialog box, as shown in Figure A.4

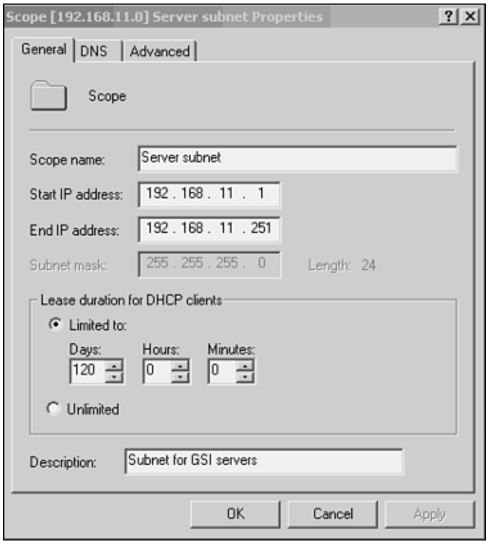

FIGURE A.4 Setting the maximum lease duration in the Scope Properties window.

Example 3.9 extends Example 3.7 to include lease duration information, as shown in Example 3.8.

Example 3.10

Example 3.10 is essentially the same as Example 3.5.

Example 3.11

Example 3.11 shows Example 3.9, with the addition of some global options, as well as global values for the default and maximum lease times. You cannot set a global value for the default and maximum lease times in the Microsoft DHCP server.

To define global options, highlight Server Options in the DHCP Manager window and select Action, Configure Options. The Server Options dialog box appears. Click DNS Domain Name and enter a value for the option, as shown in Figure A.5.

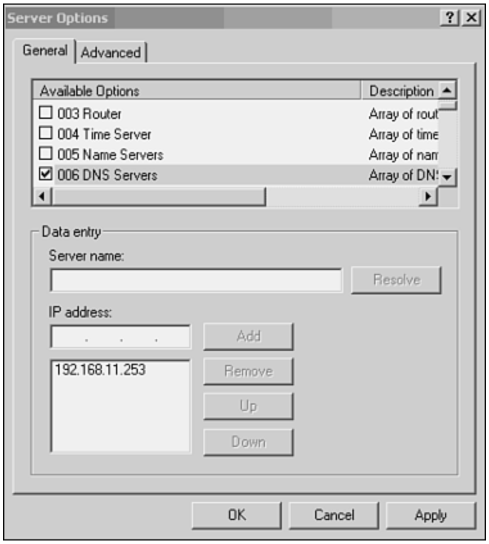

FIGURE A.5 Entering a domain name in the DHCP Options: Global dialog box.

As mentioned in the section "Examples 3.5 and 3.6," earlier in this appendix, the `domain-name-servers` option corresponds to the Microsoft `DNS Servers` option. To set a global value for this option, use the Server Options dialog box described previously. This option takes an array of IP addresses, similar to the `Router` option shown in the section "Examples 3.5 and 3.6," earlier in this appendix.

Examples 12.1 and 12.2

The Microsoft DHCP server doesn't support multiple address allocation pools or access control on pools. Therefore, no Microsoft equivalent exists for these examples.

Example 13.1

Chapter 13 provides examples in both Microsoft and ISC forms. Therefore, an example is not provided in the Microsoft form here.

Examples 14.1 through 14.6

These examples show output of the ISC DHCP server; no equivalent exists for the Microsoft server.

Example 14.7

The Microsoft DHCP server does not provide equivalents for the `default-lease-time`, `min-lease-time`, `allow-bootp`, and `authoritative` parameters. The text in this appendix for Example 3.9 explains how to set a maximum lease time.

Example 14.8

Example 3.11 shows how to configure values for options to be returned to clients.

Example 14.9

To configure BOOTP file information, highlight the server name in the DHCP Manager window and select Action, Properties. Then click the Show the BOOTP Table Folder button.

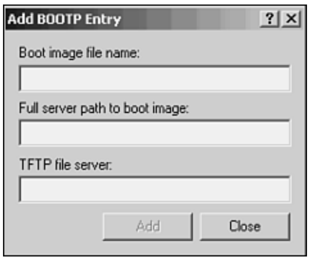

FIGURE A.6 The BOOTP Table dialog box.

Examples 14.10 and 14.11

The section "Defining New Options" in Chapter 13 describes how to define new options in the Microsoft DHCP server.

Example 14.12

No known equivalent exists for the Microsoft DHCP server.

Example 14.13

The section "Examples 3.1, 3.2, 3.3, and 3.4," earlier in this appendix, provides an example of how to define a subnet for the Microsoft DHCP server.

Example 14.14

Although the ISC server has a general notion of scopes, the Microsoft DHCP server does not. Instead, it defines three specific scopes—the global scope, the subnet scope, and the reservation scope. However, Microsoft does not refer to these as scopes; Microsoft uses the term *scope* to refer to an IP subnet. The explanation in the section "Examples 3.5 and 3.6," earlier in this appendix, shows how to define an option in the subnet scope. The section "Example 3.11," earlier in this appendix, shows how to define an option in the global scope.

To define an option that's specific to a reservation, select the scope in which the reservation appears in the DHCP Manager window, and then select Reservations. Select the reservation you want to modify, and then select Action, Configure Options. Then enter any options the same way you do in the Reservation Options dialog box. The sections "Examples 3.5 and 3.6" and "Example 3.11," earlier in this appendix, describe how to do this. Figure A.7 shows the Reservation Options dialog box for a reservation.

Examples 14.15 through 14.17

These examples show Unix-specific Bourne Shell scripts that are used to start the ISC DHCP server and relay agent, so no equivalent exists for the Microsoft server.

Examples 15.1 and 15.2

See the section "Examples 3.1, 3.2, 3.3, and 3.4," earlier in this appendix, for the Microsoft DHCP equivalents.

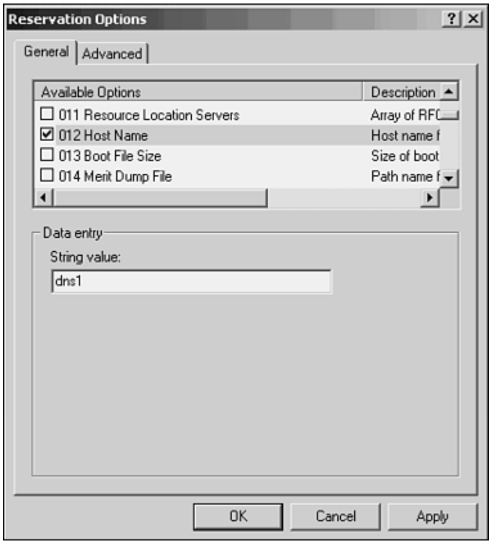

FIGURE A.7 The dialog boxes that are navigated when setting reservation-specific options.

Examples 15.3 through 15.7

See the sections "Examples 3.5 and 3.6" and "Example 3.7," earlier in this appendix, for the Microsoft DHCP equivalents.

Example 15.8

The Microsoft DHCP relay agent is configured by using the Routing and Remote Access Control Panel. To open the DHCP Relay Agent dialog box, first open the Control Panel, and then select IP Routing and then DHCP Relay Agent; then select Action, Properties.

From the DHCP Relay Agent dialog box, you can add a DHCP server by typing in its IP address and clicking the Add button. Figures A.8 and A.9 show these two dialog boxes.

You can also configure the Microsoft DHCP relay agent not to relay DHCP packets until a certain number of seconds has passed since the client started trying to acquire an IP address. You can also limit the number of hops a DHCP packet can make before the relay agent discards it. Every time a client request is relayed through a relay agent, the hop count is incremented by one. When this number reaches the number you choose in the TCP/IP Properties dialog box, the relay agent does not relay the packet. In practice, it is rarely useful to relay DHCP packets more than once, so you should set this number to 1.

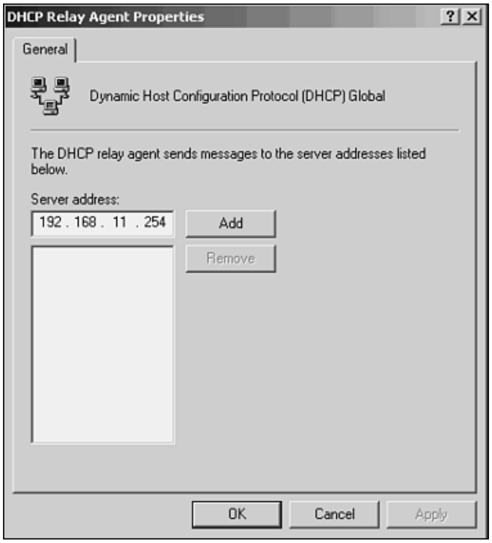

FIGURE A.8 The TCP/IP Properties Agent dialog box.

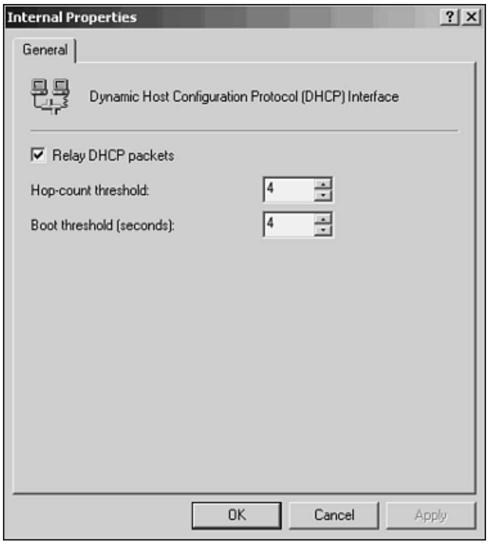

FIGURE A.9 The Windows 2000 DHCP Relay Agent dialog box.

After you have configured the DHCP Relay Agent, right-click the entry in the Interface window. The dialog box that is displayed allows you to set both the number of seconds to wait before relaying a client message and the upper limit on the hop count in a client message. Figure A.9 shows the dialog box for these settings.

Example 15.9

This example is specific to Cisco devices. No Microsoft equivalent exists.

Example 15.10

See the section "Example 3.11," earlier in this appendix, for details about setting up global options. The `ntp-servers` option is equivalent to the Microsoft DHCP server's `NTP Servers` option.

Examples 15.11 and 15.12

See the section "Examples 3.5 and 3.6," earlier in this appendix, for an explanation of configuring options for a scope.

Example 15.13

This example is specific to a Cisco router, so no Microsoft equivalent exists.

Example 15.14

The section "Example 3.11," earlier in this appendix, gives an explanation of configuring global and scope options.

Example 15.15

Section 13.3.2 discusses superscopes and explains how to implement the equivalent of what is shown in Example 15.14.

Example 15.16

The Microsoft DHCP server does not provide a way to define options that apply to a superscope.

Example 15.17

No known equivalent method exists for doing this with the Microsoft DHCP server.

Example 15.18

See the section "Examples 3.5 and 3.6," earlier in this appendix, for an explanation of configuring options.

Example 15.19

No known equivalent exists for the Microsoft DHCP server.

Example 16.1 and 16.2

DHCP client identifiers are entered through the DHCP Manager by selecting reservations in the appropriate scope and clicking Add. You must enter client identifiers in hexadecimal, so it's not easy to enter an ASCII identifier, as in Example 16.2. Figure A.10 shows a new reservation added with the same client identifier shown in Example 16.1. Note the difference in the way identifiers are entered; the ISC DHCP server expects the numbers to be entered as a sequence of byte values separated by colons, whereas the Microsoft server expects a sequence of two-digit hexadecimal values with no separator.

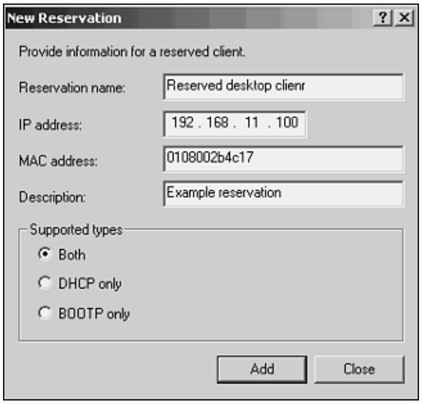

FIGURE A.10 Entering a unique identifier in the `New` Reservation dialog box.

Example 16.3

You cannot enter a client's link-layer address as a separate entity from the client identifier in the Microsoft DHCP server.

Example 16.4

No known equivalent exists for specifying the client identifier sent by the Microsoft DHCP client.

Example 16.5

As mentioned previously, you cannot enter a text client identifier.

NOTE

Examples 16.1 through 16.5 assume that it's possible to talk about client identification separately from static IP address assignment, but this is not possible with the Microsoft DHCP server. That is why the New Reservation dialog box shown in Figure A.10 has a space to enter an IP address.

Example 16.6

This example shows a configuration that includes global options, a subnet declaration with subnet options, and two static IP address assignments. See the section "Examples 3.5 and 3.6," earlier in this appendix, for information on how to define global options.

The explanation in the section "Examples 3.1, 3.2, 3.3, and 3.4," earlier in this appendix, shows how to create a new scope, which is equivalent to a subnet declaration. The Microsoft DHCP server requires that reservations be allocated from the set of addresses that are available within the scope. To define a subnet as shown in Example 16.6, you create the scope by using the New Scope Wizard as described in the section "Examples 3.1, 3.2, 3.3, and 3.4," earlier in this appendix, and include the addresses for the two static assignments in the scope. Then you exclude the IP addresses for the static assignments by entering exclusion ranges. Figure A.11 shows the subnet in Example 16.6 being added, adding the exclusion range for the two static assignments.

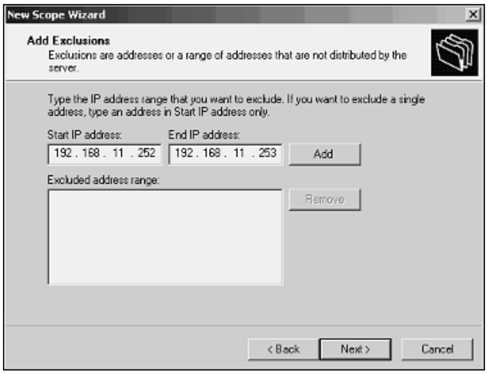

FIGURE A.11 Excluding addresses from the dynamic address pool.

To add the reservations for the two servers, select the new scope in the DHCP Manager window, select Reservations in the scope, and then select `Action`, `New Reservation`. Figure A.12 shows the host `dns-server.genericstartup.com` being added, using the `New Reservation` dialog box. This dialog box enables you to enter more than one client; after you click `Add`, the dialog box is cleared so that you can enter a new static IP address assignment.

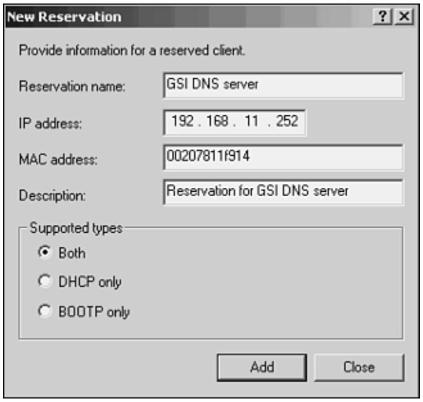

FIGURE A.12 Adding a reservation for the GSI DNS server.

Examples 16.7 and 16.8

To find a client's lease information in the Microsoft DHCP server's lease database, you must know in which scope the DHCP client obtained its IP address. The easiest thing to do is to examine the client's configuration information to find out what IP address it was assigned. Select the scope containing that IP address in the DHCP Manager window, and select Address Leases. The Address Leases window displays all the active leases in that scope, as shown in Figure A.13.

Examples 16.9, 16.10, and 16.11

No known equivalent exists for the Microsoft DHCP server.

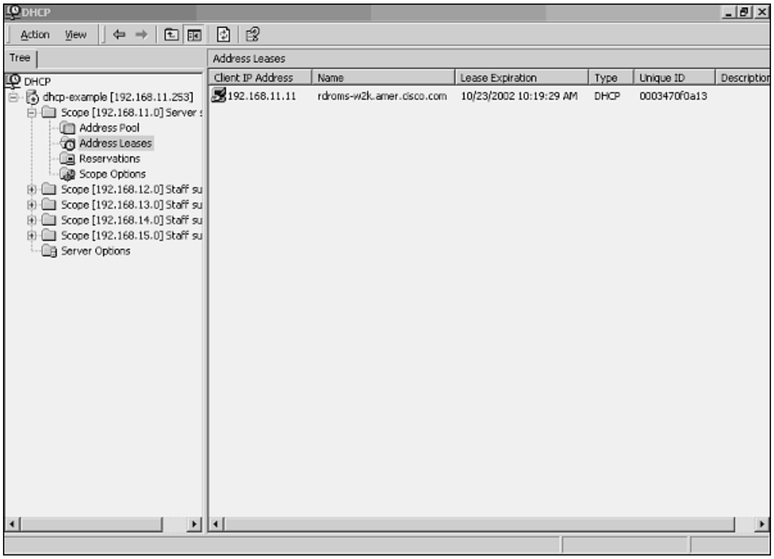

FIGURE A.13 Finding a DHCP client's lease in the Address Leases window.

Example 17.1

No known equivalent exists for the Microsoft DHCP server.

Example 18.1 through 18.9

Because the Microsoft DHCP server does not implement the failover protocol, there is no equivalent for these examples.

Example 19.1

The section "Examples 3.8 and 3.9," earlier in this appendix, describes how to set the maximum lease time. The Microsoft DHCP server does not allow configuration of a minimum or a default lease time.

Examples 20.1 through 20.14

No known equivalents exist for the Microsoft DHCP server.

Example 22.1

No known equivalent exists for the Microsoft DHCP server.

Example 22.2

You can enable the Microsoft DHCP client on some interfaces and not enable it on others. See Chapter 20, "Conditional Behavior," for more information. (However, we know of no way to configure the Microsoft DHCP server not to provide service on some interfaces.)

Example 22.3

See the section in Chapter 19 on the Microsoft DHCP client for more information.

Example 22.4

No known equivalent exists for the Microsoft DHCP server.

Examples 22.5 through 22.7

No known equivalents exist for the Microsoft DHCP client.

Examples in Appendixes

These system-specific examples are not applicable to Windows 2000.

B

ISC DHCP Server Configuration File Reference

This appendix provides a complete reference for the ISC DHCP server's configuration file. Every possible statement or syntax that can appear in the `dhcpd.conf` file is described here. This appendix begins with a description of the organization of the file and then documents all types of declarations and statements. It ends with a complete list of predefined DHCP options and their formats.

The material in this appendix is adapted from material found at `ftp://ftp.isc.org/isc/dhcp/3doc/dhcp-eval.5` and `ftp://ftp.isc.org/isc/dhcp/3doc/dhcpd.conf.5`, copyright 1996–2001, Internet Software Consortium.

How to Use This Appendix

This appendix is organized into groups of statements and declarations of similar types. It is not organized alphabetically. (To look something up alphabetically, refer to the index.) Furthermore, this appendix does not attempt to explain *how* to use the syntax it describes. For that information, refer to the relevant chapters in this book, particularly Chapter 14, "The ISC DHCP Server," Chapter 15, "Configuring a DHCP Server," Chapter 16, "Client Identification and Fixed-Address Allocation," Chapter 17, "Setting Up a Reliable DHCP Service," and Chapter 20 "Customizing the Server's Response to Clients."

Each subsection of this appendix that describes a specific element of syntax from the configuration file also provides a syntax diagram for that element. In a syntax diagram, text that appears in **boldface** is actual syntax that you enter into the file, exactly as it is shown. Text that appears in *italics* is intended to represent some information that you must provide. Such information is explained in the text following the syntax diagram.

Some text in the syntax diagram is specified using neither **boldface** nor *italics*. In such cases, you have a choice as to what syntax to write. If a portion of a syntax diagram is enclosed in square brackets, as shown in Example B.1, you can either include or omit that portion.

Example B.1

```
range low-address [ high-address ];
```

In this example, the *high-address* parameter is optional, and you can enter or omit it, depending on the desired meaning, which is described later in this appendix, in the section "The *range* Declaration." *low-address* and *high-address* are not meant to be typed in literally, but represent some general syntax that is described in the text that follows the syntax diagram.

When a selection of possible keywords can be entered in a particular part of the syntax diagram, these keywords are enclosed in square brackets and separated by vertical bars, as shown in Example B.2. The text following the syntax diagram explains whether you must enter one of these keywords.

Example B.2

```
[ allow | deny ] known clients;
```

In this example, you must enter either the **allow** keyword or the **deny** keyword, but not both.

Finally, in some cases you can specify one or more values in a particular portion of the syntax, denoted by square brackets and ellipses, as in the following example:

```
option name code code = { type-1 [ ... , type-n ] };
```

In this example, you can enter one or more types within the braces, separated by commas.

File Organization

The `dhcpd.conf` file is a sequence of statements and declarations. *Declarations* are evaluated once, when the server reads the file, and they tend to persist as structures in the DHCP server's in-memory database. *Statements* are generally evaluated each time a request is received for a client. The scope in which a statement appears determines whether it is evaluated for a given client.

Many declarations have *scopes* within which other declarations and statements can be written. Most scopes are simply collections of statements and declarations that are enclosed in braces. One exception is the *global scope*, which is the text of the `dhcpd.conf` file that appears outside all pairs of braces. Chapter 14 describes how the DHCP server evaluates scopes.

Statements fall into two major categories:

- Conditional statements
- Statements that define options or parameters

The server handles options in much the same way that it handles parameters. The only difference between them is that parameters control what the server does, and options tell the server what to send to the DHCP client. Chapter 20 describes how conditional statements can be used.

Declarations can't appear within conditional statements because conditional statements are evaluated every time a packet is received, and declarations are evaluated when the configuration is read, and they persist thereafter.

In cases where the syntax for a declaration or statement calls for an IP address, you can specify a domain name instead. This can be very convenient, but there are some things to be aware of when you specify a domain name instead of an IP address:

- Doing a DNS lookup can delay the DHCP server's response to a DHCP client.
- The DHCP server does not notice updates to the domain name server when they happen.
- A DNS lookup can return more than one IP address.

DHCP calls for IP addresses, not domain names, to be sent to clients. This means that whenever you specify a domain name in the configuration file, the DHCP server has to translate that domain name into an IP address. It does this by sending a DNS lookup request to the domain name server and waiting for a response. Each DNS lookup is processed sequentially. If the DHCP server has to do a lot of DNS lookups, this can significantly delay its response to the client.

DNS lookups for statements in the DHCP configuration file are processed when the DHCP server receives a request from a DHCP client. The DHCP server caches the result of any DNS lookup for an hour, so if you only specify a few names in the DHCP configuration file, the DHCP server won't be delayed very much in doing DNS lookups. This means, though, that there may be as much as an hour's delay before the DHCP server notices that a record has changed in the DNS.

If you use domain names in declarations, these are processed once, when the DHCP server starts. If you make a change in the domain name server to a record that is used in a declaration, the DHCP server will not pick up that change until you restart it. Also, if you use a lot of different names in the DHCP configuration file, the DHCP server can take a long time to start. For this reason, if you have a large number of fixed-address declarations (that is, thousands or more), it is better to use an IP address than a domain name in those declarations.

A DNS lookup can return more than one IP address. If you specify a domain name in the configuration file where exactly one IP address is called for, and if the lookup on that domain name returns more than one IP address, the server uses the first IP address that the DNS server returns and silently ignores the others. If you specify a domain name where more than one IP address is allowed, the DHCP server uses all the IP addresses returned by the DNS server. It is also possible for the DNS server to return no addresses for a domain name. In this case, the DHCP server reports an error.

The configuration file has a free-form syntax. Statements and declarations that don't have scopes always end with a `;` character. Declarations that have scopes always end with a `}` character, after the end of the scope. The following example shows a statement, a declaration with no scope, and a declaration with a scope:

```
option dhcp-client-identifier 1:15:2e:17:49:ad:c3;
fixed-address 10.0.1.17;
host "foo" {
    ...
}
```

You can specify comments in the configuration file. A comment begins with a `#` character, and continues to the end of the line. If the `#` character appears inside quotation marks, it does not start a comment. Comments can appear in the middle of statements and declarations. The following examples show how comments can be used:

```
# This is a comment that appears on its own line.
option domain-name-server 10.0.1.1;        # This is a comment.
fixed-address         # This is also a comment.
    10.0.1.17;        # and so is this.
option domain-name "#this is not a comment.";
```

The `shared-network` Declaration

```
shared-network name {
    [ statements ]
    [ declarations ]
}
```

You use the **shared-network** declaration to inform the DHCP server that the subnets declared within it are connected to the same network segment. Parameters specified in the scope of the **shared-network** declaration are considered to be more specific than (and thus override) parameters specified in the global scope.

name should be the name of the network segment that the **shared-network** declaration represents. The server uses this name when printing debugging messages, so it should be a name that meaningfully describes the network segment. The name may have the syntax of a valid domain name (although it is never used as such), or it may be any arbitrary name that you choose.

The `subnet` Declaration

```
subnet subnet-number netmask subnet-mask {
   [ statements ]
   [ declarations ]
}
```

You use the **subnet** declaration to provide the DHCP server with enough information to determine whether a particular IP address is on the subnet. A **subnet** declaration should exist for every subnet from which the DHCP server might receive a DHCP request. If the DHCP server allocates an IP address for a client from a particular subnet, statements in that subnet's scope are executed and are considered more specific than statements in the global scope or in the `shared-network` scope.

subnet-number should be an IP address or domain name that resolves to the subnet number of the subnet being described. *netmask* should be an IP address or domain name that resolves to the subnet mask of the subnet being described. You can use the subnet number, together with the subnet mask, to determine whether a particular IP address is on the specified subnet.

If no `subnet-mask` option is specified in scope for a client receiving an IP address on a particular subnet, the subnet mask from the **host** declaration is supplied to the client in the `subnet-mask` option. If a `subnet-mask` option statement in any scope applies to the client, however, it takes precedence over the subnet mask in the **subnet** declaration.

The `range` Declaration

```
range low-address [ high-address ] ;
```

For any subnet on which addresses are assigned dynamically, there must be at least one **range** declaration. The **range** declaration specifies that the server may allocate to DHCP clients every address from *low-address* to *high-address*. You can specify a single IP address by omitting *high-address*.

All IP addresses in the range should be on the same subnet. If the **range** declaration appears within a **subnet** declaration, all addresses should be on the declared subnet. If the **range** declaration appears within a **shared-network** declaration, all addresses should be on subnets that are already declared within the **shared-network** declaration. This is true even if the range is enclosed in a **pool** declaration and the **pool** declaration is enclosed in the **shared-network** or **subnet** declaration.

The `host` Declaration

```
host name {
   [ statements ]
   [ declarations ]
}
```

The **host** declaration provides information about a particular DHCP client. *name* should be a name for the **host** declaration. Every host declaration must have a unique name, but the only purpose of the name is to identify the host declaration; the name does not, for example, have anything to do with the client's hostname.. **host** declarations match DHCP or BOOTP clients based on either the clients' link-layer address or the `dhcp-client-identifier` option that the client sends. Chapter 16 discusses how this works in detail. BOOTP clients do not normally send a `dhcp-client-identifier` option. Therefore, you must use the link-layer IP address for all clients that might send BOOTP protocol requests.

The **host** declaration has three purposes:

- To assign a static IP address to a client
- To declare a client as "known"
- To specify a scope in which statements can be executed for a specific client

You can make the DHCP server treat some DHCP clients differently than others if **host** declarations exist for those clients. Any request coming from a client that matches a **host** declaration is considered to be from a "known" client. Requests that do not match any **host** declaration are considered to be from "unknown" clients.

You can use this knowledge to control how addresses are allocated, as described in Chapters 16 and 20.

It is possible to write more than one **`host`** declaration for a client. This can be useful if the client has a different scope on different subnets; for each IP address that requires a different scope, one **`host`** declaration should exist. A client can be in the scope of only one **`host`** declaration at a time. **`host`** declarations with static address assignments are in scope for a client only if one of the address assignments is valid for the network segment to which the client is connected. The host scope is the most specific scope. The name of every **`host`** declaration must be unique, even among host declarations that refer to the same DHCP client.

The `hardware` Declaration

`hardware` *`type address`***`;`**

A **`host`** declaration that is matched to a DHCP or BOOTP client using the client's link-layer address should include a **`hardware`** declaration. *`type`* should be the type of network hardware—currently, **`ethernet`**, **`token-ring`**, or **`fddi`**. *`address`* should be the link-layer address—a list of octets specified as hexadecimal numbers, separated by colons.

The `dhcp-client-identifier option` Statement

`option dhcp-client-identifier` *`identifier`*;

Within a **`host`** declaration, you can use a **`dhcp-client-identifier`** option statement to match a **`host`** declaration to a DHCP client. If it is being used to identify the client, the **`dhcp-client-identifier`** option statement must not be specified within any kind of conditional statement. It also must not have a value that is dependent on what the DHCP client sent, because it is executed when the configuration file is parsed, not when a packet is received. If the DHCP client sends a **`dhcp-client-identifier`** option statement with the value specified in *`identifier`*, the **`host`** declaration matches.

> **NOTE**
>
> The DHCP server does not report an error if you specify both a `dhcp-client-identifier` option and a `hardware` declaration. The DHCP server attempts to match the client to the host declaration first, using the `dhcp-client-identifier` option sent by the client and then by using the client's link-layer address.

The `fixed-address` Declaration

```
fixed-address address-1 [ , ..., address-n ];
```

To make a static IP address assignment for a client, the client must match a **host** declaration, as described earlier. In addition, the **host** declaration must contain a **fixed-address** declaration. A **fixed-address** declaration specifies one or more IP addresses or domain names that resolve to IP addresses. If a client matches a **host** declaration and one of the IP addresses specified in the **host** declaration is valid for the network segment to which the client is connected, the client is assigned that IP address.

A static IP address assignment overrides a dynamically assigned IP address that is valid on that network segment. That is, if a new static mapping for a client is added after the client has a dynamic mapping, the client cannot use the dynamic mapping the next time it tries to renew its lease. The DHCP server does not assign an IP address that is not correct for the network segment to which the client is attached and does not override a valid dynamic mapping for one network segment based on a static mapping that is valid on a different network segment.

There should be only one **fixed-address** declaration per **host** declaration.

The `pool` Declaration

```
pool {
   [ permit list ]
   [ range declarations ]
   [ statements ]
}
```

You can use the **pool** declaration to declare an address pool from which IP addresses can be allocated, with its own permit list to control client access and its own scope in which you can declare pool-specific parameters. **pool** declarations can appear within **subnet** declarations or **shared-network** declarations. The addresses in a particular pool are declared with **range** declarations. Addresses must be correct for the **subnet** declaration within which the **pool** declaration is made. If the pool declaration appears within a **shared-network** declaration, the addresses must be on subnets that were previously declared within the same **shared-network** declaration.

You can control address allocation in pools by using the pool permit list—the list of permit statements that appear in a pool declaration. A *permit statement* begins with either the **allow** or **deny** keyword, followed by a statement of what is permitted or not permitted. If a pool has a list of things that are permitted, any client that doesn't match one of the permits cannot be allocated an address from the pool. If a pool has a list of things that are not permitted, any client that doesn't match one of those permits can be allocated an address from the pool.

The `known clients` Permit

[**allow** | **deny**] **known clients**;

The **known clients** permit either allows or prevents allocation from a pool to any client that has a **host** declaration (that is, any client that is known).

The `unknown clients` Permit

[**allow** | **deny**] **unknown clients**;

The **unknown clients** permit either allows or prevents allocation from a pool to any client that has no **host** declaration (that is, any client that is not known).

The `members of` Permit

[**allow** | **deny**] **members of** "*class*";

The **members of** permit either allows or prevents allocation from the pool to any client that is a member of the named class.

The `dynamic bootp clients` Permit

[**allow** | **deny**] **dynamic bootp clients**;

The **dynamic bootp clients** permit either allows or prevents allocation from the pool to any BOOTP client.

The `all clients` Permit

[**allow** | **deny**] **all clients**;

The **all clients** permit either allows or prevents allocation from the pool to all clients. This can be useful when you want to write a **pool** declaration but you do not yet want addresses to be allocated from it. You can also use it to force all clients that were allocated addresses from a pool to obtain new addresses immediately the next time they renew their leases.

The `class` Declaration

```
class "class-name" {
   [ statements ]
   [ declarations ]
}
```

You use **`class`** declarations to group clients together based on information they send. A client becomes a member of a class based on the class's matching rules, or because the client matches a subclass of that class.

`class-name` should be the name of the class and is used in **`members of`** permit statements, as well as in **`subclass`** declarations for subclasses of the named class.

When a packet is received from a client, every **`class`** declaration is examined for a **`match`**, **`match if`**, or **`spawn`** statement, and that statement is checked to see if the client is a member of the class.

`class` scopes are considered to be more specific than **`pool`** or **`subnet`** declaration scopes, but they are less specific than **`host`** declaration scopes. (See the section "Scopes" in Chapter 14 for a description of scopes.)

The `match if` Statement

```
match if boolean-expression;
```

A client is considered a member of a class if, when the server receives a packet from the client, *`boolean-expression`* evaluates to `true`. *`boolean-expression`* may depend on the contents of the packet the client sends.

The `match` Statement

```
match data-expression;
```

If *`data-expression`*, when evaluated using the contents of a client's request, returns a value that matches a subclass of the class in which the **`match`** statement appears, the client is considered a member of both the subclass and the class. If a **`class`** declaration contains both a **`match if`** statement and a **`match`** statement, the **`match if`** statement must match before the **`match`** statement is considered.

The `spawn with` Statement

```
spawn with data-expression;
```

If *`data-expression`* evaluates to a non-null value, the server looks for a subclass of the class in which the **`spawn with`** statement appears that matches the result of the evaluation. If such a subclass exists, the client is considered a member of both the subclass and the class. If no such subclass exists, one is created and recorded in the lease database, and the client is considered a member of the new subclass as well as the class. The matching data for the subclass is the result of evaluating *data-expression*. If a class declaration contains both a **match if** statement and a **spawn with** statement, the **match if** statement must match before the **spawn with** statement is considered.

The lease limit Statement

lease limit *limit*;

The **lease limit** statement causes the DHCP server to limit the number of members of a class that can hold a lease at any one time to *limit*. This limit applies to all addresses the DHCP server allocates, not just addresses on a particular network segment. If a client is a member of more than one class with lease limits, the server assigns the client an address based on either class. If a client is a member of one or more classes with limits and one or more classes without limits, the classes without limits are not considered.

NOTE

At the time of this writing, the lease limit statement interacts badly with the DHCP failover protocol. If you are using failover, you might find that the failover peers give out as many as twice the number of leases that are specified in the lease limit statement. Unfortunately, there is no guarantee that this will be consistent; the number of leases that are allowed in the class can be anywhere from the specified limit to twice the specified limit. It is possible that this problem might be corrected in a future version of the ISC DHCP server, but this would not be a simple bug fix, and at the time of this writing, it is impossible to know whether that will happen.

The subclass Declaration

```
subclass "class-name" class-data;
subclass "class-name" class-data {
   [ statements ]
}
```

The **subclass** declaration declares a subclass of the class named by *class-name*. *class-data* should be either a text string enclosed in quotes or a list of bytes expressed in hexadecimal, separated by colons. Clients match subclasses based on the results of evaluating the **match** or **spawn with** statements in the **class** declaration for *class-name*. If the result of the evaluation matches *class-data,* the client is a member of the subclass and the class. Subclasses can have scopes, in which case their scope is considered to be more specific than the scope of *class-name*; if a subclass has no scope, it inherits the scope of *class-name*. (See the section "Scopes" in Chapter 14 for information about scopes.)

The `group` Declaration

```
group {
   [ parameters ]
   [ declarations ]
}
```

You can use the **group** declaration to provide a common scope for whatever is declared within it. For example, if several **host** declarations require the same set of parameters, you can group them within a **group** declaration and you can specify the statements defining those parameters once in the group scope instead of once in each **host** statement's scope. (See the section "Scopes" in Chapter 14 for information about scopes.)

The `option space` Declaration

```
option space space-name;
```

The **option space** declaration declares a new option space. This declaration must precede all definitions for options in the space being declared. *space-name* should be the name of the option space. Currently five option space names are predefined:

- The **dhcp** option space
- The **agent** option space
- The **server** option space
- The **nwip** option space
- The **fqdn** option space

The **dhcp** option space is the default option space. If an option name is specified without an option space, it is assumed that the name refers to an option in the **dhcp** option space. For example, the option names `dhcp.routers` and `routers` are equivalent.

The **server** option space is the option space for defining server parameters. The following two statements are exactly equivalent:

```
option server.filename "foo";
filename "foo";
```

The **agent** option space is the option space in which relay agent information suboptions appear. The nwip option space is the option space in which NetWare/IP suboptions appear. The FQDN option space is an option space that allows the individual fields of the **client FQDN** option to be examined and modified in the DHCP configuration file.

The include Directive

```
include "filename";
```

The **include** directive inserts the contents of another file in a DHCP configuration file. *filename* is the name of the file to be inserted.

The key Declaration

```
key name {
   algorithm algorithm-name;
   secret secret-data;
} [ ; ]
```

The **key** declaration defines a secret key that the DHCP program can use for authentication. *name* is the name of the key, which should be a syntactically valid domain name (the name doesn't have to appear in the DNS). This name is significant: when the DHCP server or client sends a message authenticated with the key, the recipient uses this name to look up the key that it uses to authenticate the message.

algorithm-name is the name of the algorithm to use. Currently, the ISC DHCP distribution supports only the MD5 HMAC algorithm, which can be specified by using HMAC-MD5.SIG-ALG.REG.INT.

secret is a binary secret, represented in base64, which represents the actual shared secret. The format of the secret is the same as for ISC BIND, and secret keys can be generated by using the utilities included with ISC BIND. This is described in more detail in Chapter 23, "Configuring DHCP–DDNS Interactions."

You can include a single file containing key declarations in both the ISC DHCP and ISC BIND configuration files. Although ISC DHCP and ISC BIND do not use the same configuration file syntax, the only difference between the key syntax is the semicolon after the closing brace in ISC BIND. The ISC DHCP configuration file format allows this extra semicolon for compatibility with BIND, but it does not require it.

The `zone` Declaration

```
zone name {
  primary primary-address;
  [ key key-name; ]
}
```

The **zone** declaration defines a DNS zone that the DHCP program can update. *name* is the name of the zone, which should be a fully qualified domain name. *primary-address* is the IP address to which to send updates for the zone. *key-name* is the name of the **key** directive that defines the key to use to update the zone. You may omit the reference to the key if the DNS server does not require that updates to this zone be authenticated. However, you should use a key even for testing, because in practice it is easier to get the DNS server to accept an authenticated update than a non-authenticated update, despite the additional configuration that is required.

The `failover peer` Declaration

```
failover peer name {
  [ failover statements ]
}
...
pool {
   failover peer name;
   ...
}
```

The **failover peer** declaration can take two forms. In the first form, the declaration defines a DHCP failover protocol peering relationship. The declaration should contain various failover-specific statements, which are described in the following sections. In the second form, the declaration references a previously defined failover declaration.

Any address allocation pool that is to be shared with another DHCP server using the failover protocol must contain a failover peer reference. Address allocation pools that do not contain failover peer references are not shared; if the same pool declaration appears on two DHCP servers without failover, the DHCP servers do not cooperate in sharing the addresses.

When two DHCP servers independently allocate the same IP addresses, it is very likely that IP address conflicts will occur. This problem is a dangerous enough that the ISC DHCP server will refuse to start if you define a failover peer but never reference it; this usually means that the person writing the failover configuration doesn't understand how it must be done.

The DHCP server can support more than one failover relationship. To make this happen, you simply write more than one failover peer declaration and reference the appropriate failover peer declaration for each pool in the pool declaration. This is explained in more detail in Chapter 18, "Failover Configuration."

The `primary` Statement

`primary`;

The **`primary`** statement specifies that in this failover relationship, this DHCP server is the primary server.

The `secondary` Statement

`secondary`;

The **`secondary`** statement specifies that in this failover relationship, this DHCP server is the secondary server.

The `address` Statement

`address` *`ip-address`*;

The **`address`** statement specifies the IP address on which the DHCP server listens for connections from its failover peer and from which the DHCP server attempts to connect to its failover peer. You must specify an **`address`** statement in every failover peer definition.

The `peer address` Statement

`peer address` *`ip-address`*;

The **`peer address`** statement specifies the IP address from which the DHCP server expects to receive connections from its failover peer and to which the DHCP server connects when it attempts to connect to its failover peer. You must specify a **`peer address`** statement in every failover peer definition. The address you specify in the **`address`** statement on each failover peer should be the same address that you specify in the **`peer address`** statement on the other peer.

The `port` Statement

`port` *`tcp-port`*;

The **`port`** statement specifies the TCP port on which the DHCP server listens for connections from its failover peer. You must specify a **`port`** statement in every failover peer definition. The standard port number to use for primary servers is 847; for secondary servers it is 647.

The peer port Statement

peer port *tcp-port*;

The **peer port** statement specifies the TCP port to which the DHCP server connects when it attempts to connect to its failover peer. You must specify a **peer port** statement in every failover peer definition. The TCP port number you specify in the **port** statement on each failover peer should be the same port number that you specify in the **peer port** statement on the other peer.

The max-response-delay Statement

max-response-delay *seconds*;

The **max-response-delay** statement specifies the number of seconds that can elapse with the failover connection quiet before the DHCP server assumes that it has lost contact with its peer. This should normally be no shorter than 60 seconds.

The max-unacked-updates Statement

max-unacked-updates *number*;

The **max-unacked-updates** statement specifies the number of binding updates that the DHCP server can send without receiving any acknowledgements. Each binding update must be acknowledged, so once the specified number of updates has been sent, the receipt of only one acknowledgement means that the DHCP server can send only one more update.

The mclt Statement

mclt *seconds***;**

The **mclt** statement specifies the minimum client lead time (MCLT) for the failover relationship. The MCLT should be long enough so that a lease that is *seconds* long will be long enough to be useful but short enough so that if a server entering the partner-down state has to wait *seconds* to reclaim the peer's leases, this will not be a serious problem. The **mclt** statement can be specified only on the primary peer.

The hba and split Statements

hba *colon-separated-hex-list***;**

split *number*;

The **hba** and **split** statements provide two different ways of specifying the hash bucket assignment (HBA) for load balancing. The **hba** statement allows each bit in the hash bucket to be specified individually, whereas the **split** statement allows the HBA to be specified as a ratio. *number* is the smaller number in the ratio, and 256 is always the larger number, so a split of 128 splits clients evenly between the two failover peers. There is no real reason to ever use a split value other than 128. Every primary failover peer definition must include one of these two statements. Secondary failover peer definitions must not include either statement.

The `load balance max secs` Statement

```
load balance max secs seconds;
```

If the **secs** field of the DHCP packet has a value that is greater than the value specified in **load balance max secs,** that will not cause the DHCP server receiving the packet bypasses load balancing. Most DHCP clients report in the **secs** field of the DHCP packet the number of seconds since they first attempted a transaction. The longer the client has been attempting a transaction without success, the more likely it is that the failover peer that should be responding to the client is unable to respond to do so. A value of 3 seconds is generally recommended.

Programming Statements

The ISC DHCP server provides the ability to conditionally evaluate statements based on the values the client sends, and also to set and unset variables and to log messages to the system logger. Chapter 20 describes how you can do this.

The `if` Statement

```
if boolean-expression {
   [ statements ]
}
```

The most basic conditional statement is the **if** statement. If *boolean-expression* evaluates to `true`, the statements enclosed in braces following *boolean-expression* are executed. If *boolean-expression* evaluates to `false`, those statements are skipped.

The `else` Clause

```
if boolean-expression {
   [ statements ]
} else {
   [ statements ]
}
```

An **if** statement can have an **else** clause. If *boolean-expression* turns out to be false, the statements following the **if** statement are skipped but the statements within braces following the **else** clause are executed.

The elsif Clause

```
if boolean-expression-1 {
   [ statements-1 ]
} elsif boolean-expression-2 {
   [ statements-2 ]
} else {
   [ statements-3 ]
}
```

It is possible to chain a series of conditionals so that only one group of statements out of a set is executed. In the preceding syntax diagram, if *boolean-expression-1* evaluates to true, the collection of statements labeled *statements-1* is executed, and *statements-2* and *statements-3* are skipped, regardless of what the result of evaluating *boolean-expression-2* might have been. *boolean-expression-2* is evaluated only if *boolean-expression-1* evaluates to false. If *boolean-expression-2* evaluates to true, the collection of statements labeled *statements-2* is executed. Otherwise, the statements labeled *statements-3* are executed. An **elsif** clause doesn't have to be followed by an **else** clause, but if there is an **else** clause, it must be the last clause in the chain. It is possible to chain together an arbitrary number of **elsif** clauses.

The switch Statement

```
switch ( expression-1 ) {
   case expression-2:
     [ statements ]
   case expression-3:
     [ statements ]
     break;
   ...
   default:
     [ statements ]
}
```

A **switch** statement allows the user to direct the DHCP program to compute the value of an expression and then compare it against a series of expressions, one in each **case** statement, until it finds one that matches. Execution of the statements within the **switch** statement begins after the matching **case** statement. If no **case** statement matches, then execution begins after the **`default`** statement, if there is one. The syntax is much like that of the C `switch` statement, but the semantics are a bit different. The expression in each **`case`** statement is evaluated at runtime; it does not have to be a constant. Like the C `switch/case` statement, though, a **`break`** statement is required to exit the **`switch`** statement; otherwise, once execution of the `switch` statement has begun, it continues through **`case`** statements and **`default`** statements until the end of the **`switch`** statement is reached.

The `on` Statement

```
on event [ | event [ | event ] ] {
   [ statements ]
}
```

The **`on`** statement allows the user to specify an action that will occur when a certain event happens. This is generally useful in the DHCP server. The **`expiry`**, **`release`**, and **`commit`** events are supported. When a client acquires or renews a lease, the **`commit`** event happens. When a client releases a lease, the **`release`** event happens. When a lease expires on its own, the **`expiry`** event happens. The actions to perform when events occur are stored in the lease database, so they occur even if the server is stopped and restarted between the time when the event is set and the time when it happens. This is most useful for doing DNS updates, although it can also be useful with the **`log`** statement.

The `log` Statement

```
log ( priority, expression );
```

The **`log`** statement causes the DHCP server to log the result of evaluating *`expression`* to the system logger at the priority specified by *`priority`*. The expression must evaluate to a printable ASCII string. The **`log`** statement is intended for debugging and does not make things easy for the user.

The `set` Statement

```
set variable = expression;
```

The **`set`** statement stores the result of evaluating *`expression`* in a variable named *`variable`*. This variable is recorded on the lease that is being processed, if a lease is being processed, and persists across restarts of the DHCP server. If no lease is being processed, the **`set`** statement has no effect.

After a variable has been set on a lease, even the expiration of the lease cannot erase the variable; only the **`unset`** statement can remove it. Currently only integer and string values can be stored in variables. Variables are scoped to the lease, so every lease can have a variable with the same name, and each of these variables will be distinct.

The `unset` Statement

```
unset variable;
```

The **`unset`** statement removes the variable named *`variable`* from the lease that is being processed, if a lease is being processed. If a lease is not being processed, this statement has no meaning.

Expressions

The DHCP server can evaluate expressions while executing statements. The DHCP server's expression evaluator currently supports expressions that return the following types:

- A *Boolean* is a true or false, on or off value.
- An *integer* is a 32-bit quantity that can be treated as signed or unsigned, depending on the context.
- A *string of data* is a collection of zero or more bytes. Any byte value is valid in a data string—the DHCP server maintains a length rather than depending on a `NUL` termination.
- A *DNS value* is a special data structure used exclusively by the `ns-update` operator in performing DNS updates.

Expression evaluation is performed whenever statements that use expressions are evaluated—typically when a request is received from a DHCP client, when an event occurs, and on startup. When a request is received from a client, values in the packet the client sends can be extracted and used to determine what to send back to the client. If you can refer to a value in the client packet for which there is no value, the result is the null value. If you try to get data from a client packet in a context where no client packet has been received, you might see an error message. Null values are treated specially in expression evaluation. The precise way in which null values are handled varies according to the operator. In general, a Boolean expression that returns a null value is considered false. A data expression that returns a null value generally results in the statement using the value not having any effect.

Indeterminate Operators

This section defines all the operators the ISC DHCP server currently supports that return an indeterminate type.

The `( )` Operator

(*`expression`* **)**

The **`( )`** operator returns the result of evaluating *`expression`*. This operator can be useful for clarifying code or for overriding operator precedence.

The *`variable`* Operator

`variable`

If a variable appears in an expression, it returns the contents of that variable in the current lease. If there is no current lease, or if there is no such variable in the current lease, this operator returns the null value.

The `null` Operator

`null`

The **`null`** operator returns the null value.

Boolean Operators

This section defines all the operators the ISC DHCP server currently supports that return Boolean values.

The `static` Operator

`static`

The **`static`** operator evaluates to `true` if an address is being assigned to the DHCP client from a fixed-address declaration.

The = Operator

`data-expression-1` **=** *`data-expression-2`*

The **=** operator compares the result of evaluating *`data-expression-1`* and *`data-expression-2`*. If they are the same, the result is `true`; otherwise, it is `false`. If either *`data-expression-1`* or *`data-expression-2`* evaluates to the null value, the result is also null.

The `and` Operator

`boolean-expression-1` **`and`** *`boolean-expression-2`*

The **`and`** operator evaluates to `true` if *`boolean-expression-1`* and *`boolean-expression-2`* both evaluate to `true`. If either expression evaluates to the null value, the result is null. If either expression evaluates to `false`, the result is false.

The `or` Operator

`boolean-expression-1` **`or`** *`boolean-expression-2`*

The **`or`** operator evaluates to `true` if either *`boolean-expression-1`* or *`boolean-expression-2`* evaluates to `true`. If both expressions evaluate to `false`, or to the null value, the result is false.

The `not` Operator

`not` *`boolean-expression`*

The **`not`** operator evaluates to `true` if *`boolean-expression`* evaluates to `false`, and it evaluates to `false` if *`boolean-expression`* evaluates to `true`. If *`boolean-expression`* evaluates to the null value, the result is also null.

The `exists` Operator

`exists` *`option-name`*

The **`exists`** operator returns `true` if the specified option exists in the DHCP packet that is being examined. Otherwise, it evaluates to `false`.

The `known` Operator

`known`

The **`known`** operator evaluates to `true` if the client whose request is currently being processed is known—that is, if there is a **`host`** declaration for it. Otherwise, it evaluates to `false`.

The `defined` Operator

`defined (` *`variable`* **`)`**

The **`defined`** operator evaluates to `true` if a variable named *`variable`* is defined on the lease that is currently being processed. If no lease is being processed, the result is the null value. If the variable is not defined, the result of the evaluation is `false`.

Data Operators

This section describes all the operators that the ISC DHCP server currently supports that return strings of data.

The `substring` Operator

`substring (`*`data-expr`***`, `***`offset`***`, `***`length`***`)`**

The **`substring`** operator evaluates the data expression and returns the substring of the result of the evaluation that starts *`offset`* bytes from the beginning, continuing for *`length`* bytes. *`offset`* and *`length`* are both numeric expressions. If *`data-expr`*, *`offset`*, or *`length`* evaluates to the null value, the result is also null. If *`offset`* is greater than or equal to the length of the evaluated data, a zero-length data string is returned. If *`length`* is greater than the remaining length of the evaluated data after *`offset`*, a data string containing all data from *`offset`* to the end of the data is returned.

The `suffix` Operator

`suffix (`*`data-expr`***`, `***`length`***`)`**

The **`suffix`** operator evaluates *`data-expr`* and returns the last *`length`* bytes of the result of that evaluation. *`length`* is a numeric expression. If *`data-expr`* or *`length`* evaluates to the null value, the result is also null. If *`length`* evaluates to a number greater than the length of the data, all the data is returned.

The `pick` Operator

`pick (`*`first-data-expr`*** `, `***`data-expr`*** `[ ... , `***`last-data-expr`*** `])`**

The **`pick`** operator evaluates each *`data-expr`* in succession, from left to right, until one of them evaluates to something other than the null value. The result of that evaluation is the result of evaluating the **`pick`** operator. The **`pick`** operator can choose among any number of expressions.

The `binary-to-ascii` Operator

`binary-to-ascii (`*`base`***`, `***`width`***`, `***`separator`***`, `***`expression`***`)`**

The **`binary-to-ascii`** operator evaluates the numeric expressions *`base`* and *`width`* and the data expressions *`separator`* and *`expression`*. If any of these expressions evaluates to the null value, the result is null. Otherwise, the binary data that results from evaluating *`expression`* is broken up into chunks that are *`width`* bits wide. Each chunk is converted to a printable ASCII representation of the number in the base specified by *`base`* and then they are all concatenated together, separated by the result of evaluating *`separator`*, and this becomes the result of the evaluation. *`width`* must be a multiple of 8.

The `reverse` Operator

`reverse (`*`width`*`, `*`expression`*`)`

The **`reverse`** operator evaluates the numeric expression *`width`* and the data expression *`expression`*. If either of these expressions evaluates to the null value, the result is null. Otherwise, the binary data that results from evaluating *`expression`* is broken up into chunks that are *`width`* bits wide. The chunks are then concatenated back together in reverse order, and that is the result of the evaluation. *`width`* must be a multiple of 8.

The `option` Operator

`option` *`option-name`*

The **`option`** operator returns the contents of the specified option in the packet that is being considered. If no such option appears in the packet, or if the expression is evaluated in a situation where no packet exists, the result of the evaluation is the null value. If no option space is specified, the **`dhcp`** option space is used.

The `config-option` Operator

`config-option` *`option-name`*

The **`config-option`** operator returns the contents of the specified option as it has been determined from processing the configuration file. If no such option appears in the configuration, the result of the evaluation is the null value. If no option space is specified, the **`server`** option space is used.

The `hardware` Operator

`hardware`

The **`hardware`** operator returns a data string whose first element is the **`htype`** field of the packet that is being considered and whose subsequent elements are first `hlen` bytes of the `chaddr` field of the packet, as described in Appendix C, "The DHCP Message Format." If no packet exists, or if the **`hlen`** field in the packet is invalid, the result is null.

The `packet` Operator

`packet (`*`offset`*`, `*`length`*`)`

The **`packet`** operator returns the specified portion of the packet that is being examined, or null in contexts in which no packet exists. *`offset`* and *`length`* are applied to the contents of the packet, as with the **`substring`** operator. The beginning of the packet is actually the first byte of payload in the UDP packet, so it is not possible to examine the link-layer, IP, or UDP headers.

The `leased-address` Operator

`leased-address`

The **`leased-address`** operator evaluates to a data string that contains the IP address that is being leased to the client, represented as binary data in network byte order. If no IP address is being leased to the client, the result of the evaluation is the null value.

The `host-decl-name` Operator

`host-decl-name`

The **`host-decl-name`** operator evaluates to a data string that contains the name of the **`host`** declaration that has matched the client. If no **`host`** declaration has matched the client, or if no client request is being processed, the result of the evaluation is the null value.

The *text* Operator

"*text*"

A text string, enclosed in quotes, can be specified as a data expression. It evaluates to an ASCII-encoded data string that contains the text between the quotes.

The *string* Operator

byte-1 [... **:** *byte-n*]

A list of hexadecimal byte values, separated by colons, can be specified as a data expression. A single hexadecimal number, appearing where a data string is expected, is interpreted as a data string that contains a single byte.

The `concat` Operator

`concat (`*`data-expr1`***`,`** *`data-expr2`***`)`**

With the **concat** operator, *`data-expr1`* and *`data-expr2`* are evaluated, and the result of concatenating the results of the two evaluations is returned. If either data expression evaluates to the null value, the result is null.

The `encode-int` Operator

`encode-int (`*`numeric-expr`***`,`** *`width`***`)`**

With the **`encode-int`** operator, *`numeric-expr`* is evaluated and encoded as a data string of the specified *`width`*, in network byte order (with the most significant byte first). If *`numeric-expr`* evaluates to the null value, the result is also the null value.

Numeric Operators

A *numeric operator* is an expression that evaluates to an integer. In general, the precision of numeric expressions is at least 32 bits, but the precision of numeric expressions may be more than 32 bits with some CPU architectures. In addition to the operators described in the following sections, the usual mathematical operations are available: addition (**+**), subtraction (**-**), division (**/**), multiplication (*****), remainder of division (**%**), bitwise AND (**&**), bitwise OR (**|**), and bitwise XOR (**^**).

The `extract-int` Operator

`extract-int (`*`data-expr`***`, `***`width`***`)`**

The **`extract-int`** operator extracts an integer value in network byte order from the result of evaluating *`data-expr`*. *`width`* is the width in bits of the integer to extract. Currently, the only supported widths are 8, 16, and 32. If the evaluation of *`data-expr`* doesn't provide sufficient bits to extract an integer of the specified size, the null value is returned.

The `number` Operator

`number`

If a decimal number appears in an expression, it evaluates to the binary value of that number, up to the maximum precision of numeric expressions—at least 32 bits.

The `lease-time` Operator

`lease-time`

The **`lease-time`** operator evaluates to the duration of the lease being given to the client, in seconds. If no lease is being given to the client, the result is the null value.

The `client-state` Operator

`client-state`

The **`client-state`** operator evaluates to a number that represents the state that the DHCP client is in. This operator is applicable only in the DHCP client. Constants are defined for each of the possible states, which are `BOOTING`, `REBOOT`, `SELECT`, `REQUEST`, `RENEW`, and `REBIND`.

The `ns-update` Operator

`ns-update (`*`dns-expr`***` [ ... , `***`last-dns-expr`***`])`**

The **`ns-update`** operator performs the DNS update operation indicated by its arguments, and it returns the result as an integer. Constants are defined for the possible result values: **`NS_NOERROR`**, **`NS_FORMERR`**, **`NS_NOTAUTH`**, **`NS_NOTIMP`**, **`NS_NOTZONE`**, **`NS_NXDOMAIN`**,

NS_NXRRSET, **NS_REFUSED**, **NS_SERVFAIL**, **NS_YXDOMAIN**, and **NS_YXRRSET**. If any of the expressions passed to **ns-update** evaluates to null, the update is not done and **ns-update** returns the null value.

DNS Operators

A *DNS operator* is an expression that evaluates to a data structure that can be used by the **ns-update** operator to perform a DNS update.

The `add` Operator

add (*class***,** *type***,** *name***,** *value***,** *ttl***)**

The **add** operator instructs **ns-update** to add a resource record in the specified class, of the specified type, and with the specified value to the DNS at the specified name, with the specified time to live (TTL). If any of these evaluates to the null value, the **add** operator evaluates to the null value as well. *class* is normally **C_IN**, but other classes can also be specified. *type* is the resource record (RR) type to add, and is usually either **T_A** or **T_PTR**. *name* is the name at which to add the RR, and it should be a fully qualified domain name. *value* is the value for the record. For **A** records, it is the IP address represented as binary data in network byte order. For other records, it is an ASCII text string. *ttl* is specified as a number of seconds.

The `delete` Operator

delete (*class***,** *type***,** *name* [**,** *value*]**)**

The **delete** operator operator instructs **ns-update** to delete a resource record or RR set. The arguments correspond to the arguments for the **add** operator, except that no TTL should be specified. If *value* is specified, only the RR matching the name and value is deleted. If *value* is not specified, the entire RR set is deleted.

The `exists` Operator

exists (*class***,** *type***,** *name* [**,** *value*]**)**

The **exists** operator operator instructs **ns-update** to place a prerequisite on the DNS update that is being constructed. If just *class*, *type*, and *name* are specified, the prerequisite is that an RR set of the specified type must be attached to the specified name in the DNS. If *value* is also specified, there must be an RR in the DNS on the specified name, and it must have the specified value.

The `not exists` Operator

not exists (*class***,** *type***,** *name* [**,** *value*]**)**

The **not exists** operator instructs **ns-update** to place a prerequisite on the DNS update that is being constructed. If just *class*, *type*, and *name* are specified, the prerequisite is that an RR set of the specified type must not be attached to the specified name in the DNS. If *value* is also specified, there must not be an RR in the DNS that is attached to the specified name with the specified value.

Parameter Statements

Parameter statements control how the DHCP server behaves—how long leases should be, whether to respond to a client's request, and so on. These statements can appear in any scope, as well as within conditional statements within a scope, and they affect the response to any client for which that scope is valid.

The allow, deny, and ignore Statements

The **allow**, **deny** and **ignore** statements are parameters you can use to control the server's behavior. These statements should not be confused with **allow** and **deny** pool permits; although both begin with **allow** or **deny**, they have distinct meanings and different syntax. Whereas permits are evaluated during address allocation, **allow** and **deny** statements are evaluated after the address is allocated, so they cannot affect the address allocation process. The **ignore** statement is exactly equivalent to the **deny** statement, except that it specifies that the DHCP server should not log the fact that it didn't respond.

The bootp Flag

```
[ allow | deny | ignore ] bootp;
```

The **bootp** flag tells the server whether to respond to BOOTP requests. Responses to BOOTP requests are allowed by default. The **allow bootp** and **deny bootp** statements should only be used to specify how the DHCP server responds to BOOTP clients with fixed-address allocations; to control dynamic BOOTP allocation, you should use pool permits.

The booting Flag

```
[ allow | deny | ignore ] booting;
```

The **booting** flag tells the server whether to respond to queries from a particular client. By default, booting is allowed, but if it is disabled for a particular client, the server never responds to that client.

The duplicates Flag

```
[ allow | deny | ignore ] duplicates;
```

The **duplicates** flag tells the server whether to assign a DHCP client with a particular client identifier a lease if a client with the same link-layer address has already obtained a lease by using a different client identifier. Duplicate client identifiers for the same link-layer address are allowed by default.

The `declines` Flag

`[ allow | deny | ignore ] declines;`

The **`declines`** flag tells the server whether to honor `DHCPDECLINE` messages sent by the DHCP client. A malicious or buggy client can easily pollute the DHCP server's lease database by declining every lease in it in quick succession. By default, the DHCP server is configured to honor the `DHCPDECLINE` message.

The `client-updates` Flag

`[ allow | deny ] client-updates;`

The **`client-updates`** flag tells the server what policy to follow when a DHCP client indicates that it intends to update its own forward mapping in the DNS. For more information on how this flag is used, see Chapter 23.

The `default-lease-time` Statement

`default-lease-time` *`time`***`;`**

The **`default-lease-time`** statement specifies the duration of the lease that the DHCP server assigns if the client requesting the lease does not ask for a specific expiration time. *`time`* is the duration of the lease in seconds—by default, 12 hours.

The `max-lease-time` Statement

`max-lease-time` *`time`***`;`**

The **`max-lease-time`** statement specifies the maximum duration of the lease that the DHCP server assigns to a DHCP client. Dynamic BOOTP lease lengths are not limited by this parameter. *`time`* is the maximum duration of a client lease in seconds—by default, 24 hours.

The `min-lease-time` Statement

`min-lease-time` *`time`***`;`**

The **`min-lease-time`** statement specifies the minimum duration of a lease that the DHCP server will assign to a client. *`time`* is the minimum duration of a client lease in seconds—by default, 0. You can use the **`min-lease-time`** parameter to force DHCP clients to take leases that are longer than the lease durations that they request.

The `min-secs` Statement

`min-secs` *`seconds`***;**

The **`min-secs`** statement specifies the minimum number of seconds since a client began trying to acquire a new lease that the DHCP server should wait before responding to the client's request.

The number of seconds is based on what the client reports. The maximum value that the client can report is 255 seconds. Generally, if you set *`seconds`* to 1, the DHCP server does not respond to the client's first request, but it always responds to the client's second request.

You can use this parameter to set up a secondary DHCP server that doesn't offer an address to a client until the primary server is given a chance to do so. If the primary server is down, the client binds to the secondary server; otherwise, clients should always bind to the primary server. Note that this does not, by itself, permit a primary server and a secondary server to share a common IP address allocation pool.

The `dynamic-bootp-lease-cutoff` Statement

`dynamic-bootp-lease-cutoff` *`date`***;**

The **`dynamic-bootp-lease-cutoff`** statement sets the ending time for all leases assigned dynamically to BOOTP clients. Because BOOTP clients do not have a way of renewing leases and do not know their leases can expire, the DHCP server normally assigns unlimited leases to all BOOTP clients. However, it might make sense in some situations to set a cutoff date for all BOOTP leases—for example, the end of a school term or the time at night when a facility is closed and all machines are required to be powered off.

`date` should be the date on which all assigned BOOTP leases end. The date is specified in this format:

```
W YYYY/MM/DD HH:MM:SS
```

`W` is the day of the week expressed as a number from `0` (Sunday) to `6` (Saturday). `YYYY` is the year, including the century. `MM` is the month, expressed as a number from `1` to `12`. `DD` is the day of the month, counting from `1`. `HH` is the hour, from `0` to `23`. `MM` is the minute, and `SS` is the second. The time is always expressed in Coordinated Universal Time (UTC), not in local time.

The `dynamic-bootp-lease-length` Statement

`dynamic-bootp-lease-length` *`length`*;

The **`dynamic-bootp-lease-length`** statement specifies the duration of leases that are dynamically assigned to BOOTP clients. At some sites, it might be possible to assume that a lease is no longer in use if its holder has not used BOOTP or DHCP to get its address within a certain time period. The period is specified in *length* as a number of seconds. If a client restarts using BOOTP during the timeout period, the lease duration is reset to *`length`*, so a BOOTP client that boots frequently enough never loses its lease.

CAUTION

You should use this parameter with extreme caution.

The `get-lease-hostnames` Statement

`get-lease-hostnames` *`flag`*;

The **`get-lease-hostnames`** statement specifies that what the DHCP server should do if it has not been given a host-name option for the client it is configuring. If *`flag`* is **`true`**, the server do a reverse lookup on the IP address assigned to the client and sends the result to the client in the `host-name` option. If *`flag`* is **`false`**, or if no **`get-lease-hostnames`** statement is specified, the server doesn't do the reverse lookup, and sends no `host-name` option to the client.

The `authoritative` Statement

`[` **`not`** `]` **`authoritative;`**

When the DHCP server receives a `DHCPREQUEST` message from a DHCP client, requesting a specific IP address, DHCP requires that the server determine whether the IP address is valid for the network to which the client is attached. If the address is not valid, the DHCP server should respond with a `DHCPNAK` message, forcing the client to acquire a new IP address.

To make this determination for IP addresses on a particular network segment, the DHCP server must have complete configuration information for that network segment. Unfortunately, it is not safe to assume that DHCP servers are configured with complete information. Therefore, the DHCP server normally assumes that it does not have complete information and thus is not sufficiently authoritative to safely send `DHCPNAK` messages as required by DHCP.

This default assumption should not be true for any network segment that is in the same administrative domain as the DHCP server. For such network segments, you should specify the **authoritative** statement, so that the server sends DHCPNAK messages as required by the protocol. If the DHCP server receives requests only from network segments in the your administrative domain, you can specify the **authoritative** statement at the top of the configuration file (in the global scope).

NOTE

Version 2.0 and earlier of the DHCP server makes the opposite assumption: that the DHCP server is configured with all configuration information for all network segments of which it is aware. If this assumption is not valid for your configuration, you must write `not authoritative` statements for all network segments where this assumption is not true (or at the top of the configuration file). Version 1.0 does not support the `authoritative` statement.

The `always-reply-rfc1048` Statement

always-reply-rfc1048 *flag*;

Some BOOTP clients expect RFC 1048-style responses but do not follow RFC 1048 when sending their requests. You can tell that a client has this problem if it does not get the options you have configured for it, and if you see in the server log the message `(non-rfc1048)` printed with each `BOOTREQUEST` that is logged.

If you want to send RFC 1048 options to such a client, set the **always-reply-rfc1048** parameter in that client's **host** declaration to `true`. The DHCP server responds with an `RFC-1048-style vendor` options field. You can set this flag in any scope, and it affects all clients covered by that scope.

The `use-lease-addr-for-default-route` Statement

use-lease-addr-for-default-route *flag*;

If the **use-lease-addr-for-default-route** parameter is `true` in a particular scope, instead of sending the value specified in the `routers` option (or sending no value at all), the DHCP server uses the IP address of the lease being assigned to the client in the `routers` option.

The `server-identifier` Statement

server-identifier *hostname*;

You can use the **server-identifier** statement to define the value that is sent in the `dhcp-server-identifier` option for a given scope. The value specified must be an IP address for the DHCP server, and it must be reachable by all clients served by a particular scope.

Use of the **`server-identifier`** statement is not recommended. You should use it only to force a value to be sent when it is incorrect to send the default value (that is, the first IP address associated with the physical network interface on which the request arrives).

Usually, the **`server-identifier`** statement is needed when a physical interface has more than one IP address and the one being sent by default isn't appropriate for some or all clients served by that interface.

Another common case in which **`server-identifier`** is needed is when you define an IP alias for the purpose of having a consistent IP address for the DHCP server, and you want the clients to use this IP address when contacting the server. If the DHCP server crashes, you can start a hot spare with the same IP alias but with a different primary address, and the spare server will receive `DHCPREQUEST` messages from clients in the `RENEWING` state.

Supplying a value for the `dhcp-server-identifier` option is equivalent to using the **`server-identifier`** statement.

The `vendor-option-space` Statement

`vendor-option-space` *`option-space`***`;`**

The **`vendor-option-space`** statement instructs the server to construct a `vendor-encapsulated-options` option, using all the defined options in the option space. If no `vendor-encapsulated-options` option is defined, the server sends this option to the client, if appropriate.

The `site-option-space` Statement

`site-option-space` *`option-space`***`;`**

The **`site-option-space`** statement determines the option space from which site-local options are taken. Site-local options have codes ranging from 128 to 254. If no **`site-option-space`** is specified, site-specific options are taken from the default option space.

The `always-broadcast` Statement

`always-broadcast` *`flag`***`;`**

The **`always-broadcast`** statement directs the DHCP server to set the broadcast bit in its response to any client in whose scope this directive appears, and if the client is on the local network, to broadcast the response to the client, even if the client did not request a broadcast response. This is not normally useful, but for some buggy clients, it can be helpful.

The `ddns-domainname` Statement

`ddns-domainname` `"`*`name`*`"`**`;`**

The **`ddns-domainname`** statement directs the DHCP server to use the domain name specified in *`name`* as the domain in which a DHCP client's name will be stored when that DHCP client's `A` record is updated.

The `ddns-hostname` Statement

`ddns-hostname` `"`*`name`*`"`**`;`**

The **`ddns-hostname`** statement directs the DHCP server to use the name specified in *`name`* as the hostname that will be used when that DHCP client's `A` record is updated. Normally this value should simply be whatever the client sends and should not be set by the DHCP server administrator.

The `ddns-rev-domainname` Statement

`ddns-rev-domainname` `"`*`name`*`"`**`;`**

The **`ddns-rev-domainname`** statement directs the DHCP server to use the domain name specified in *`name`* as the domain in which a DHCP client's name will be stored when that DHCP client's `PTR` record is updated.

The `lease-file-name` Statement

`lease-file-name` *`name`***`;`**

The **`lease-file-name`** statement directs the DHCP server to use the specified name for its lease file.

The `pid-file-name` Statement

`pid-file-name` *`name`***`;`**

The **`pid-file-name`** statement directs the DHCP server to use the specified name for its process ID (PID) file.

The `ddns-updates` Statement

`ddns-updates` *`flag`***`;`**

The **`ddns-updates`** statement tells the DHCP server whether to perform DNS updates for DHCP clients that match the scope in which it appears. If *flag* is **`true`**, the DHCP server performs DNS updates; if *`flag`* is **`false`**, it does not. This parameter allows for fine-grained control over which clients receive DNS updates. It has no effect unless DNS updates are enabled globally with the **ddns-update-style** statement.

The omapi-port Statement

omapi-port *port*;

The **omapi-port** statement instructs the DHCP server to listen for OMAPI connections on the specified port. OMAPI is an experimental protocol that is new in version 3 of the ISC DHCP server. Limited documentation for OMAPI is available in the ISC DHCP distribution.

The omapi-key Statement

omapi-key *key*;

The **omapi-key** statement instructs the DHCP server to authenticate all OMAPI connections by using the specified key, which must be defined in a **key** declaration. If no OMAPI key is specified, OMAPI connections are not authenticated, which is rather dangerous.

The stash-agent-options Statement

stash-agent-options *flag*;

Unicast DHCPREQUEST messages do not go through a relay agent. Relay agent options are added by the DHCP relay agent, not the DHCP client. This means that relay agent options are present when a lease is first acquired but not when it is renewed. The **stash-agent-options** statement tells the DHCP server whether to keep a copy of the relay agent information option it receives when a DHCP client acquires its lease, for use when the lease is renewed. If *flag* is **true**, relay agent information options are saved; otherwise, they are not. By default, relay information options are saved.

The ddns-ttl Statement

ddns-ttl *ttl*;

The **ddns-ttl** statement tells the DHCP server what TTL value to send when it does a DNS update. *ttl* is specified as a number of seconds.

The update-optimization Statement

update-optimization *flag*;

The **update-optimization** statement tells the DHCP server whether to do DNS update optimization. If update optimization is enabled, the DHCP server only does a

DNS update for a particular lease when it is first assigned, when it is released, and when it expires. This can reduce the DNS update load on the DHCP server substantially, but it opens up the possibility that in some cases the DNS will get out of sync with the DHCP server's recollection of what it has stored in the DNS.

The `ping-check` Statement

`ping-check` *`flag`*;

The **`ping-check`** statement tells the DHCP server whether to send an ICMP echo request to probe an IP address before it offers it to a DHCP client. The `ping` check can protect the DHCP server from allocating a lease to a client for an IP address that is in use, but it delays the DHCP server's response by 1 second, which in some cases is too long.

The `update-static-leases` Statement

`update-static-leases` *`flag`*;

The **`update-static-leases`** statement tells the DHCP server whether to perform DNS updates for DHCP clients whose addresses are statically allocated. This is not normally done because the DHCP server does not keep track of statically assigned leases and therefore would not know when to remove the DHCP client's DNS information from the DNS database. Also, the DHCP server can't remember whether it has updated the DNS information for the client, so it updates it every time the client renews its lease.

The `log-facility` Statement

`log-facility` *`facility`*;

The **`log-facility`** statement tells the DHCP server what syslog facility code to use when logging DHCP server messages to the system logger. Possible codes are **`kern`**, **`user`**, **`mail`**, **`daemon`** (this is the default), **`auth`**, **`syslog`**, **`lpr`**, **`news`**, **`uucp`**, **`cron`**, **`authpriv`**, **`ftp`**, and **`local0`** through **`local7`**. Not all values are defined on all systems. The **`log-facility`** statement doesn't take effect until the DHCP server has parsed the `dhcpd.conf` file, so the initial DHCP server logging always goes to the default log facility.

Statements That Define Values to Send to Clients

Four statements define specific values to send to DHCP clients:

- the `filename` statement
- the `server-name` statement

- the `next-server` statement
- the `option` statement

The first three statements define the values to send in the `filename`, `server-name`, and `siaddr` fields of the BOOTP or DHCP packet. The `option` statement is used to define the options sent back to the client in the `option` portion of the DHCP packet.

The `filename` Statement

```
filename "filename ";
```

You can use the **`filename`** statement to specify the name of the initial boot file to be loaded by a client. *`filename`* should be the name of a file that is recognizable to whatever file transfer protocol the client is expected to use to load the file. Some clients might prefer to receive this information in the `bootfile-name` option.

The `server-name` Statement

```
server-name "name";
```

You can use the **`server-name`** statement to inform the client of the name of the server from which it should load its boot file. *`name`* should be the name that is provided to the client. Some clients prefer to receive the server name in the **`server-name`** option.

The `next-server` Statement

```
next-server server-name;
```

You use the **`next-server`** statement to specify the host address of the server from which the initial boot file (specified in the `filename` statement) is to be loaded. `server-name` should be a numeric IP address or a domain name. If no `next-server` parameter applies to a given client, the DHCP server's IP address is used.

The `option` Statement

```
option option-name option-values;

option option-name = data-expression;
```

The **`option`** statement defines the value that is sent to a client for a particular DHCP option, if the server finds that it is appropriate to send a value for that option. *`option-name`* is the name of the option whose value is specified—either an option space name, followed by a period (`.`), followed by the name of that option or the name of an option in the **`dhcp`** option space.

The **`option`** statement can be in one of two forms. In the first, recommended form, the value or values for an option are specified in the format specified by the option's declaration. The second form allows the entire contents of the option to be specified as the result of a data expression. The second form is not recommended because the configuration file parser can't verify that the values are correct, but it is useful for returning an option to the client whose value is derived from what the client sent.

`option-values` should include values that conform to the declaration of the option. If the **`option`** definition (see the section "The **`option`** Definition") specifies a record value—that is, a series of types enclosed in braces—those values should be separated by spaces. If the **`option`** definition specifies an array of values, each element in the array should be separated by commas. An option whose value is an array of records should be specified with the individual values in the record, separated by spaces, and the records themselves should be separated by commas. Example B.3 shows **`option`** statements that illustrate each of these three possibilities.

Example B.3

```
option policy-filter 10.0.0.0 10.0.255.255;
option routers 10.0.0.1, 10.0.0.2;
option static-routes 10.0.0.0 10.0.0.18, 192.168.0.0 10.0.0.17;
```

The `option` Definition

`option` *`option-name`* **`code`** *`code`* = *`definition`*;

In the **`option`** definition, *`option-name`* should be the name of the option being declared. If the option is not being defined in the **`dhcp`** option space, `option-name` should be the name of the option space in which it is being defined, followed by a period (`.`), followed by the actual name of the option. *`code`* should be the option code—which for current DHCP options should be a number between 0 and 255.

`definition` should be the definition of the structure of the option—what data type or types it is made up of and whether it is an array. The ISC DHCP server currently supports a few simple types, such as integers, Booleans, strings, and IP addresses, and it supports the ability to define arrays of single types or arrays of fixed sequences of types (records).

The simple option types that are supported are described in the following sections.

The `boolean` Type

`option` *`name`* **`code`** *`code`* = **`boolean;`**

An option of type **boolean** is a flag with a value of either on or off (true or false). Here is an example of the use of the **boolean** type:

```
option use-zephyr code 180 = boolean;
option use-zephyr on;
```

The integer Type

option *name* **code** *code* = [**signed** | **unsigned**] **integer** *width***;**

Integers can be either signed or unsigned. If neither is specified, signed is assumed. *width* can be 8, 16, or 32, and refers to the number of bits in the integer. The following two lines show the definition of the sql-connection-max option and its use:

```
option sql-connection-max code 192 = unsigned integer 16;
option sql-connection-max 1536;
```

The ip-address Type

option *name* **code** *code* **= ip-address;**

You can express an option whose structure is an IP address either as a domain name or as a dotted quad. The following is an example of the use of the **ip-address** type:

```
option sql-server-address code 193 = ip-address;
option sql-server-address sql.example.com;
```

The text Type

option *name* **code** *code* **= text;**

An option whose type is **text** encodes an ASCII text string, as in the following example:

```
option sql-default-connection-name code 194 = text;
option sql-default-connection-name "PRODZA";
```

The string Type

option *name* **code** *code* **= string;**

An option whose type is **string** is essentially a collection of bytes and can be specified either as quoted text, like the **text** type, or as a list of hexadecimal byte values separated by colons. Here is an example:

```
option sql-identification-token code 195 = string;
option sql-identification-token 17:23:19:a6:42:ea:99:7c:22;
```

Arrays

option *name* **code** *code* **= array of type;**

Options can contain arrays of any of the types shown in the preceding syntax diagram, except for the text and data string types, which aren't currently supported in arrays. The following is an example of an array definition:

```
option kerberos-servers code 200 = array of ip-address;
option kerberos-servers 10.20.10.1, 10.20.11.1;
```

Records

option *name* **code** *code* **=** { *type-1* [**,** ... *type-n*] **};**

Options can contain data structures that consist of a sequence of data types, which is called a *record type*. Here is an example:

```
option contrived-001 code 201 = { boolean, integer 32, text };
option contrived-001 on 1772 "contrivance";
```

It is also possible to have options that are arrays of records. Here is an example:

```
option new-static-routes code 201 = array of {
    ip-address, ip-address, ip-address, integer 8 };
option new-static-routes
    10.0.0.0 255.255.255.0 net-0-rtr.example.com 1,
    10.0.1.0 255.255.255.0 net-1-rtr.example.com 1,
    10.2.0.0 255.255.224.0 net-2-0-rtr.example.com 3;
```

The Standard DHCP Options

The following are the option definitions for all the standard DHCP options:

```
option subnet-mask code 1 = ip-address;
option time-offset code 2 = signed integer 32;
option routers code 3 = array of ip-address;
option time-servers code 4 = array of ip-address;
option ien116-name-servers code 5 = array of ip-address;
option domain-name-servers code 6 = array of ip-address;
option log-servers code 7 = array of ip-address;
option cookie-servers code 8 = array of ip-address;
option lpr-servers code 9 = array of ip-address;
option impress-servers code 10 = array of ip-address;
```

```
option resource-location-servers code 11 = array of ip-address;
option host-name code 12 = string;
option boot-size code 13 = unsigned integer 16;
option merit-dump code 14 = text;
option domain-name code 15 = text;
option swap-server code 16 = ip-address;
option root-path code 17 = text;
option extensions-path code 18 = text;
option ip-forwarding code 19 = boolean;
option non-local-source-routing code 20 = boolean;
option policy-filter code 21 = array of { ip-address, ip-address };
option max-dgram-reassembly code 22 = unsigned integer 16;
option default-ip-ttl code 23 = unsigned integer 8;
option path-mtu-aging-timeout code 24 = unsigned integer 32;
option path-mtu-plateau-table code 25 = array of unsigned integer 16;
option interface-mtu code 26 = unsigned integer 16;
option all-subnets-local code 27 = boolean;
option broadcast-address code 28 = ip-address;
option perform-mask-discovery code 29 = boolean;
option mask-supplier code 30 = boolean;
option router-discovery code 31 = boolean;
option router-solicitation-address code 32 = ip-address;
option static-routes code 33 = array of { ip-address, ip-address };
option trailer-encapsulation code 34 = boolean;
option arp-cache-timeout code 35 = unsigned integer 32;
option ieee802-3-encapsulation code 36 = boolean;
option default-tcp-ttl code 37 = unsigned integer 8;
option tcp-keepalive-interval code 38 = unsigned integer 32;
option tcp-keepalive-garbage code 39 = boolean;
option nis-domain code 40 = text;
option nis-servers code 41 = array of ip-address;
option ntp-servers code 42 = array of ip-address;
option vendor-encapsulated-options code 43 = string;
option netbios-name-servers code 44 = array of ip-address;
option netbios-dd-server code 45 = array of ip-address;
option netbios-node-type code 46 = unsigned integer 8;
option netbios-scope code 47 = text;
option font-servers code 48 = array of ip-address;
option x-display-manager code 49 = array of ip-address;
option dhcp-requested-address code 50 = ip-address;
option dhcp-lease-time code 51 = unsigned integer 32;
option dhcp-option-overload code 52 = unsigned integer 8;
```

```
option dhcp-message-type code 53 = unsigned integer 8;
option dhcp-server-identifier code 54 = ip-address; the
option dhcp-parameter-request-list code 55 = array of unsigned integer 8;
option dhcp-message code 56 = text;
option dhcp-max-message-size code 57 = unsigned integer 16;
option dhcp-renewal-time code 58 = unsigned integer 32;
option dhcp-rebinding-time code 59 = unsigned integer 32;
option vendor-class-identifier code 60 = string;
option dhcp-client-identifier code 61 = string;
option nwip-domain code 62 = string;
option nisplus-domain code 64 = text;
option nisplus-servers code 65 = array of ip-address;
option tftp-server-name code 66 = text;
option bootfile-name code 67 = text;
option mobile-ip-home-agent code 68 = array of ip-address;
option smtp-server code 69 = array of ip-address;
option pop-server code 70 = array of ip-address;
option nntp-server code 71 = array of ip-address;
option www-server code 72 = array of ip-address;
option finger-server code 73 = array of ip-address;
option irc-server code 74 = array of ip-address;
option streettalk-server code 75 = array of ip-address;
option streettalk-directory-assistance-server code 76 = array of ip-address;
option slp-directory-agent code 78 = array of { boolean, ip-address };
option nds-servers code 85 = array of ip-address;
option nds-tree-name code 86 = string;
option nds-context code 87 = string;
option uap-servers code 98 = text;
option subnet-selection code 118 = ip-address;
option nwip.nsq-broadcast code 5 = boolean;
option nwip.preferred-dss code 6 = array of ip-address;
option nwip.nearest-nwip-server code 7 = array of ip-address;
option nwip.autoretries = unsigned integer 8;
option nwip.autoretry-secs = unsigned integer 8;
option nwip.nwip-1-1 = boolean;
option nwip.primary-dss = ip-address;
option fqdn.no-client-update = boolean;
option fqdn.server-update = boolean;
option fqdn.encoded = boolean;
option fqdn.rcode1 = unsigned integer 8;
option fqdn.rcode2 = unsigned integer 8;
option fqdn.hostname = text;
```

```
option fqdn.domainname = text;
option fqdn.fqdn = text;
option agent.circuit-id code 1 = string;
option agent.remote-id code 2 = string; the
```

C

The DHCP Message Format

Every DHCP message includes a `fixed-format` section and a `variable-format` section. The `fixed-format` section is divided into several fields that carry the client's hardware and IP addresses, the server's IP address, and other control information. The `variable-format` section holds options that carry additional control information about the message and configuration parameters.

The `fixed-format` Section

Figure C.1 illustrates the format of the `fixed-format` section of a DHCP message, and Table C.1 explains the use of each of the fields identified in Figure C.1.

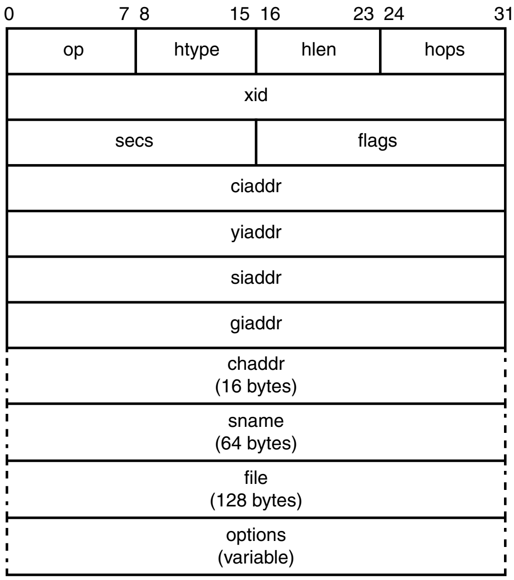

FIGURE C.1 The fields in the `fixed-format` section of a DHCP message.

TABLE C.1 The Fields in the `fixed-format` Section of a DHCP Message

Field	**Length**	**Description**
`op`	1	Message operation code; set to `1` in messages sent by a client, and set to `2` in messages sent by a server. The two possible values for `op` are carried forward from BOOTP for backward compatibility and are sometimes called `BOOTREQUEST` and `BOOTREPLY`, respectively.
`htype`	1	Hardware address type; see Table C.2.
`hlen`	1	Link-layer address length (in bytes); defines the length of the link-layer address in the `chaddr` field.
`hops`	1	Number of relay agents that forwarded this message.
`xid`	4	Transaction identifier; used by clients to match responses from servers with previously transmitted requests.
`secs`	2	Elapsed time (in seconds) since the client began the DHCP process.
`flags`	2	The leftmost bit of this field, called the *broadcast bit,* can be set to `1` to indicate that messages to the client must be broadcast (see the section "Using the Broadcast Flag" in Chapter 7, "Transmitting DHCP Messages," for details).
`ciaddr`	4	Client's IP address; set by the client when the client confirms that its IP address is valid.

TABLE C.1 Continued

Field	Length	Description
yiaddr	4	Client's IP address; set by the server to inform the client of the client's IP address—that is, "your" IP address.
siaddr	4	IP address of the next server for the client to use in the configuration process; for example, the server to contact for TFTP download of an operating system kernel.
giaddr	4	Relay agent ("gateway") IP address; filled in by the relay agent with the address of the interface through which the DHCP message was received.
chaddr	16	Client's link-layer address.
sname	64	DNS name of the next server for the client to use in the configuration process.
file	128	Name of the file for the client to request from the next server; for example, the name of the file that contains the operating system for this client.

The htype Field

Table C.2 lists the possible values for the htype field. RFC 2131 states that the htype field contains the hardware type, as defined in RFC 1700, "Assigned Numbers." This list has been updated since RFC 1700 was issued, and the latest version is available at www.iana.org, the Web site for the Internet Assigned Numbers Authority.

TABLE C.2 Values for the htype Field in the DHCP Message

htype **Value**	**Network Hardware Type**
1	Ethernet (10Mbps–1000Mbps). In practice, this is used for all Ethernet networks, even when IEEE 802.3 encapsulation is used.
2	Experimental Ethernet (3MB).
3	Amateur Radio AX.25.
4	Proteon ProNET Token Ring.
5	Chaos.
6	IEEE 802 networks. In practice, this is used for IBM Token Ring networks.
7	ARCNET.
8	Hyperchannel (FDDI).
9	Lanstar.
10	Autonet Short Address.
11	LocalTalk.
12	LocalNet (IBM PCNet or SYTEK LocalNET).
13	Ultra link.
14	SMDS.

TABLE C.2 Continued

`htype` **Value**	**Network Hardware Type**
15	Frame Relay.
16	ATM.
17	HDLC.
18	Fibre Channel.
19	ATM.
20	Serial Line.
21	ATM.
22	MIL-STD-188-220.
23	Metricom.
24	IEEE 1394.1995.
25	MAPOS.
26	Twinaxial.
27	EUI-64.
28	HIPARP.
29	IP and ARP over ISO 7816-3.
30	ARPSec.
31	IPsec tunnel.
32	InfiniBand (TM).

The `variable-format` Section

Figure C.2 shows the format of each option in the `variable-format` section, and Table C.3 explains the use of each of the fields identified in Figure C.1. Appendix D, "DHCP Options Summary," includes a summary of all the options that can be included in the `variable-format` section.

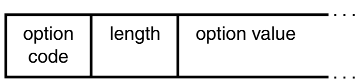

FIGURE C.2 The fields in the `variable-format` section of a DHCP message.

TABLE C.3 The Fields in the `variable-format` Section of a DHCP Message

Field	**Length**	**Description**
`option code`	1 byte	Identifies the specific option; see Appendix D for a list of DHCP options.
`length`	1 byte	Number of bytes in the option value.
`option value`	`length` bytes	Data value for this option.

D

DHCP Options Summary

This appendix includes two tables of DHCP options. Table D.1 lists the options in alphabetical order, and Table D.2 lists the options in numeric order. Table D.1 includes the names that the ISC and Microsoft DHCP servers use to identify DHCP options, along with a short description of each option. The Microsoft DHCP server does not give names to all options; the entries for those options are blank. Unless otherwise indicated, all the DHCP options are defined in RFC 2132.

TABLE D.1 DHCP Options, Listed Alphabetically

Option Name	Code	ISC Option Name	Microsoft Option Name	Description
All subnets are local	27	`all-subnets-local`	`All subnets are local`	`0` specifies that the client should assume that some IP subnets have smaller MTUs; `1` specifies that all subnets have same MTU as subnet to which the client is connected
ARP cache timeout	35	`arp-cache-timeout`	`ARP Cache Timeout`	Timeout (in seconds) for ARP cache entries
Autoconfiguration	116	`autoconfiguration`		`0` specifies that the client should not perform link-local IP address autoconfiguration; `1` specifies that the client should perform auto-configuration[1]
Boot file size	13	`boot-size`	`Boot File Size`	Size of the client bootfile in 512-byte blocks
Boot file name	67	`bootfile-name`	`Boot File Name`	Name of the bootfile to use when the File field is used to carry options
Broadcast address	28	`broadcast-address`	`Broadcast Address`	The broadcast address for the subnet to which the client is attached
Client identifier	61	`dhcp-client-identifier`	`Client's unique identifier`	
Cookie server	8	`cookie-servers`	`Cookie Servers`	Cookie servers[2]
Default IP TTL	23	`default-ip-ttl`	`Default Time-to-Live`	Default TTL client should use for outgoing datagrams
DHCP Authentication option	90	`authentication`		Option used to authenticate DHCP packets transmitted by clients and servers[3]

TABLE D.1 Continued

Option Name	Code	ISC Option Name	Microsoft Option Name	Description
DHCP Message Type	53	`dhcp-message-type`		Identifies the type of the DHCP message; see the section "`DHCP message type`" in Chapter 9 for details
Domain Name Servers	6	`domain-name-servers`	`DNS Servers`	List of DNS server IP addresses
Domain name	15	`domain-name`	`Domain Name`	Default name for DNS name resolution
End	255	`option-end`		Indicates end of options in field
Ethernet encapsulation	36	`ieee802-3-encapsulation`	`Ethernet encapsulation`	`0` specifies Ethernet Version 2 encapsulation[4]; `1` specifies IEEE 802.3[5]
Extensions path	18	`extensions-path`	`Extensions Path`	The name of the file containing additional options to be interpreted according to RFC 2132 format
Finger server	73	`finger-server`	`Finger Servers`	List of finger server[23] IP addresses
Host name	12	`host-name`	`Host Name`	Client hostname
Impress server	10	`impress-servers`	`Impress Servers`	Imagen Impress printer servers
Interface MTU	26	`interface-mtu`	`MTU Option`	Value of MTU that the client should use for this interface
Length of DHCP lease	51	`dhcp-lease-time`		Lease duration (expressed in seconds) for the assigned IP address
IP forwarding	19	`ip-forwarding`	`IP Layer Forwarding`	`0` specifies that datagram forwarding between interfaces is to be disabled; `1` specifies that forwarding is to be enabled
IRC server	74	`irc-server`	`Internet Relay Chat (IRC) Servers`	List of Internet Relay Chat[24] server IP addresses

TABLE D.1 Continued

Option Name	Code	ISC Option Name	Microsoft Option Name	Description
Log server	7	`log-servers`	`Log Servers`	MIT-LCS log servers
LPR server	9	`lpr-servers`	`LPR Servers`	LPDP (Unix `lpr`)[6] servers
Mask supplier	30	`mask-supplier`	`Mask Supplier Option`	`0` specifies that the client should not respond to ICMP subnet mask request messages; `1` specifies that the client should respond
Maximum datagram	22	`max-dgram-reassembly`	`Max DG Reassembly Size`	Maximum size of datagram that reassembly the client should expect to reassemble
Maximum DHCP	57	`dhcp-max-message-size`		Maximum DHCP message size message size accepted by the client
Merit dump file	14	`merit-dump`	`Merit Dump File`	Name of file for memory dump
Message	56	`dhcp-message`		Message from the server to be displayed to the user by the client
Mobile IP home agents	68	`mobile-ip-home-agents`	`Mobile IP Home Agents`	List of mobile IP home agents
Name server	5	`ien116-name-servers`	`Name Servers`	IEN 116 name servers
Name Service Search Order	117	`name-service-search-order`		Ordered list of name service types that the client should use[7]
NDS context	87	`nds-context`		Initial Netware Directory Service (NDS) for client[8]
NDS server	85	`nds-servers`		List of NDS servers[8]
NDS tree name	86	`nds-tree-name`		NDS tree name for the client to use[8]
NetBIOS over TCP/IP gram distribution	45	`netbios-dd-server`	`NetBIOS over TCP/IP NBDD`	List of NetBIOS Datagram data-Distribution (NBDD) servers[9]
NetBIOS over TCP/IP name server	44	`netbios-name-servers`	`WINS/NBNS Servers`	List of NetBIOS Name Server (NBNS) IP addresses[9]

TABLE D.1 Continued

Option Name	Code	ISC Option Name	Microsoft Option Name	Description
NetBIOS over TCP/IP type	46	`netbios-node-type`	`WINS/NBT Node type`	Client's NetBIOS over TCP/IP node node type[9]; see the section "`NetBIOS address over TCP/IP node type`" in Chapter 9 for details
NetBIOS over TCP/IP scope	47	`netbios-scope`	`NetBIOS Scope ID`	Client's NetBIOS over TCP/IP scope[9]
NetWare/IP domain	62	`nwip-domain`		Name of NetWare/IP domain for client[10]
NetWare/IP information	63	`nwip.nsq-broadcast nwip.preferred-dss nwip.nearest-nwip-server nwip.autoretries nwip.autoretry-secs nwip.nwip-1-1 nwip.primary-dss`		NetWare/IP parameters; see the section "NetWare/IP Suboptions " in Chapter 9 for details[10]
NIS domain Name	40	`nis-domain`	`NIS Domain Name`	Client's Network Information Service (NIS) domain
NIS servers	41	`nis-servers`	`NIS Servers`	List of NIS servers
NIS+ domain Name	64	`nisplus-domain`	`NIS+ Domain Name`	Client's Network Information Service+ (NIS+) domain
NIS+ servers	65	`nisplus-servers`	`NIS+ Servers`	List of NIS+ server IP addresses
NNTP server	71	`nntp-server`	`Network News Transport Protocol (NNTP) Servers`	List of NNTP servers

TABLE D.1 Continued

Option Name	Code	ISC Option Name	Microsoft Option Name	Description
Non-local source routing	20	`non-local-source-routing`	`Non-local Source Routing`	`0` specifies that forwarding of datagrams with nonlocal source routes is to be disallowed; `1` specifies that forwarding of such datagrams is to be allowed
NTP servers	42	`ntp-servers`	`NTP Servers`	List of Network Time Protocol (NTP)[11] servers
Option overload	52	`dhcp-option-overload`		Specifies whether File and Sname fields are used to carry options
Pad	0	`pad`		Carries no data
Parameter request list	55	`dhcp-parameter-request-list`		List of options requested by the client
Path MTU aging	24	`path-mtu-aging-seconds`	`Path MTU Aging Timeout`	Timeout (in seconds) for aging PMTU values[12]
Path MTU plateau table	25	`path-mtu-plateau-table`	`Path MTU Plateau Table`	List of MTU sizes for PMTU discovery[12]
Perform mask discovery	29	`perform-mask-discovery`	`Perform Mask Discovery`	`0` specifies that the client should not perform ICMP subnet mask discovery; `1` specifies that the client should perform subnet mask discovery
Perform router discovery	31	`router-discovery`	`Perform Router Discovery`	`0` specifies that the client should not perform router discovery[13]; `1` specifies that the client should perform router discovery
Policy filter	21	`policy-filter`	`Policy Filter Masks`	List of policy filters for nonlocal source routing
POP3 server	70	`pop-server`	`Post Office Protocol`	List of Post Office Protocol 3

TABLE D.1 Continued

Option Name	Code	ISC Option Name	Microsoft Option Name	Description
		(POP3)[25] server IP	(POP3) Servers	addresses
Requested IP address	50	dhcp-requested-address		IP address requested by the client
Relay Agent Information the relay Option	82	relay-agent-information		Additional information supplied by agent[14]
Resource location servers	11	resource-location-servers servers	Resource Location Servers	Resource Location Protocol[15]
Root path	17	root-path	Root Path	Name of the client's root disk
Router solicitation for router	32	router-solicitation-address	Router Solicitation Address	The address the client should use address discovery[13]
Routers	3	routers	Router	IP addresses of default routers to use on the client's network segment
Server identifier	54	dhcp-server-identifier		The IP address of the DHCP server
SLP Directory Agent	78	slp-directory-agent		The Service Location Protocol Directory agent[16]
SLP Service Scope	79	slp-service-scope		The Service Location Protocol service scope[16]
SMTP server	69	smtp-server	Simple Mail Transport Protocol (SMTP) Servers	List of Simple Mail Transport Protocol (SMTP) servers
Static route	33	static-routes	Static Route Option	List of static routes; each static route includes a destination
StreetTalk directory assistance server	76	streettalk-directory-assistance-server	StreetTalk Directory Assistance (STDA) servers	List of STDA servers
StreetTalk server	75	streettalk-server	StreetTalk Servers	List of StreetTalk servers
Subnet mask	1	subnet-mask		The subnet mask for the network segment to which the client is connected
Subnet Selection	118	subnet-selection		The subnet number to use in

TABLE D.1 Continued

Option Name	Code	ISC Option Name	Microsoft Option Name	Description
Option				selecting the client IP address[17]
Swap server	16	`swap-server`	`Swap Server`	The swap server for the client
T1	58	`dhcp-renewal-time`		The time, in seconds, until the client should renew the lease on the assigned address
T2	59	`dhcp-rebinding-time`		The time, in seconds, until the client should rebind the lease on the assigned address
TCP default TTL	37	`default-tcp-ttl`	`Default TTL Option`	The default TTL for TCP segments
TCP keepalive garbage	39	`tcp-keepalive-garbage`	`Keepalive Garbage`	`0` specifies that the client should not send a garbage byte with a TCP keepalive message; `1` specifies that the client should send a garbage byte
TCP keepalive interval	38	`tcp-keepalive-interval`	`Keepalive Interval`	The time, in seconds, to wait before sending a keepalive message on a TCP connection
TFTP server name	66	`tftp-server-name`	`Boot Server Host Name`	The name of the TFTP server to use when the Sname field is used to carry options
Time offset	2	`time-offset`	`Time Offset`	The time offset, in seconds, from Coordinated Universal Time (UTC) for the client's network segment
Time server	4	`time-servers`	`Time Server`	The time server's IP addresses[18]
Trailer	34	`trailer-encapsulation`	`Trailer Encapsulation`	`0` specifies that the client should

TABLE D.1 Continued

Option Name	Code	ISC Option Name	Microsoft Option Name	Description
				not negotiate the use of trailers through ARP[19]; `1` specifies that the client should negotiate the use of trailers
UAP servers	98	`uap-servers`		List of URLs specifying User Authentication Protocol servers[20]
User Class	77	`User Class`		User class information[21]
Vendor class identifier	60	`vendor-class-identifier`		Vendor identification
Vendor specific information	43	`vendor-encapsulated-options`	`Vendor Specific Info`	Vendor-specific information; see the section "`vendor-specific information`" in Chapter 9 for a description of the contents
WWW server	72	`www-server`	`World Wide Web (WWW) Servers`	List of World Wide Web server IP addresses
X Window System Display Managers	49	`x-display-managers`	`X Window System Display`	List of X Window System display managers[22]
X Window System font server	48	`font-servers`	`X Window System Font`	List of X Window System font servers[22]

[1]RFC 2563
[2]RFC 865
[3]RFC 3118
[4]RFC 894
[5]RFC 1042
[6]RFC 1179
[7]RFC 2937
[8]RFC 2241
[9]RFC 1001, RFC 1002
[10]RFC 2242
[11]RFC 1305
[12]RFC 1191
[13]RFC 1256
[14]RFC 3046
[15]RFC 887
[16]RFC 2610
[17]RFC 3011
[18]RFC 868
[19]RFC 893
[20]RFC 2485
[21]RFC 3004
[22]RFC 1198
[23]RFC 1288
[24]RFC 2811, RFC 2812, RFC 2813
[25]RFC 1939

TABLE D.2 DHCP Options, Listed in Numeric Order by Option Code

Option Name	Code	ISC Option Name	Microsoft Option Name
Pad	0	pad	
Subnet mask	1	`subnet-mask`	
Time offset	2	`time-offset`	`Time Offset`
Routers	3	`routers`	`Router`
Time server	4	`time-servers`	`Time Server`
Name server	5	`ien116-name-servers`	`Name Servers`
Domain Name Servers	6	`domain-name-servers`	`DNS Servers`
Log server	7	`log-servers`	`Log Servers`
Cookie server	8	`cookie-servers`	`Cookie Servers`
LPR server	9	`lpr-servers`	`LPR Servers`
Impress server	10	`impress-servers`	`Impress Servers`
Resource location servers	11	`resource-location-servers`	`Resource Location Servers`
Host name	12	`host-name`	`Host Name`
Boot file size	13	`boot-size`	`Boot File Size`
Merit dump file	14	`merit-dump`	`Merit Dump File`
Domain name	15	`domain-name`	`Domain Name`
Swap server	16	`swap-server`	`Swap Server`
Root path	17	`root-path`	`Root Path`
Extensions Path	18	`extensions-path`	`Extensions Path`
IP forwarding	19	`ip-forwarding`	`IP Layer Forwarding`
Non-local source routing	20	`non-local-source-routing`	`Non-local Source Routing`
Policy filter	21	`policy-filter`	`Policy Filter Masks`
Maximum datagram reassembly	22	`max-dgram-reassembly`	`Max DG Reassembly Size`
Default IP TTL	23	`default-ip-ttl`	`Default Time-to-Live`
Path MTU aging	24	`path-mtu-aging-seconds`	`Path MTU Aging Timeout`
Path MTU plateau table	25	`path-mtu-plateau-table`	`Path MTU Plateau Table`
Interface MTU	26	`interface-mtu`	`MTU Option`
All subnets are local	27	`all-subnets-local`	`All Subnets Are Local`
Broadcast address	28	`broadcast-address`	`Broadcast Address`
Perform mask discovery	29	`perform-mask-discovery`	`Perform Mask Discovery`
Mask supplier	30	`mask-supplier`	`Mask Supplier Option`
Perform router discovery	31	`router-discovery`	`Perform Router Discovery`
Router solicitation address	32	`router-solicitation-address`	`Router Solicitation Address`
Static route	33	`static-routes`	`Static Route Option`
Trailer	34	`trailer-encapsulation`	`Trailer Encapsulation`
ARP cache timeout	35	`arp-cache-timeout`	`ARP Cache Timeout`

TABLE D.2 Continued

Option Name	Code	ISC Option Name	Microsoft Option Name
Ethernet encapsulation	36	`ieee802-3-encapsulation`	`Ethernet Encapsulation`
TCP default TTL	37	`default-tcp-ttl`	`Default TTL Option`
TCP keepalive interval	38	`tcp-keepalive-interval`	`Keepalive Interval`
TCP keepalive garbage	39	`tcp-keepalive-garbage`	`Keepalive Garbage`
NIS domain Name	40	`nis-domain`	`NIS Domain Name`
NIS servers	41	`nis-servers`	`NIS Servers`
NTP servers	42	`ntp-servers`	`NTP Servers`
Vendor specific information	43	`vendor-encapsulated-options`	`Vendor Specific Info`
NetBIOS over TCP/IP name server	44	`netbios-name-servers`	`WINS/NBNS Servers`
NetBIOS over TCP/IP datagram distribution	45	`netbios-dd-server`	`NetBIOS over TCP/IP NBDD`
NetBIOS over TCP/IP node type	46	`netbios-node-type`	`WINS/NBT Node type`
NetBIOS over TCP/IP scope	47	`netbios-scope`	`NetBIOS Scope ID`
X Window System font server	48	`font-servers`	`X Window System Font`
X Window System Display Managers	49	`x-display-managers`	`X Window System Display`
Requested IP address	50	`dhcp-requested-address`	
Length of DHCP lease	51	`dhcp-lease-time`	
Option overload	52	`dhcp-option-overload`	
DHCP Message Type	53	`dhcp-message-type`	
Server identifier	54	`dhcp-server-identifier`	
Parameter request list	55	`dhcp-parameter-request-list`	
Message	56	`dhcp-message`	
Maximum DHCP message size	57	`dhcp-max-message-size`	
T1	58	`dhcp-renewal-time`	
T2	59	`dhcp-rebinding-time`	
Vendor class identifier	60	`vendor-class-identifier`	
Client identifier	61	`dhcp-client-identifier`	`Client's Unique Identifier`
NetWare/IP domain	62	`nwip-domain`	

NetWare/IP information	63	`nwip.nsq-broadcast`
		`nwip.preferred-dss`
		`nwip.nearest-nwip-server`
		`nwip.autoretries`
		`nwip.autoretry-secs`
		`nwip.nwip-1-1`
		`nwip.primary-dss`	

TABLE D.2 Continued

Option Name	Code	ISC Option Name	Microsoft Option Name
NIS+ domain Name	64	`nisplus-domain`	`NIS+ Domain Name`
NIS+ servers	65	`nisplus-servers`	`NIS+ Servers`
TFTP server name	66	`tftp-server-name`	`Boot Server Host Name`
Boot file name	67	`bootfile-name`	`Boot File Name`
Mobile IP home agents	68	`mobile-ip-home-agents`	`Mobile IP Home Agents`
SMTP server	69	`smtp-server`	`Simple Mail Transport Protocol (SMTP) Servers`
POP3 server	70	`pop-server`	`Post Office Protocol (POP3) Servers`
NNTP server	71	`nntp-server`	`Network News Transport Protocol (NNTP) Servers`
WWW server	72	`www-server`	`World Wide Web (WWW) Servers`
Finger server	73	`finger-server`	`Finger Servers`
IRC server	74	`irc-server`	`Internet Relay Chat (IRC) Servers`
StreetTalk server	75	`streettalk-server`	`StreetTalk Servers`
StreetTalk directory assistance server	76	`streettalk-directory-assistance-server`	`StreetTalk Directory Assistance (STDA) servers`
User Class	77	`User Class`	
SLP Directory Agent	78	`slp-directory-agent`	
SLP Service Scope	79	`slp-service-scope`	
Relay Agent Information Option	82	`relay-agent-information`	
NDS server	85	`nds-servers`	
NDS tree name	86	`nds-tree-name`	
NDS context	87	`nds-context`	
DHCP Authentication option	90	`authentication`	
UAP servers	98	`uap-servers`	
Autoconfiguration	116	`autoconfiguration`	
Name Service Search Order	117	`name-service-search-order`	
Subnet Selection Option	118	`subnet-selection`	
End	255	`option-end`	

E

Bibliography and Other Resources

Web Resources for DHCP

The following resource is a Web site that accompanies this text:

`www.dhcp-handbook.com`

DHCP users and implementers can also find helpful information at the following site:

`www.dhcp.org`

On the DHCP site you'll find an archive of minutes and other documents from the IETF Dynamic Host Configuration Working Group (DHCWG), as well as links to DHCP RFCs, Internet Drafts, and other DHCP resources.

The DHCWG has a mailing list for discussion of DHCP and IETF issues: `dhcwg@ietf.org`. To join the mailing list, go to the following site:

`www1.ietf.org/mailman/listinfo/dhcwg`

The following is the Internet Software Consortium Web site, which includes information about the ISC DHCP software and related mailing lists:

`www.isc.org`

The IETF Web site includes information about the IETF, a list of IETF working groups, a schedule of upcoming meetings, and proceedings from past meetings:

`www.ietf.org`

The IETF Web site also includes a page for the DHCWG, which maintains links to all current RFCs and Internet Drafts related to DHCP:

`www.ietf.org/html.charters/dhc-charter.html`

The following is the RFC Editor's Web site, where you can find electronic copies of RFCs referenced in this book:

`www.rfc-editor.org`

RFCs Related to DHCP

The following list includes all RFCs that pertain to DHCP specification, including primary specification documents, options accepted as Internet Standards since the DHCP specification was published, and a description of the current process for submitting proposals for new DHCP options:

RFC 1534. Droms, R. "Interoperation Between DHCP and BOOTP." October 1993.

RFC 2131. Droms, R. "Dynamic Host Configuration Protocol." March 1997.

RFC 2132. Alexander, S., and R. Droms. "DHCP Options and BOOTP Vendor Extensions." March 1997.

RFC 2241. Provan, D. "DHCP Options for Novell Directory Services." November 1997.

RFC 2242. Droms, R., and K. Fong. "NetWare/IP Domain Name and Information." November 1997.

RFC 2485. Drach, S. "DHCP Option for the Open Group's User Authentication Protocol." January 1998.

RFC 2563. Troll, R. "DHCP Option to Disable Stateless Auto-Configuration in IPv4 Clients." May 1999.

RFC 2610. Perkins, C., and E. Guttman. "DHCP Options for Service Location Protocol." June 1999.

RFC 2937. Smith, C. "The Name Service Search Option for DHCP." September 2000.

RFC 2939. Droms, R. "Procedures and IANA Guidelines for Definition of New DHCP Options and Message Types." September 2000.

RFC 3004. Stump, G., R. Droms, Y. Gu, R. Vyaghrapuri, A. Demirtjis, B. Beser, and J. Privat. "The User Class Option for DHCP." November 2000.

RFC 3011. Waters, G. “The IPv4 Subnet Selection Option for DHCP.” November 2001.

RFC 3046. Patrick, M. “DHCP Relay Agent Information Option.” January 2001.

RFC 3074. Volz, B., S. Gonczi, T. Lemon, and R. Stevens. “DHC Load Balancing Algorithm.” February 2001.

RFC 3118. Droms, R., and W. Arbaugh. “Authentication for DHCP Messages.” June 2001.

RFC 3203. T’Joens, Y., C. Hublet, and P. De Schrijver. “DHCP Reconfigure Extension.” December 2001.

Other RFCs of Interest

RFC 790. Postel, J. ”Assigned Numbers.” September 1981.

RFC 791. Postel, J. “Internet Protocol.” September 1981.

RFC 792. Postel, J. “Internet Control Message Protocol.” September 1981.

RFC 821. Postel, J. “Simple Mail Transfer Protocol.” August 1982.

RFC 826. Plummer, D. C. “Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48-bit Ethernet Addresses for Transmission on Ethernet Hardware.” November 1982.

RFC 854. Postel, J., and J. K. Reynolds. “Telnet Protocol Specification.” May 1983.

RFC 865. Postel, J. “Quote of the Day Protocol.” May 1983.

RFC 868. Postel, J., and K. Harrenstien. “Time Protocol.” May 1983.

RFC 887. Acetta, M. “Resource Location Protocol.” December 1983.

RFC 893. Leffler, S., and M. Karels. “Trailer Encapsulations.” April 1984.

RFC 894. Hornig, C. “Standard for the Transmission of IP Datagrams over Ethernet Networks.” April 1984.

RFC 903. Finlayson, R., T. Mann, J. C. Mogul, and M. Theimer. “Reverse Address Resolution Protocol.” June 1984.

RFC 950. Mogul, J. C., and J. Postel. “Internet Standard Subnetting Procedure.” August 1985.

RFC 951. Croft, W. J., and J. Gilmore. “Bootstrap Protocol.” September 1985.

RFC 977. Kantor, B., and P. Lapsley. “Network News Transfer Protocol.” February 1986.

RFC 988. Deering, S. “Host Extensions for IP Multicasting.” July 1986.

RFC 1001. NetBIOS Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, End-to-End Services Task Force. “Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods.” March 1987.

RFC 1002. NetBIOS Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, End-to-End Services Task Force. “Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications.” March 1987.

RFC 1035. Mockapetris, P. “Domain Names—Implementation and Specification.” November 1987.

RFC 1042. Postel, J., and J. K. Reynolds. “Standard for the Transmission of IP Datagrams over IEEE 802 Networks.” February 1988.

RFC 1048. Prindeville, P. A. “BOOTP Vendor Information Extensions.” February 1988.

RFC 1122. Braden, R. T. “Requirements for Internet Hosts—Communication Layers.” October 1989.

RFC 1123. Braden, R. T. “Requirements for Internet Hosts—Application and Support.” October 1989.

RFC 1127. Braden, R. T. “Perspective on the Host Requirements RFCs.” October 1989.

RFC 1179. McLaughlin, L. “Line Printer Daemon Protocol”. August 1990.

RFC 1191. Mogul, J. C., and S. E. Deering. “Path MTU Discovery.” November 1990.

RFC 1256. Deering, S. E. “ICMP Router Discovery Messages.” September 1991.

RFC 1288. Zimmerman, D. “The Finger User Information Protocol.” December 1991.

RFC 1305. Mills, David L. “Network Time Protocol (Version 3) Specification, Implementation.” March 1992.

RFC 1321. Rivest, R. “The MD5 Message-Digest Algorithm.” April 1992.

RFC 1459. Oikarinen, J., and D. Reed. “Internet Relay Chat Protocol.” May 1993.

RFC 1497. Reynolds, J. “BOOTP Vendor Information Extensions.” August 1993.

RFC 1519. Fuller, V., T. Li, J. Yu, and K. Varadhan. “Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy.” September 1993.

RFC 1542. Wimer, W. “Clarifications and Extensions for the Bootstrap Protocol.” October 1993.

RFC 1661. Simpson, W. (ed.). "The Point-to-Point Protocol (PPP)." July 1994.

RFC 1700. Reynolds, J., and J. Postel. "Assigned Numbers." October 1994.

RFC 1819. Delgrossi, L., and L. Berger (eds.). "Internet Stream Protocol Version 2 (ST2) Protocol Specification—Version ST2+." August 1995.

RFC 1878. Pummill, T., and B. Manning. "Variable Length Subnet Table for IPv4." December 1995.

RFC 1918. Rekhter, Y., B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. "Address Allocation for Private Internets." February 1996.

RFC 1931. Brownell, D. "Dynamic RARP Extensions for Automatic Network Address Acquisition." April 1996.

RFC 1939. Myers, J., and M. Rose. "Post Office Protocol—Version 3." May 1996.

RFC 1945. Berners-Lee, T., R. Fielding, and H. Frystyk. "Hypertext Transfer Protocol—HTTP/1.0." May 1996.

RFC 2002. Perkins, C. "IP Mobility Support." October 1996.

RFC 2104. Krawczyk, H., M. Bellare, and R. Canetti. "HMAC: Keyed-Hashing for Message Authentication." February 1997.

RFC 2136. Vixie, P. (ed.), S. Thomson, Y. Rekhter, and J. Bound. "Dynamic Updates in the Domain Name System (DNS UPDATE)." April 1997.

RFC 2165. Veizades, J., E. Guttman, C. Perkins, and S. Kaplan. "Service Location Protocol." June 1997.

RFC 2253. Wahl, M., S. Kille, and T. Howes. "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names." December 1997.

RFC 2373. Hinden, R., and S. E. Deering. "IP Version 6 Addressing Architecture." July 1998.

RFC 2464. Crawford, M. "Transmission of IPv6 Packets over Ethernet Networks." December 1998.

RFC 2535. Eastlake, D. "Domain Name System Security Extensions." March 1999.

RFC 2608. Guttman, E., C. Perkins, J. Veizades, and M. Day. "Service Location Protocol, Version 2." June 1999.

Additional Reading

Albitz, P., and C. Liu. *DNS and Bind*. Sebastopol, CA: O'Reilly and Associates, 1998.

Comer, D. *Internetworking with TCP/IP: Principles, Protocols, and Architecture*. Upper Saddle River, NJ: Prentice Hall, 1995.

ISO 9594; ISO 9594-1. “Information Processing Systems—Open Systems Interconnection: The Directory—Overview of Concepts, Models and Services.” International Organization for Standardization and International Electrotechnical Committee. December 1998.

Stevens, W. R. *TCP/IP Illustrated, Volume I: The Protocols*. Reading, MA: Addison-Wesley, 1994.

F

DHCP Server and Operating System Versions

This appendix briefly describes the ISC and Microsoft DHCP servers, with the intent of providing readers with an idea of the different features to consider in any DHCP server. It also describes some operating system dependencies that are specific to the ISC DHCP server (which runs on a variety of operating systems).

Choosing a DHCP Server

The ISC DHCP server and the Microsoft DHCP server are not the only servers available today. Indeed, several other DHCP servers exist that (although not as popular as the ISC and Microsoft servers) are nicely implemented and may be more or less appropriate, depending on your needs. Following is a description of some of the advantages and disadvantages of the ISC and Microsoft servers. You can use this information to help compare other DHCP servers as well.

Operating System Platforms

The first difference you are likely to notice between the ISC and Microsoft DHCP servers is that the latter comes as part of the Windows Server package for Windows NT, Windows 2000, and Windows XP, and it runs only on that operating system. The ISC DHCP server, on the other hand, runs on many Unix and Unix-like systems, but it does not run on any version of Microsoft Windows. However, the ISC DHCP server comes with source code. So, in theory, it can run on almost any system.

Thus, if you run a Windows-only site, you have at least one compelling reason to run the Microsoft DHCP server. If you run a Unix-only site, or a site in which you run central computing services from Unix machines, this may be a good reason to choose the ISC DHCP server or another DHCP server that runs on Unix.

DHCP requires that the server know on what network interface a packet arrives and determine on what interface a packet is sent. On operating systems that cannot provide this capability, if more than one network interface is installed, the server cannot operate correctly.

On some operating systems, the network application programming interface (API) that enables the server to communicate on more than one network interface at a time requires the application to perform link-layer packet framing. The ISC DHCP server supports link-layer framing for FDDI, ethernet, and token ring.

Table F.1 shows a list of operating systems on which the ISC DHCP server runs, as well as network hardware and configurations it supports on each operating system. The Microsoft DHCP server runs only on Microsoft server operating systems, such as Windows NT Server, Windows 2000 Server, and Windows XP Server, and it supports multiple interfaces and all the network hardware types listed in Table F.1.

TABLE F.1 Operating Systems and Network Types Supported by the ISC DHCP Server

Target Operating System	ISC DHCP Server	Multiple Interfaces	Token Ring	FDDI	Ethernet
Windows 2000 Server					
NetBSD	×	×	×	×	×
Linux	×	×	×	×	×
Solaris	×	×		×	×
MacOS X	×	×		×	×
FreeBSD	×	×		×	×
BSD/OS	×	×		×	×
Tru64 Unix	×	×		×	×
AIX	×		×	×	×
HP-UX	×		×	×	×

User Interface

Another difference between the ISC DHCP server and the Microsoft DHCP server is that the ISC DHCP server is configured through a human-readable text file, which you can edit with a text editor, with automated shell scripts, or with both. The Microsoft DHCP server is controlled entirely through a graphical user interface (GUI), which enables you to view and update the server configuration in a convenient hierarchical format but prevents you from modifying the server configuration with any sort of automated script.

This can actually be a key difference between the two servers. For a simple installation, it might be best to configure the DHCP server by using a GUI. For a more complicated installation—or for an installation in which you are trying to do things that weren't envisioned by the people who designed the GUI—being able to directly modify the server configuration with a program can be a big help.

Database Formats

The Microsoft and ISC DHCP servers use different database formats. The Microsoft database is a binary database, which only Microsoft applications can read. The database format is not documented, and Windows does not include a utility to dump the contents of the database into a human-readable form or to load data into the database.

The ISC DHCP server keeps its entire database in memory for quick access, and it stores a log of all transactions on disk. The log is a human-readable text file, and the last entry in the file for an IP address represents the current state of the lease on that address. This database format has some major advantages: When the database is loaded into memory, references and updates to it are quick. Because the on-disk log is stored in a human-readable format, and because the format is documented, it's easy to directly examine the database to determine the state of a particular lease. Writing shell or Perl scripts to automatically modify the database is also very easy.

The disadvantage of the ISC log-structured database is that the DHCP server must parse the entire file when it starts up before it can begin answering requests, whereas the Microsoft DHCP server's binary file enables it to begin responding to requests almost immediately after startup. Depending on the speed of the DHCP server machine's processor, it might take a site with 50,000 active leases several minutes to load the lease database.

Support for BOOTP Clients

Some DHCP servers support only DHCP clients. For example, some versions of the Microsoft DHCP server do not support BOOTP clients. For sites that still have BOOTP clients deployed and want to serve both DHCP and BOOTP clients, using the same server, the ISC DHCP server or a recent version of the Microsoft server are good choices.

Serving both BOOTP and DHCP from the same server makes sense because BOOTP and DHCP use UDP port 67 and share the same relay agent infrastructure. If you want to use different BOOTP and DHCP server software, you must run BOOTP and DHCP on two separate server machines. Also, DHCP/BOOTP relay agents must be configured to send all packets to both DHCP and BOOTP servers because relay agents generally don't know the difference between DHCP and BOOTP packets. In addition to generating an unwanted load on both servers, this configuration might also double the amount of traffic the relay agent generates.

ISC DHCP Server Operating System Dependencies

Because the ISC DHCP server runs on such a wide variety of operating systems, quite a few system dependencies are specific to these operating systems, as discussed in the following sections.

Problems with the 255.255.255.255 Broadcast Address

On almost every operating system in which the ISC DHCP distribution uses the BSD socket API to send and receive packets on the network, a bug exists in the API that prevents it from correctly sending packets to the 255.255.255.255 broadcast address. Instead, the kernel uses the local subnet broadcast address. For example, on a subnet with a subnet mask of 255.255.255.0 and a network number of 10.100.17.0, the kernel substitutes the address 10.100.17.255 if the DHCP server specifies an IP destination address of 255.255.255.255.

This substitution of a different broadcast address is a problem because DHCP specifies that when the DHCP server broadcasts a response to the DHCP client, it must broadcast the response to the 255.255.255.255 broadcast address. Because the client doesn't know what network it is on, it has no way of knowing whether 10.100.17.255 is a host IP address or the broadcast address for the subnet. Many DHCP clients ignore this, but unfortunately the Microsoft DHCP client does not. As a result, the server receives `DHCPDISCOVER` messages from the Microsoft client and broadcasts responses, but the Microsoft client never hears the responses. The Microsoft client continues broadcasting `DHCPDISCOVER` messages until it gives up, and as a result the client can't use the network.

Fortunately, on most such operating systems, you can get around this problem by installing a host route to the 255.255.255.255 address. You can also avoid this problem in other ways, as described in the following sections, under the operating systems on which the solutions were tried. However, if one solution does not work, you might want to try the suggested solution for another operating system instead.

Linux Difficulties

To use the ISC DHCP distribution on Linux, you must use version 3.0 or later. In addition, you must configure raw packet and Linux packet filter support into your Linux kernel. If you do not, you get one of a variety of errors when you first start the DHCP server, indicating that it can't create a socket or bind to it. The error message indicates that you must set `CONFIG_FILTER=y` and `CONFIG_PACKET=y` in your Linux kernel configuration file and rebuild the kernel.

In current versions of the kernel, you must link these capabilities into the kernel; if you configure them as loadable modules, they are not loaded automatically. You can force the modules to load on system startup, but it's more reliable to just link them into the kernel that you start up.

It is very easy to edit the kernel configuration and rebuild the kernel without actually ending up with the new kernel in the boot filesystem. Many users have gone through this process and then concluded that the problem is not in the kernel. Every time someone has reported this conclusion on the ISC DHCP mailing list, it has turned out that they weren't running the correct kernel. So if you continue to get error messages like this, you haven't correctly configured the kernel.

Some older versions of the Linux kernel (2.0.33 through 2.1) provide the capability to use multiple network interfaces with the BSD socket API. This is actually preferable to the Linux packet filter/raw packet API in many ways, but on some kernels, it is necessary to configure a route to the broadcast address for this to work. With Linux, the process for adding a route to the broadcast address varies, depending on the version of the Linux `netutils` package you are using. Example F.1 shows the easiest method, which works on recent versions of `netutils`.

Example F.1

```
% route add -host 255.255.255.255 dev eth0
```

If you are using more than one network interface, you must run this line for each interface. Older versions of `netutils` do not accept an address of 255.255.255.255 on the command line because, as a signed 32-bit number, its value is -1, and that is the same as the error code returned by the `inet_aton()` function if it can't convert the address; therefore, the `route` command assumes the address is bad. To work around this, you can add an entry to your `/etc/hosts` file for the all-ones IP address, as shown in Example F.2.

Example F.2

```
255.255.255.255 all-ones
```

Next, you specify the hostname you used as the destination for the `route` command, as shown in Example F.3.

Example F.3

```
% route add -host all-ones dev eth0
```

On some older versions of `netutils`, even this command doesn't work. You can try the command shown in Example F.4. It might work, but you should upgrade your version of `netutils` if you can.

Example F.4

```
% route add -net 255.255.255.0 dev eth0
```

Some versions of the Linux 2.1 kernel have a configuration parameter that you can turn on and off to indicate whether the kernel should enable processes to act as BOOTP agents. If this feature is not enabled, the DHCP server cannot run. If you have trouble getting the server to work on a Linux 2.1 kernel, you might need to enable this, as shown in Example F.5.

Example F.5

```
% echo 1 >/proc/sys/net/ipv4/ip_boot_agent
```

Versions of the Linux kernel prior to 2.0.33 do not support DHCP service on more than one network interface. Even if you do not have more than one network interface, you should upgrade your Linux kernel before running DHCP; later kernels are known to work better with DHCP service in general.

HP-UX Difficulties

HP-UX has the broadcast address bug described earlier and can support only a single network interface. On some versions of HP-UX, it might be possible to use the `route` commands suggested in the section on Linux. If these don't work, another method that is known to work on some versions of HP-UX is to modify the `/etc/rc.config.d/netconf` file, as shown in Example F.6. (You must modify this to suit your configuration; see the HP documentation for more information.)

Example F.6

```
INTERFACE_NAME[0]=lan0
IP_ADDRESS[0]=1.1.1.1
SUBNET_MASK[0]=255.255.255.0
BROADCAST_ADDRESS[0]="255.255.255.255"
LANCONFIG_ARGS[0]="ether"
DHCP_ENABLE[0]=0
```

Solaris Difficulties

Multiple-interface support on Solaris seems to be stable as of version 3.0. This support uses the DLPI (Data Link Provider Interface) API, a low-level packet-driver API. However, some problems were reported with using the DLPI API with some network interface cards. If the DHCP server starts up with no errors but never seems to receive any packets, you might have this problem. If you have only one network interface, you can modify the `includes/site.h` file by adding `#define USE_SOCKETS`, and then type `make clean; make`. The DHCP server that this builds uses the sockets API instead of DLPI. Except for the restriction that the sockets API can be used with only one network interface on Solaris, no known problems exist on Solaris when using this API.

G
Glossary

Symbols and Numbers

/n Indicates a subnet mask or *prefix length* of *n* bits. *See also* subnet mask.

6bone An experimental network testbed for IPv6.

A

address classes The definitions of the network number and host identifier component of IP addresses. Class A, B, and C addresses define unicast addresses, Class D addresses are used for multicast, and Class E addresses are reserved for future use.

address pool A set of IP addresses that are available for assignment by a DHCP server to clients on a specific network segment.

application layer The component of the TCP/IP protocol models that includes specific protocols used by application programs.

ARP (Address Resolution Protocol) A protocol used by TCP/IP hosts to resolve an IP address into a link-layer address.

ARP cache timeout A setting that defines the lifetime for entries in the ARP cache.

authentication In DHCP, the process of reliably identifying a DHCP client or server to other DHCP participants and of guaranteeing the integrity of DHCP messages.

automatic allocation The assignment of an IP address with an infinite lease to a client by a DHCP server. *See also* dynamic allocation, static allocation.

B

bootfile A file that contains additional information needed by a computer using TCP/IP. For example, a *bootfile* can contain the operating system code for a computer that has no local disk storage.

BOOTP (Bootstrap Protocol) A protocol that provides configuration information to network devices through the network and, thus, eliminates the need for the system administrator to manually configure each network device. BOOTP is a predecessor of DHCP.

bridge A device that connects two network media so that they appear to the attached computers as a single network segment.

broadcast A technique for delivering a network message to more than one destination. Many network technologies include the capability to broadcast a frame to all the devices that are attached to a network segment.

broadcast flag A bit in the `flags` field of a DHCP message that controls the use of broadcast when a server or a relay agent is sending messages to a DHCP client.

C

cable modem network A networking technology that uses the coaxial cable infrastructure used by cable TV to provide network connections to residences at speeds up to several megabits per second.

Class A, B, and C addresses The three classes of IP addresses used for unicast messages. See Chapter 4, "Configuring TCP/IP Stacks," for the formats of these addresses.

client An application, a computer, or another device that uses services provided by other applications or computers, usually through a network.

client identifier A value chosen by a client to be used as its identifier by DHCP servers. This value is used instead of the client's link-layer address.

client/server model A paradigm for organizing distributed systems in which applications contact a server to perform application-specific functions.

collision In DHCP, the use of the same client identifier by different DHCP clients.

colon-separated hexadecimal (colon hex) A textual representation of IPv6 IP addresses in which groups of 16 bits are written in hexadecimal notation, separated by colons. For example, 3FFE:C15:C001:F000:204:DDFF:FEBB:6642 is an IPv6 IP address represented in colon hex notation.

D

data link layer The component of the TCP/IP model that is responsible for delivering an IP datagram across a network segment to the next hop on the path to the datagram's destination.

datagram An IP protocol message.

default router (default route) A router used as the next hop to the destination when there is no specific route in the IP routing table; some implementations specify default routers as *default route* entries in the routing table.

dentist's office Used as a shorthand or example of a small office with a local network (origin unknown).

DHCP (Dynamic Host Configuration Protocol) A protocol that automates the process of configuring network hosts by allowing hosts to obtain IP addresses and configuration parameters through the network. DHCP eliminates the need for manual configuration of hosts and manual assignment of IP addresses by network administrators.

DHCP client A computer or another device that uses DHCP to obtain configuration parameters.

DHCP client port UDP port 68, which is reserved for transmission of messages to DHCP and BOOTP clients.

DHCP option A specific configuration parameter that is carried in the `variable-format` section of a DHCP message.

DHCP server An application program or a computer that provides configuration parameters to DHCP clients.

DHCP server port UDP port 67, which is reserved for DHCP and BOOTP transmission of messages to DHCP servers.

`DHCPACK` message A message that is transmitted by a DHCP server to a client to confirm the use of the parameters requested by the client in a `DHCPREQUEST` message.

`DHCPDECLINE` message A message that is transmitted by a DHCP client to a server to decline an offered address because the client has determined that it is already in use by another client.

`DHCPDISCOVER` message A message that is transmitted by a DHCP client to find servers that are willing to offer an address and configuration parameters.

`dhcpd.conf` The configuration file that controls the operation of the ISC DHCP server.

`dhcpd.leases` The file in which the ISC DHCP server records information about leases on addresses that are assigned to DHCP clients.

`DHCPFORCERENEW` message A message that is transmitted by a DHCP server to a client to cause the client to send a `DHCPREQUEST` message to the server. A `DHCP-FORCERENEW` message is used to cause the client to request new parameters before its current lease expires.

`DHCPINFORM` message A message that is used by a DHCP client that doesn't need an IP address to obtain other configuration parameters.

`DHCPNAK` message A message that is transmitted by a DHCP server to a client to inform the client that it cannot use the parameters requested by the client in a `DHCPREQUEST` message.

`DHCPOFFER` message A message that is transmitted by a DHCP server in response to `DHCPDISCOVER` requests to offer a client an address and configuration parameters.

`DHCPRELEASE` message A message that is transmitted by a DHCP client to a server to explicitly terminate the lease on the client's IP address prior to the expiration of the lease.

`DHCPREQUEST` message A message that is used by a DHCP client to request initial configuration parameters, to confirm the validity of an address already assigned to the client, and to obtain an extension on the lease for an address assigned to the client.

DHCPv6 A version of DHCP for IPv6.

DHCWG (Dynamic Host Configuration Working Group) The working group of the IETF that is responsible for the development of DHCP.

DNS (Domain Name System) A service for translating names for Internet hosts into IP addresses.

dotted decimal A notation for IP addresses in which each byte of the address is written as a decimal number, separated by periods ("dots"). For example, 192.168.0.1 is an IP address represented in dotted-decimal notation.

dynamic allocation Assignment of an IP address to a DHCP client for a finite period of time; the duration of the assignment is known as a *lease*. *See also* automatic allocation, static allocation.

E

encapsulation The technique of carrying data within a protocol message; for example, an IP datagram is encapsulated as the data area in an Ethernet frame.

Ethernet A network technology that carries data across a shared medium.

Ethernet encapsulation The specific format for transmitting an IP datagram in an Ethernet frame. DHCP can use either Ethernet version 2 (RFC 894) or IEEE 802.3 (RFC 1042) encapsulation.

EUI-64 identifier A 64-bit unique identifier defined by IEEE. Assignment of EUI-64 identifiers is managed by the IEEE Registration Authority.

expression In the `/etc/dhcp.conf` file, a mathematical statement whose value is derived from constants in the expression and by values extracted from the DHCP packet, as specified in the expression. See Appendix B, "ISC DHCP Server Configuration File Reference," for examples of the use of expressions.

F

failover protocol A protocol used to provide synchronization between DHCP servers that allows each server to provide service to DHCP clients if the other server fails or becomes unreachable.

finite state machine A technique for describing the behavior of a system as a set of states, transitions between states, and outputs.

firewall A device that is situated between an organization's network and the Internet that filters IP traffic to limit forwarding of unwanted IP datagrams.

fixed-format section The part of a DHCP message that has the same format in every message. The fixed-format section is divided into several fields.

frame A packet that is transmitted across network hardware.

FTP (File Transfer Protocol) A service that includes an application-layer protocol for transferring files between computers.

H

hashed message authentication code A technique for generating a message authentication code (MAC) based on a keyed message digest generator. *See also* MD5, MAC.

hexadecimal A base-16 representation for integers.

host Any networked device that does not act like a router (that is, does not forward IP datagrams).

host requirements documents (RFC 1121, RFC 1122, and RFC 1127) Documents that provide summary and analysis of all the protocols in the TCP/IP suite.

host table A file that contains a list of hosts and their link-layer addresses and DNS names.

hoteling The temporary daily use of offices by staff members.

HTML (Hypertext Markup Language) A system of tags that is used to define the appearance of a document.

HTTP (Hypertext Transfer Protocol) A protocol that is used to transmit documents, usually specified in HTML (for example, between World Wide Web servers and browsers).

I

ICMP (Internet Control Message Protocol) A protocol in the TCP/IP suite that provides error reporting and information functions.

IESG (Internet Engineering Steering Group) An organization that guides the operation of the IETF and sets the status of Internet protocols.

IETF (Internet Engineering Task Force) An organization that is involved in the evolution of the Internet and development and coordination of new Internet protocols.

interface identifier A 64-bit value that can be generated from an interface link-layer address. Interface identifiers are used in IPv6 addresses (see RFC 2343).

internet A set of network segments connected by routers over which hosts use TCP/IP.

Internet The global collection of public and private networks that use TCP/IP.

Internet Draft A preliminary protocol specification published by the IETF for review and comment by the Internet community.

internet layer The component of the TCP/IP model that is responsible for end-to-end delivery of protocol messages between computers.

IP (Internet Protocol) The protocol in the TCP/IP suite that implements the internet layer.

IP address A 32-bit number that is assigned to a network interface that uniquely identifies the computer and specifies the network segment to which the computer is connected.

IP datagram The basic message unit for data delivered by IP.

IPNAT (IP network address translation) *See* NAT.

IPng (IP next generation) *See* IPv6.

IPv6 (IP version 6) A new version of IP that is under development by the IETF.

IRC (Internet Relay Chat) A service for real-time messaging among groups of network users.

ISC (Internet Software Consortium) An organization that produces reference implementations of TCP/IP protocols and services, including DHCP and DNS.

ISC DHCP client A freely available implementation of a DHCP client produced by the ISC.

ISC DHCP relay agent A freely available implementation of a DHCP relay agent produced by the ISC.

ISC DHCP server A freely available implementation of a DHCP server produced by the ISC.

L

LDAP (Lightweight Directory Access Protocol) A protocol that is used to access directory information from directories based on the X.500 models.

lease The period of time over which a DHCP client can use its assigned IP address.

limited broadcast IP address The IP address used for broadcast by hosts on a network segment. The limited broadcast IP address is usually 255.255.255.255, but some legacy devices use 0.0.0.0.

link-layer address The unique identifier for a network interface on a network segment. A frame usually includes the link-layer addresses of both the destination and source network interfaces.

Linux An operating system that is modeled after Unix, for which source code is freely available.

M

MAC (message authentication code) A value that is computed from the contents of an authenticated message that cannot be forged. The recipient uses the MAC to confirm that the contents of the message were not changed.

MAC (Media Access Control) address *See* link-layer address.

magic cookie or cookie A well-known number. In DHCP, it is a 32-bit number that identifies the format of the options in the variable-format section of a DHCP message.

MD5 (Message Digest Algorithm 5) An algorithm for generating a message digest that is based on a secret key.

MIB (management information base) A data structure that consists of named objects shared by SNMP agents and managers. A MIB gives agents and managers a common naming scheme for values to be managed through SNMP.

mobile device A networked computer or another device that can be moved among different network segments.

MTU (maximum transmission unit) The largest frame that can be accepted by a network segment.

multicast A technique for delivering a datagram to multiple destinations by transmitting a single copy of the datagram across any network segment.

N

NAT (network address translation) A technique for rewriting the IP addresses in IP datagrams. It is often used to convert internal, private addresses (for example, from network 10.0.0.0) to external addresses that can be routed through the Internet. A *NAT box* or *NAT* is a device attached to an internal network that provides NAT service.

NDS (NetWare Directory Service) A hierarchical naming or directory service for NetWare.

NetBIOS (Network Basic Input/Output System) A protocol suite originally developed by IBM and now widely used among computers using the Windows operating systems.

NetWare A protocol suite and set of network services developed by Novell.

NetWare/IP A version of the NetWare protocol suite that uses IP for datagram transport.

network analyzer A device that receives all frames from the network segment to which it is attached and displays the contents of the frames in a variety of formats; sometimes called a *packet sniffer* after the name of a widely used commercial network analyzer.

network segment A distinct physical network. All computers that are connected to a network segment can transmit directly to each other.

NFS (Network File System) An open-standard protocol originally developed by Sun Microsystems for the exchange of file contents through a network.

NIC (network interface card) A device that connects a computer or another device to a network.

NIS (Network Information Service), NIS+ (Network Information Service+) A directory service included with the Sun Microsystems Solaris operating system.

NNTP (Network News Transport Protocol) An application protocol for transmission of data, used by the Network News (`netnews`) service.

NTP (Network Time Protocol) An application protocol for coordinating internal clocks among networked computers.

O

octet An 8-bit data item. Usually synonymous with *byte*.

option A configuration parameter that is carried in the `variable-format` section of a DHCP message. Each option includes an *option code* that identifies the option, a *length*, and the *option data*.

options or variable-format section A section of a DCHP message that carries DHCP options, whose length is variable and whose contents and format depend on the option type.

OSPF (Open Shortest Path First) A routing protocol that is based on a link-state algorithm.

overlay or shared segment *See* shared network.

P

packet A generic name for a protocol message. *Packet* is sometimes used as a synonym for a link-layer frame, an IP datagram, or a TCP segment.

physical layer The component of the TCP/IP protocol model that delivers data encoded in a physical representation, such as electrical current, radio waves, or light.

ping An application that uses ICMP Echo Request/Echo Reply messages to determine whether two computers can exchange network messages.

PMTU (path maximum transfer unit) The largest IP datagram that can be transmitted without fragmentation along the path between two computers. In IPv4, PMTU is used to control the size of TCP segments to avoid fragmentation.

POP (Post Office Protocol) A protocol that is used to access the contents of a mailbox on a mail server. For example, a desktop computer can use POP to access mail messages on an organization's mail server.

PPP (Point to Point Protocol) A protocol that is used for transmission of IP datagrams across point-to-point networks such as telephone modem connections.

prefix length The number of bits in the network identifier portion (prefix) of a IP address. The notation */n* indicates a prefix of *n* bits.

proxy ARP A use of ARP in which a device answers an ARP request for an IP address that is not assigned to it.

R

RARP (Reverse Address Resolution Protocol)/DRARP (Dynamic RARP) Internet protocols. A TCP/IP host uses RARP to find its IP address from a static database based on its link-layer address; DRARP allows a new host to be registered in the address database automatically.

RAS (remote access server) A device that provides network access to remote devices connected through, for example, a dialup connection or a tunnel.

relay agent A device that forwards DHCP messages from clients to servers, allowing centrally located DHCP servers to provide DHCP service for clients on several network segments.

renumbering The process of assigning new IP addresses to all the hosts in a network.

reservation An entry in a DHCP server database that manually assigns a specific address to a DHCP client. *See* static allocation.

reserved addresses IP addresses that are reserved for DHCP clients.

RFC (Request for Comments) A document in the series of documents published by the RFC Editor that pertains to the Internet and Internet protocols. The RFCs include protocol specifications, status reports on Internet protocols, and other reports about Internet-related topics.

RIP (Routing Information Protocol) A routing protocol that is based on a vector-distance algorithm.

router A network device that has more than one network interface and forwards IP datagrams between network segments.

router discovery A mechanism (part of ICMP) through which TCP/IP hosts can find routers connected to the same network segment.

routing table A table that contains the addresses of routers to which IP datagrams are to be forwarded for delivery.

RR (resource record) An entry in a DNS server that holds DNS data.

S

scope In the Microsoft DHCP server, a collection of IP addresses and associated parameter values to be assigned to DHCP clients. More generally, scope is the range of influence of an option declaration or variable.

shared network A network segment that has been assigned two or more IP subnets.

shell script A program that is written in a Unix command interpreter language such as `sh` or `csh`.

SLP (Service Location Protocol) A protocol for identifying and locating network services. SLP can provide flexible and dynamic configuration of services also provided through DHCP options.

SMTP (Simple Mail Transport Protocol) A protocol that is used to deliver e-mail messages.

source routing A technique in which the source specifies the path for an IP datagram to its destination, rather than making routing decisions in routers.

state transition diagram A graphical representation of the states and transitions that describe the behavior of a finite state machine.

static route A fixed routing table entry that is not obtained through a routing protocol.

static (or fixed) allocation Preconfigured assignment of IP address and configuration parameters to a DHCP client, usually in the DHCP server configuration file. *See also* automatic allocation, dynamic allocation.

StreetTalk A directory service that is included with Banyan Vines.

subnet A set of IP addresses that share a common network number, defined either by the class of the IP address or by a subnet mask.

subnet declaration In the ISC DHCP server configuration file, the definition of an IP subnet for which the server should manage address assignments.

subnet mask A 32-bit number that identifies which bits of the address make up the subnet address and which are used as the host address.

subnetting A technique for dividing Class A, B, or C addresses into smaller groups of IP addresses or *subnets* that more closely match the addressing requirements for network segments (see RFC 950, RFC 1878, and RFC 1519).

superscope In the Microsoft DHCP server, a network segment that has been assigned more than one IP subnet.

switch A device that provides a direct network connection between pairs of devices connected to its ports. The switch appears to be a dedicated network segment to each of the connected devices.

T

T1 The time at which a DHCP client begins to attempt to extend the lease on its IP address from the DHCP server that originally assigned the address.

T2 The time at which a DHCP client begins to attempt to extend the lease on its assigned IP address from any available DHCP server.

TCP (Transmission Control Protocol) A protocol that provides reliable, connection-oriented delivery of arbitrarily long messages or streams of data.

TCP keepalive A technique for maintaining a TCP connection by periodically transmitting bytes that are not part of the data in the TCP connection.

TCP/IP (Transmission Control Protocol/Internet Protocol) The suite of computer communication protocols used in the Internet; the name comes from the two most important protocols in the suite, TCP and IP. *See* TCP, IP.

TCP/IP stack An implementation of the TCP/IP suite.

TFTP (Trivial File Transfer Protocol) A simple application protocol that is built on UDP for copying files between networked computers.

trailers Headers of TCP/IP messages at the end of each message rather than at the front (see RFC 893).

transaction ID A 32-bit identification number that is used to match DHCP response messages with DHCP request messages.

TTL (time to live) A field in an IP datagram that limits the lifetime of the datagram in an internet. The TTL is initially set to a positive integer and decremented by routers; if the TTL goes to zero, the datagram is discarded.

U

UCS (Universal Character Set) A standard that defines encodings for the characters from most of the writing systems in existence today.

UDP (User Datagram Protocol A protocol that provides best-effort, connectionless delivery of discrete messages. UDP is the protocol that is used to carry DHCP messages.

unicast Delivery of a protocol message to a single destination computer. *Unicast* describes the normal delivery of IP datagrams.

URL (uniform resource locator) A syntactic notation for identifying network objects that includes an access protocol as well as the location and name of the object.

UTC (Universal Time Coordinated; formerly Greenwich Mean Time) The time of day at the Prime Meridian, which does not follow any seasonal adjustments to local time.

UTF-8 (UCS Transformation Format) An encoding that represents 7-bit ASCII values in a single byte and then uses multibyte values to represent other characters from the Unicode Standard.

V

valid lifetime In IPv6, a parameter associated with an IPv6 address or prefix that defines the time through which the address or prefix may be used.

variable-format section The portion of a DHCP message in which options are carried.

virtual hosting A system that provides for multiple copies of a service—for example, multiple World Wide Web servers—through a single physical interface by assigning more than one IP address to the interface.

W

WINS (Windows Internet Naming System) A protocol that provides dynamic NetBIOS to IP address name registration and resolution.

X

X.500 A directory system that was originally part of the OSI protocol suite and is now widely used with TCP/IP.

X Window System A windowing system that can be used for remote graphical displays.

Symbols

A

D

E

F

G

H

I - J - K

L

M

O

P

Q - R

S

T

U

V

W - Z