INTRODUCTION TO IP ADDRESS MANAGEMENT

IEEE Press
445 Hoes Lane
Piscataway, NJ 08854

Books in the IEEE Press Series on Network Management

Telecommunications Network Management Into the 21st Century,
Co-Editors Thomas Plevyak and Salah Aidarous, 1994

Telecommunications Network Management: Technologies and Implementations,
Co-Editors Thomas Plevyak and Salah Aidarous, 1997

Fundamentals of Telecommunications Network Management,
by Lakshmi Raman, 1999

Security for Telecommunications Management Network,
by Moshe Rozenblit, 2000

Integrated Telecommunications Management Solutions,
by Graham Chen and Quinzheng Kong, 2000

Managing IP Networks: Challenges and Opportunities,
Co-Editors Thomas Plevyak and the late Salah Aidarous, 2003

Next Generation Telecommunications Networks, Services, and Management,
Co-Editors Thomas Plevyak and Veli Sahin, 2010

Introduction to IP Address Management,
by Timothy Rooney, 2010

INTRODUCTION TO IP ADDRESS MANAGEMENT

Timothy Rooney

IEEE Press
Series On
Network
Management

Thomas Plevyak and Veli Sahin, *Series Editors*

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Rooney, Timothy.
Introduction to IP address management / Timothy Rooney.
p. cm.
Includes bibliographical references and index.
Summary: "The book begins with a basic overview of IP networking, followed by chapters describing each of the three core IPAM technologies: IPv4 and IPv6 addressing, DHCP, and DNS. The next three chapters describe IPAM management techniques and practice, followed by chapters on IPv4-IPv6 co-existence, security and the IPAM business case"– Provided by publisher.
ISBN 978-0-470-58588-7 (pbk.)
1. Internet domain names. 2. Internet addresses. 3. TCP/IP (Computer network protocol) I. Title.
TK5105.8835.R659 2010
004.67'8–dc22

2010016634

Printed in Singapore.

10 9 8 7 6 5 4 3 2 1

CONTENTS

PREFACE

Today's IP networks are growing increasingly complex, as new IP services and technologies are deployed. The increasing proliferation of IP-based devices and applications serves only to accentuate the importance of the performance of the IP network supporting these business-critical applications. If end-user devices such as laptops or VoIP phones cannot obtain an IP address via DHCP (Dynamic Host Configuration Protocol), they will be rendered unproductive and users will call the help desk. Likewise, if DNS (Domain Name System) is down, application navigation by name, phone number, or web address will likewise impair productivity and induce help desk calls. Hence, effective IP Address Management (IPAM) has become critical to maintaining high-performing IP services such as data, video, and Voice over IP (VoIP).

The practice of IPAM entails the application of network management disciplines to IP address space and associated network services, namely DHCP and DNS. The consequence of inaccurately configuring DHCP is that end users may not be able to obtain IP addresses to access the network. Without proper DNS configuration, usability of the network will greatly suffer because the name-to-address lookup process may fail. Imagine having to navigate to a website or send an email or an instant message by IP address instead of by name! It's equally important that these DHCP and DNS configurations be based on a common IP address plan, which maps out the IP address hierarchy, subnets, address pools, and domains.

The linkages among the IP address plan, DHCP server configuration, and DNS server configuration are inseparable; a change of an IP address will affect DNS information and perhaps DHCP as well. These critical network functions provide the foundation for today's converged services IP networks, which comprise most enterprise and service provider networks, so they must be managed using a rigorous approach.

This book provides a concise introduction to the technologies of IP addressing, DHCP, and DNS, as well as IPAM practice and techniques needed to manage them cohesively. A companion book, *IP Address Management Principles and Practice*, provides a deeper dive into IPAM technologies and techniques.

The objectives of this book are to help you:

- Learn the basics of IPv4 and IPv6 addressing and subnetting, DHCP, and DNS networking technologies

- Understand IPAM practices, including managing your IP address inventory and tracking of address transactions, such as allocating and splitting address space, discovering network occupancy, and managing faults and performance
- Understand the costs and justifications for properly implementing an IPAM strategy
- Learn about IPv4-IPv6 co-existence technologies and approaches

CONVENTIONS

This book is typeset in Times Roman font. *Times Roman italic* font is used for terms introduced for the first time or to provide emphasis.

To differentiate prose from example configuration information within a DHCP or DNS server for example, Courier font is used in the following manner:

`Courier plain` font — denotes keywords or literal text within a configuration file or screen.

`Courier italic` font — denotes a parameter name that in practice is substituted for a value reflecting the denoted data element or type.

ORGANIZATION

The book begins with a basic overview of IP networking, followed by chapters describing each of the three core IPAM technologies: IPv4 and IPv6 addressing, DHCP, and DNS. The next three chapters describe IPAM management techniques and practice, followed by chapters on IPv4-IPv6 co-existence, security, and the IPAM business case.

Chapter 1 — IP Networking Overview. The opening chapter provides a very basic overview of IP networking, including a discussion of protocol layering, addressing, and routing.

Chapter 2 — Internet Protocol Addressing. Chapter 2 describes the Internet Protocol (IPv4 and IPv6) primarily from an IP addressing perspective.

Chapter 3 — Dynamic Host Configuration Protocol (DHCP). Chapter 3 provides an overview of the DHCP protocol for IPv4 and IPv6 address assignment, including a discussion of basic operation and additional parameter assignment functions core to many advanced IP services such as broadband service or voice over IP.

Chapter 4 — The Domain Name System (DNS). Chapter 4 provides a basic DNS overview, including a discussion of DNS concepts, the basic resolution process, the domain tree for forward and reverse domains, and resource records.

Chapter 5 — IP Address Management Overview. This chapter introduces the concepts of IP address management (IPAM), including its major components, motivation, benefits, and basic approaches.

Chapter 6 — IP Address Management Practices. Everyday IP address management functions are described in Chapter 6, including IP address allocation and assignment, renumbering, moves, splits, joins, DHCP and DNS server configuration, inventory assurance, fault management, performance monitoring, and disaster recovery. This chapter is framed around the FCAPS network management model, emphasizing the necessity of a disciplined "network management" approach to IPAM.

Chapter 7 — IP Address Management Workflow. This chapter describes various approaches to automating IPAM functions through workflow. An introduction to workflow begins the chapter, followed by intra- and extra-IPAM automation examples, benefits, and scenarios. Examples such as IP address requests, Internet Registry reporting, and asset tracking are described.

Chapter 8 — IPv6 Deployment and IPv4 Co-Existence. Chapter 8 describes various technologies and strategies for deploying IPv6 over an existing IPv4 network.

Chapter 9 — Security Considerations. This chapter describes security related topics with respect to DHCP network access control approaches, DNS vulnerabilities and mitigation, and DNSSEC.

Chapter 10 — IP Address Management Business Case. Chapter 10 provides a business-oriented conclusion to the book, describing the business case for IPAM. This includes derivation of the business case cost basis, identification of savings when using an IP address management system, associated costs, and finally net results. An example business case is also provided.

ACKNOWLEDGMENTS

First, and foremost, I'd like to thank the following reviewers who provided extremely useful feedback, suggestions and encouragement in the process: Janet Hurwitz (developer and work/life balancer extraordinaire), Alex Drescher (truly one of the top DNS experts in the world), Brian Hart (tireless IP networking genius), and Michael Dooley (overall great guy with a unique blend of excellent technical, managerial, and inter-personal skills—thanks for the idea for this book!).

I'd also like to thank the following individuals with whom I've had the pleasure to work and from whom I've learned tremendously about communications technologies and IPAM in particular: John Ramkawsky, Greg Rabil, Steve Thompson, Andy D'Ambrosio, Sean Fisher, Chris Scamuffa, David Cross, Scott Medrano, Marco Mecarelli, Frank Jennings, Jim Offut, Rob Woodruff, Ralph Senseny, and those I've worked with at BT Diamond IP, INS and Alcatel-Lucent. From my past life at Bell Laboratories, I thank John Marciszewski, Anthony Longhitano, Sampath Ramaswami, Maryclaire Brescia, Krishna Murti, Gaston Arredondo, Robert Schoenweisner, Tom Walker, Ray Pennotti, and especially my mentor, Thomas Chu.

I'd also like to thank my family, my wife, LeeAnn, and my daughters, Maeve and Tess, for putting up with my endless hours in writer's isolation and for supporting me throughout this experience!

1

IP NETWORKING OVERVIEW

IP NETWORKING 101

Each party engaged in a communication, whether two people speaking or two computers exchanging information, must comply with a set of conventions that govern the rules of such communication. Language and culture generally guide such conventions for human conversation. A *protocol* defines these conventions for computers. And it's usually easier to get computers to comply with these conventions than people. A protocol dictates the sequence and syntax of communications as well as recovery mechanisms required during error conditions. There are actually several protocols or protocol layers that are used during computer communications, each providing a specific set of functions to support a level of commonality for communicating over a variety of media. We'll delve deeper into this later in this chapter, but let's start out with a simple analogy to human communications to introduce the key aspects of Internet Protocol (IP) addressing and why address management is important.

When two people converse, one person may initiate the discussion in one of many ways: by physically approaching the other and speaking, calling him or her on the telephone, sending him or her an instant message, and so on. In each of these scenarios, the initiator of the conversation identifies and locates the

Introduction to IP Address Management, By Timothy Rooney

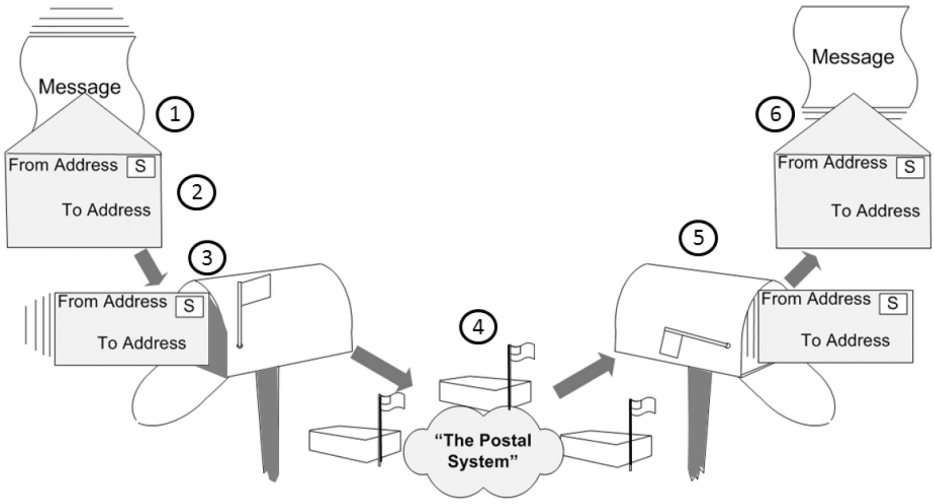

Figure 1-1: The postal delivery analogy

intended recipient, then attempts to begin a conversation using the chosen medium. When I want to talk to my friend Steve, for example, I can look up his number online or in a phone book, dial his number, and when he answers the phone, I can identify myself and begin the conversation. At a basic level, IP communications follows a similar process. When an IP device seeks to communicate with another, it must identify and locate the intended recipient, then initiate communications over a link, while also identifying itself to the recipient in the process.

Perhaps the best, though admittedly overused, analogy for IP communications is that of postal letter delivery. Nevertheless, let's consider this process of "sneaker mail," then relate it to IP communications. The basic postal delivery process is depicted in Figure 1-1, beginning with my writing a letter to Steve and communicating it via postal mail.

After writing my letter, I enclose it in an envelope. This is step 1. Next, I write my return (From) address and Steve's (To) address on the envelope, and stamp it to pay my postal service provider. At this point, I'm ready to mail it, so step 3 consists of depositing my letter in my outgoing mailbox. After my mailperson picks up my letter, the fourth step entails forwarding of the letter within the postal system to the local post office serving Steve's address. After the letter has been delivered to the post office or distribution center serving Steve's address, a local delivery mailperson drops the letter in Steve's mailbox. When Steve walks out to the mailbox, he can open the letter and read my letter. Message delivered!

So let's map this postal message flow to sending a message over an IP network, referring to Figure 1-2. In this case, we're communicating electronically over the Internet, though this analogy holds whether communication ensues over

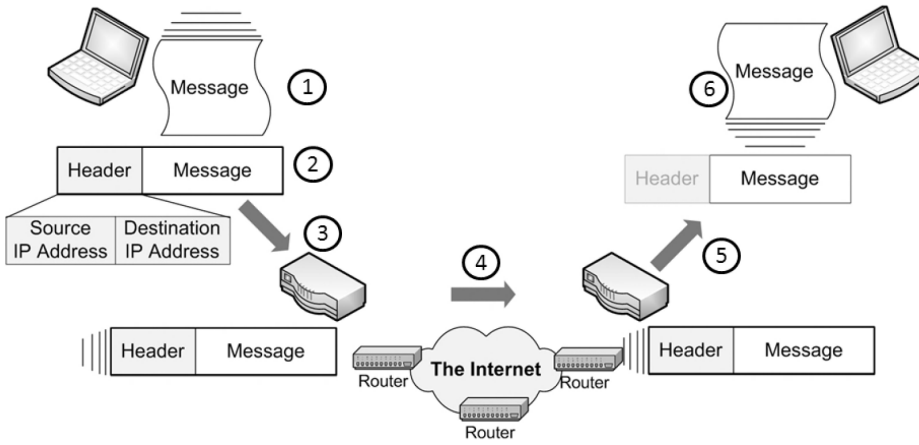

Figure 1-2: Internet Protocol communications

a private enterprise, service provider, home IP network, or a combination thereof. Just as Steve and I have postal mail addresses, we both need *IP addresses* to communicate with each other over the Internet. No one else in the world has the same mailing address as Steve; likewise, no one else in the world has the same IP address as Steve (technically this isn't necessarily true when IP addresses are translated between me and Steve, but let's go with it for now). Let's assume that each of our computers is configured with its respective IP address and that I know Steve's IP address.

Step 1 entails the creation or typing of my message to Steve. In step 2, my computer, knowing my IP address and Steve's, places my message within a data packet, or specifically an *IP packet*. An IP packet is simply the message to be communicated, prefixed with an IP header. The IP header, like our letter envelope, contains my (From) source IP address as well as Steve's (To) destination IP address, among other fields. Having formulated my IP packet, I'm now ready to send it. From my home network, I have a broadband router, to which my computer transmits my IP packet as step 3. This transmission may occur over a cable or a wireless connection between my computer and the router.

In step 4, my router forwards my IP packet to the Internet via my broadband service provider (no stamp required, they'll bill me later). Devices in the Internet called *routers* forward my IP packet ultimately to Steve's broadband service provider and the broadband router in his house. Routers examine each IP packet's header information to determine where to forward the packet to reach its destination IP address efficiently. Having been delivered to Steve's broadband router, step 5 consists of forwarding the packet to Steve's computer, whose IP address matches the IP packet's destination IP address field. In step 6, Steve's computer strips off the IP header to yield the message I had typed. Message delivered!

In both postal and IP communications, the source and destination addresses are specified and are unique, an infrastructure of people and/or machines

TABLE 1-1: Key Similarities Between Postal and IP Communications

	Postal Communications	IP Communications
Message contents	Letter, package, or parcel	Application data such as an instant message
Message container	An envelope or box with To and From street addresses	An IP packet including an IP header with source and destination IP addresses
Sending of the message	Performed by depositing the letter in outgoing mailbox	Performed by transmitting the IP packet from the computer to the local router
Message routing	The letter is physically transported by air, sea, and/or ground via one or more postal offices or distribution centers, ultimately reaching the postal delivery center serving the To address specified	Routers forward the IP packet over a variety of media (e.g., fiber, copper, wireless) to other routers ultimately reaching the router serving the destination IP address
Message delivery	Postal personnel deliver the letter to the street address specified on the envelope	The local router delivers the IP packet to the computer configured with the destination IP address
Message receipt	The envelope is opened and discarded, and the letter is read	The IP header is stripped from the IP packet and the message or packet payload is delivered to the application (instant message window in this case)

successively forwards the message toward its addressed destination, and it is ultimately delivered to the recipient. Table 1-1 summarizes the key similarities among the postal and IP communications examples.

The two core concepts common to these communications analogies are routing and addressing. As we've implied so far, routing is dependent on proper addressing! Let's examine this relationship in more detail.

IP Routing

The postal system operates by "routing" letters and packages as efficiently as possible to regional distribution centers, local centers, and finally to the curb. Scanning and tracking systems along the way direct parcels closer to their ultimate destination via various means of transportation through one or more distribution centers along the way. Typically this routing is performed by first examining the "To" (destination) addressed country, postal code, city and state or province, and finally the street address. The encoding of the general (country, postal code) and the specific (street address) in the "To" address enables different entities in the postal system to use different portions of the

address to route efficiently. Distribution centers can forward packages based only on country and postal code information; once the parcel arrives at a local center serving the destination postal code, the local center then needs to examine the street address for final delivery.

If Steve lives down the street, my letter will simply traverse my local post office, perhaps a distribution center, and back to Steve's local post office for delivery. If Steve lives across the country, my letter will likely route from my local post office through one or more regional centers, then to Steve's local distribution center for delivery. If Steve lives in a different country, my letter will likely be required to enter the country through a customs agent. The customs agent may analyze the letter and either allow its further delivery within the country or deny it by returning it to the sender or confiscating or disposing of it.

Routers perform analogous functions in routing IP packets. Routers mimic the scanning systems of the postal system by examining the network portion of the destination IP address within each IP packet and forwarding it on, getting closer to the destination with each hop. Upon reaching the local serving router, this router examines the full IP address in order to deliver it to the intended recipient. Hence, IP addresses are comprised of a network portion and a host* portion, concatenated together as we'll discuss in the next chapter.

If the packet is destined for a network operated privately, for example, by a corporation or enterprise, the packet will likely meet with examination analogous to the customs agent. And like the customs agent, this enterprise gateway or firewall can allow or deny further transmission of the packet to its destination. By the way, just as storms or other events can cause flight delays or mail reroutes in the postal system, routers can detect analogous outage or congestion events to reroute IP packets as needed. Yes neither rain, nor snow, nor dark of night will stop postal mail or IP packet delivery!

IP Addresses

As we've seen, each device on an IP network must be uniquely identifiable, by means of an IP address. Hence, each device desiring to communicate on an IP network requires an IP address. So your computer at home, your voice-over-IP phone at work, even your cell phone likely has an IP address, at least at the time it's powered up and ready to communicate. In our example above, we assumed that the IP address of each computer was already programmed in, but how does this IP address get in there? The IP address for each device can be assigned and configured in each device either manually or automatically.

The manual address assignment approach using a fixed IP address works well for fixed infrastructure IP devices like routers and servers. But for the vast majority of IP addressable devices such as laptops and cell phones, which are highly mobile, the fixed address assignment approach does not work well. This is because

*The term *host* refers to a device on an IP network. We will use the terms *device* and *host* interchangeably.

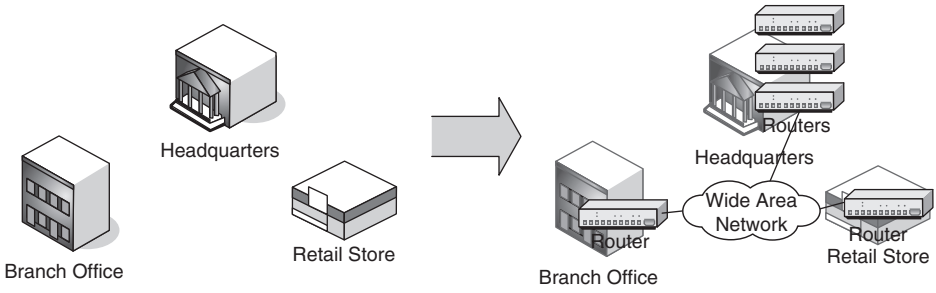

Figure 1-3 (a): Organization's locations (b): Overlay of IP network foundation

the assigned IP address must be relevant to the current network or subnet to which the IP device is connected. If these IP devices move about, they need to be IP addressable within the context of their current location on the IP network, rendering the manual method very cumbersome. Even the postal service doesn't offer a "find me" service to deliver my mail to me wherever I happen to be!

To illustrate this location-sensitivity requirement, consider a small organization with three offices as illustrated in Figure 1-3(a). To enable network communications among these sites, we interconnect them over a wide area network, which may be the Internet or, in this case, a private network from a service provider. To enable communications and routing, we've installed at least one router in each location as illustrated in Figure 1-3(b). This figure shows an overlay of a simple IP network among these locations.

To enable routing among these locations, we need to assign each location a unique set of IP addresses. In this way, the Branch Office will be home for one set of IP addresses (or one IP network), the Retail Store a different set, and Headquarters, yet another unique set of IP addresses. Let's use the set of IP addresses shown for each router as in Figure 1-3(c). Each router will support a set of IP addresses, from which individual IP addresses would be assigned to printers, laptops, voice over IP phones, and other IP devices in that location.

We'll describe the structure and format of IP addresses in more detail in Chapter 2, but an IP address is composed of four numbers, separated by decimal points or dots. Each of the four numbers can range in value from 0 to 255. In our example, IP address sets 10.0.1.0-255, 10.0.2.0-255, and 10.0.3.0-255 have been allocated to headquarters, 10.1.1.0-255 to the branch office, and 10.2.1.0-255 to the retail store. Each of these IP address sets is called a subnetwork or *subnet*, as each represents a portion or subset of the overall enterprise's set of IP addresses.

Note that each set of addresses falls within a contiguous IP address range that corresponds to the network portion of the IP addresses within each set. Recall from our postal analogy that the network portion of the address is used for efficient routing until the IP packet is delivered to the local serving router.

The interconnecting wide area network (WAN) also has a network address (10.254.1.0-255). The routers in each location must be configured with this network information in order to properly route IP communications traffic. In this

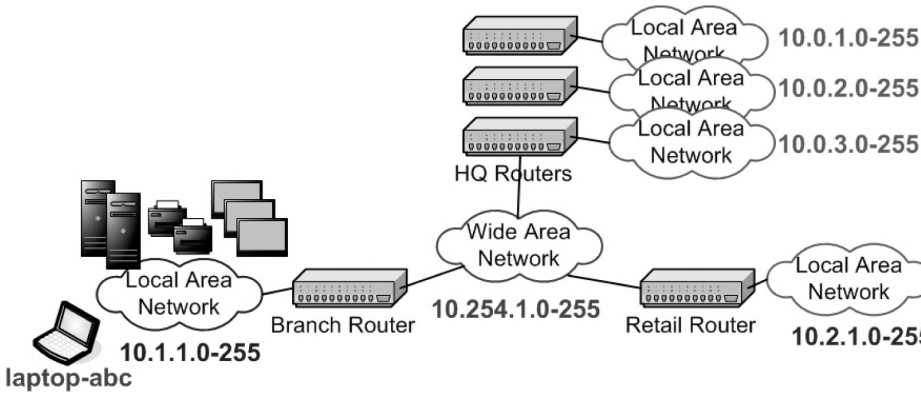

Figure 1-3 (c): More detailed IP network breakdown

way, our Branch Office router is responsible for IP addresses 10.1.1.0-255, and so all IP packets with a destination IP address falling in this range will be forwarded to the Branch Office router. This partitioning of IP addresses to particular sites or routers is analogous to the splitting of geographic locations by zip code and corresponding postal distribution centers.

Now that we've partitioned our IP addresses and configured our routers in accordance with our addressing plan, let's look at address assignment for a given device, say my laptop. Let's say I'm in the branch office on Monday as signified by *laptop-abc* in Figure 1-3(c). I'll need an IP address from the 10.1.1.0-255 subnet, let's say 10.1.1.52. This is because the Branch router "owns" the 10.1.1.0-255 subnet and serves as my "local post office" for delivery of IP packets to devices in the branch office. When I send an instant message to a colleague at Headquarters, my messages are routed to a Headquarters router and responses are routed to me via the Branch router.

Now let's assume I'm called to a meeting at the retail store on Tuesday. When I arrive at the retail store and connect to the store's network with my manually configured 10.1.1.52 IP address, I will quickly realize that I cannot communicate on the network. This is because my IP address, part of the 10.1.1.0-255 network, is served by the Branch router. Thus, when I begin sending an IP communication,* say by opening a web page, entering a www address, my web browser sends an IP packet to the destination web site IP address, using my laptop's IP address, 10.1.1.52, as the source IP address. The web server acknowledges the communication and responds with the requested web page, addressing it back to IP address 10.1.1.52. The routers all "know" that the Branch router delivers IP packets to IP addresses on the 10.1.1.0-255 subnet, so they route the response IP packet intended for my web browser back to the branch office!

*Technically my IP communication will not even get past the Retail router which would drop my packets because my source IP address is not valid for the retail subnet. But this simplified example illustrates the fallacy of this communications attempt even if it would be technically feasible.

From my perspective, I'm not getting a response from the web server. Is the network down or is the web server down? As I call the help desk complaining about the network outage, the network team eventually discovers that my IP address is not appropriate for the subnet to which I am connected. They walk me through the cumbersome process of manually changing my laptop's IP address to 10.2.1.187 (only to walk me back through the reverse process when I return to the branch office on Wednesday). Once my address is relevant to my location, I'm able to communicate bi-directionally with the web server and other IP application servers such as email servers because my address falls within the set assigned to the local serving router.

This simple example illustrates the importance of not only having an IP address for IP communications, but one that's appropriate to the subnet to which you're connected. Thus, with even minimal mobility of associates going to meetings, visiting customers, or generally traveling about, the clumsy manual help desk process outlined above is impractical. To require people to call the help desk when they need a new IP address (not only when the network really is down) reduces end user productivity (and patience!), as well as increases the costs of help desk operations and network and server technical support. Plus it's extremely difficult to walk a "technically challenged" person through the process of manually entering an IP address on a device! And as the variety of devices that people use to connect increases, the variety of IP address entry methods likely increases as well, adding to the support team burden. Clearly, an automated mechanism for assigning a unique IP address relevant to the subnet of connection is crucial to reducing costs while maintaining overall end user and support staff satisfaction and productivity. The Dynamic Host Configuration Protocol (DHCP) is one such mechanism that enables an IP device connecting to a network to automatically obtain a unique and location-relevant IP address.

After the address assignment problem is solved, our next objective is to eliminate error-prone human entry of IP addresses. Earlier, we glossed over the fact that I already knew Steve's IP address or that the www address I typed was magically translated into an IP address to enable creation of an IP packet. So if communication on an IP network requires IP addresses, how do we get away with entering text names to send emails or connect to websites? The solution is the Domain Name System (DNS), which enables users to enter names for services, web sites, or email boxes, obviating the need to enter IP addresses.

We generally take this for granted, which is a good thing! Imagine carting around the equivalent to an Internet phone book with web sites and associated IP addresses. DNS works "behind the scenes" to provide a name-to-address look-up mechanism to bridge this gap between human consumable names to network consumable IP addresses. Unless you're a numbers wizard, entering http://www.ipamworldwide.com in your web browser is vastly more simple than entering (and remembering) http://192.0.2.201. Fortunately, this usability problem was recognized early in the development of the Internet and DNS was devised to automate this directory lookup function.

Once you type in a www address, your computer looks up the www address in DNS and obtains the corresponding IP address, which it then uses as the destination IP address in the IP header. Other forms of name-to-address translation have been developed over the years, including hosts files, Yellow Pages (YP), and Windows Internet Naming Service (WINS), but DNS has sustained its dominant status as the de facto, production network-proven name-to-address translation service for IP communications today. Before further exploring the core elements of IP address management, namely IP address allocation, DHCP, and DNS, let's take a more detailed look at the basics of IP networking by exploring the inner workings of how routers deliver IP packets to their respective destinations.

IP NETWORKING 102

After reviewing our simple IP addressing example above, you may be wondering, why don't we just eliminate the routers and use one massive network that everyone can share instead of employing this subnetting process that leads to readdressing of my laptop when I move from location to location? After all, I could just use one IP address anywhere in the network. While this approach is theoretically possible utilizing a bridged network across all of these sites, this does not scale well because performance of communications will suffer with growing intersite distance and number of devices on the network. This is due to that fact that in a shared or bridged network, every device's messages are sent to every other device on the network. And because the network is a shared medium, collisions may result when two or more devices attempt to send messages at the same time.

Collisions in the networking world have the same effect as those in interpersonal communications. As more members join the "conversation," collisions arise more frequently. As collisions occur parties to the conversation (at least the polite ones) back off momentarily before reattempting to initiate communications. The more parties involved in the conversation, the more frequently collisions occur, and the larger the backlog of messages awaiting communications. This attempt/back-off/reattempt process escalates quickly until ultimately no party attempting to communicate gets a message to a recipient. As the backlog of reattempts to communicate builds, frustration escalates and gridlock results. The same effect occurs when too many devices attempt to communicate on a large, monolithic, shared medium. And the same general solution, which for human conversation consists of groups naturally branching off from the main conversation into subgroups of smaller sets of people, can be applied to IP networks.

By reducing and limiting the number of parties communicating on the same medium, we can localize the *collision domain* and reduce the number of collisions and backlog on the network as a whole. The deployment of switches and routers supports this partitioning of the network into separate collision domains. Switches enable partitioning of the collision domain itself by virtue of interconnecting

pair-wise switch ports, each of which is physically connected to a host. Routers provide collision domain boundary points by terminating yet interconnecting collision domains. Routers in particular leverage the concept of protocol layering to separate collision domains. Let's review the concept of protocol layering, which will lead us back to a more detailed discussion of the roles of switches and routers.

Protocol Layering

The International Standards Organization (ISO) has defined a layered protocol model, separating responsibilities for different aspects of controlling communications (1). The layered model consists of seven layers and is denoted the Open Systems Interconnect (OSI) model. The term *protocol stack* refers to the fact that several protocol layers are "stacked" one upon another to usher data and commands from my web browser onto the wire or over the air and through the network to the destination. In fact, the Internet Protocol's ability to run over various media such as these and others is a powerful consequence of the use of a layered protocol stack. The OSI model enables a common implementation of the Internet Protocol across a variety of lower layer data link and physical layers, including cable, Digital Subscriber Line (DSL), fiber, Ethernet, wireless and so on. Figure 1-4 illustrates the OSI model with a brief summary of key functions of each layer.

Application Layer (Layer 7). The application layer provides the primary end user exposure and functionality. A web browser, file transfer program, or email client are examples of applications.

Presentation Layer (Layer 6). The presentation layer is responsible for defining the data format and syntax between application endpoints in the communication. For example, this layer specifies standard graphics formatting and translation for communications across the network.

OSI Layer	Key Roles
Application	End user application network interface; e.g., email, web
Presentation	Application data translation as necessary
Session	Regulation of communications between two applications across the network
Transport	End-to end error recognition and recovery; e.g., TCP, UDP
Network	Message addressing, route determination, flow control; e.g., IP
Data Link	Framing of bits and error-free transmission of frames; e.g., Ethernet, frame relay
Physical	Bit transmission techniques; e.g., twisted pair, coax, air

Figure 1-4: OSI Protocol stack summary (1)

Session Layer (Layer 5). The session layer is responsible for regulating the end-to-end communications of applications across the network, providing such services as security and authentication. NetBIOS is an example of a session layer protocol.

Transport Layer (Layer 4). The transport layer is responsible for end-to-end communications integrity, assuring flow control between the two endpoints, as well as data checking, requesting retransmissions, and proper ordering of information. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are transport layer protocols within the TCP/IP stack.

Network Layer (Layer 3). The network layer is responsible for the formatting of information into packets and/or packet fragments for communications and for routing over one or more networks. IP is a network layer protocol.

Data Link Layer (Layer 2). The data link layer is responsible for formatting of information into frames for communications over the physical network, including error checking for data integrity. Ethernet, Token Ring, ATM, and frame relay are examples of data link layer protocols. The data link layer is commonly split into Logical Link Control (LLC) and Media Access Control (MAC) sublayers. The familiar term "MAC address" refers to a device's layer 2 or media access control address.

Physical Layer (Layer 1). The physical layer defines the electronic interfaces and characteristics including voltage and current specifications for transmission of data and control (e.g., preamble) bits. EIA-232 (RS-232) provides an example of a physical layer specification.

OSI and TCP/IP Layers

These protocol layers not only permit interoperability for multiple applications and underlying physical networks, they also segment the responsibility required for successfully communicating over a data network. For example, some layers such as the data link and network layers, provide error checking and correction to facilitate accurate communications and reduce retransmission requirements. Others, such as the transport layer, are responsible for end-to-end communications integrity and proper ordering of information. Overall, the standardization of protocol layer definitions enables successful end-to-end communications while facilitating interoperability.

This seven-layer stack shown in Figure 1-4 is sometimes portrayed as a five-layer stack in the Internet context with the Application layer sitting above TCP, IP, data link, and physical layers, respectively, as shown in Figure 1-5. Protocol layering enables not only the transmission of IP packets over a variety of media, it also permits a variety of end user applications to communicate over IP, which in turn run over various lower layer protocols. For example, an email client

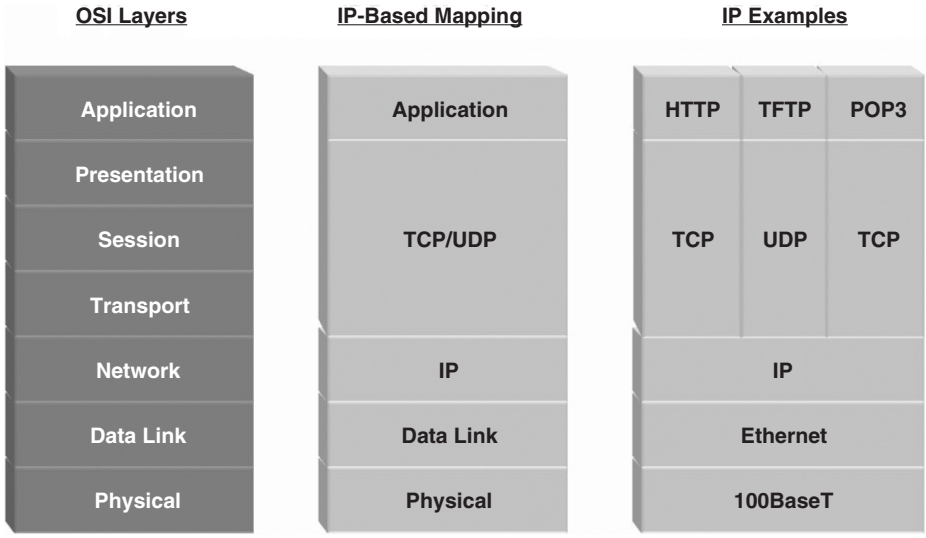

Figure 1-5: The OSI and TCP/IP Protocol stacks

application can communicate to an email server using a Post Office Protocol version 3 (POP3) application, which is layered on TCP, and in turn IP, which can then be layered on an Ethernet, Token Ring, frame relay, or other layer 2 protocol, and ultimately a particular physical layer. Another example illustrated in Figure 1-5 features HTTP (web browsing), TFTP (trivial file transfer protocol), and POP3 running over respective transport protocols and IP, Ethernet, and 100BaseT. This provides a seamless end user experience in using common "desktop applications" whether communicating over an Ethernet 100BaseT network, an 802.11 wireless network, or an ATM-based DSL network.

Layering also enables components of the stack to be developed and offered by different organizations. In practice, protocol layering works effectively at the application to TCP/IP boundary and the TCP/IP to data link boundary, though TCP and UDP generally operate exclusively with IP at the transport-network layer boundary. An application programming interface (API) enables a variety of applications from different vendors to utilize common function calls into the TCP/IP stack, which is commonly included with the operating system. The de facto API for TCP/IP applications is the *sockets* interface originally implemented on BSD UNIX (on which DNS BIND was also originally implemented). The sockets interface defines program calls to enable applications to interface with TCP/IP layers to communicate over IP networks. Microsoft's Winsock API is also based on the sockets interface.

Intra-Link Communications. To get our arms around how data flows through these layers within protocol stacks, consider Figure 1-6. Beginning with the application data in the upper left of the figure, notice the addition of headers

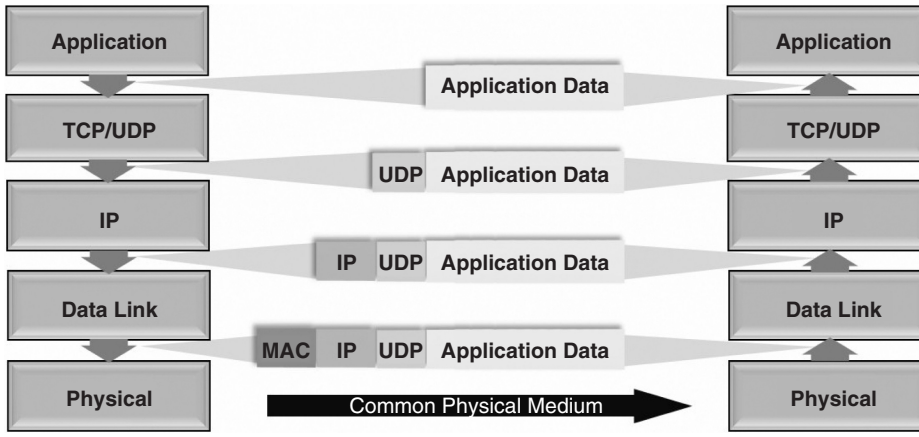

Figure 1-6: Protocol layering

as the application data payload traverses down the stack. The data link frame that is transmitted on the local network encloses the application data and upper layer headers. The frame recipient on the right side of the figure then processes and strips off each header at corresponding layers as the data is passed up the stack to the destination application. The headers and payload shown at each layer boundary are exactly the same on both ends of this intra-link communication.

As application data is prepared for transmission on an IP network using a TCP/IP stack, the application calls the sockets API to communicate the data down the stack. A TCP or UDP header is added based on application selection, and then an IP header is appended. Recalling our earlier postal analogy, we can now see that the IP header was just one such header added to a network message. The "message" to which the IP header was appended included the application data (e.g., the instant message text) plus the transport layer (TCP or UDP) header. In this multi-layered stack model, each layer adds a header to enable it to perform its respective function.

The application generally dictates the UDP or TCP header parameters, which consists of application-specific port numbers as well as checksum, flow control, and other data. As we've discussed, the IP header contains the sender's and the receiver's IP addresses. The sender's IP address is assigned manually or automatically (e.g., via DHCP) and the recipient's IP address is entered either via the application user interface or is fetched from a DNS server.

Below the IP layer, the IP packet, which itself comprises and IP header, a TCP or UDP header, and application data, is enclosed within a data link layer frame for transmission over the physical network. Communication on a given data link requires encapsulation of the IP packet within a data link frame, which itself requires source and destination data link (MAC) addresses. To transmit the IP packet within a data link frame, the transmitting host must determine the recipient's data link (MAC) address. So the device must map the destination IP

address to a destination MAC address for transmission within a layer 2 frame on the physical network. But another wrinkle arises depending on whether the destination IP address is on the same data link as the sender.

Are We on the Same Link? Generally each device can determine if the intended recipient resides on the same link by virtue of its configured IP address and subnet address range. So if the destination IP address falls within the same subnet range as the sender's IP address, the device is considered on the same link. Otherwise, if the intended recipient does not reside on the same subnet, the source device must identify a router on the link that can forward or route the data to the intended destination. Using either a routing table, or for most non-router devices, a configured *default route*, the device can determine the next hop to which to send the data. A default route is the next hop destination to which all packets are sent in the absence of a known next hop toward the intended destination address. This is kind of like my outgoing mailbox in the postal analogy, which is where I place all of my outgoing communiqués. This default route, commonly referred to as a default gateway, is typically the IP address of a router serving the subnet to which all nonlink local packets are to be sent.

For example, per Figure 1-3(c), if *laptop-abc* transmits data to another device within the branch office, that is, with destination IP address in the range of 10.1.1.0–10.1.1.255, it can be sent directly (intra-link); if it transmits data to a device in HQ or the retail store, it must send it to the branch router for routing. Whether sending the data to an intra-link destination or a router, the sending device must determine the recipient's or router's data link (MAC) address in order to formulate and transmit the IP packet within a data link layer frame. This MAC address is the data link layer address for the intended device. But we determined the destination IP address from DNS, not the MAC address; so how is this MAC address determined?

The *Address Resolution Protocol (ARP)* enables a device to determine the MAC address of a device on the same link corresponding to an IP address. The device transmitting the message knowing the intended next hop IP address (recipient on-link or router) formulates an ARP data link broadcast frame requesting MAC resolution of the IP address. A *broadcast frame* is a data link frame addressed to all devices on the link. For Ethernet data links, for example, a broadcast frame uses the destination Ethernet address of all 1's, meaning the broadcast address for all devices connected to this Ethernet link. The device that is configured with the sought IP address responds with an ARP response frame indicating its corresponding layer 2 address. This enables the source device to formulate the Ethernet frame for transmission of the IP packet. Most devices cache this information in an ARP cache, providing a temporary storage of this IP-to-MAC address correlation to reduce repeated ARP queries, for example, for multi-frame communications.

Limiting Broadcast Domains. Data links like Ethernet comprise the collision domains (also referred to as broadcast domains) we referred to earlier. The

data link layer is chiefly concerned with accessing the network for transmission, detecting collisions, and performing error checking on frames. All devices connected to a common Ethernet network receive frames sent from every other connected device. Hence it's easy to see how our earlier analogy about backlogs and gridlock as the number of parties attempting to communicate holds true. As the number of devices within the collision domain increases, and/or the number of communications attempts per device increases and/or the volume of communications per device increases, the more likely data collisions will occur, degrading network performance. If we can confine the number of participants in each collision domain, we can improve overall performance and end user productivity and satisfaction.

This brings us back to how switches and routers constrain broadcast domains. Historically, Ethernet local area networks (LANs) were deployed with wiring to each office, cubicle, or end station funneled back to one or more Ethernet hubs. Hubs literally broadcast frames received on any given port to every other port, thus comprising an indivisible collision domain. Switches were developed to directly interconnect (or switch) traffic between source and destination ports without blindly broadcasting all data to all ports. Switch ports effectively provide a direct point-to-point connection between the end device and its switch port. The switch detects the MAC address of each connected device then leverages this information to directly interconnect the port on which the frame is incoming to the appropriate destination port. This minimizes superfluous broadcast traffic to devices on all other ports. Of course layer 2 broadcast traffic is broadcast to all switch ports, but this is certainly an improvement!

Modern layer 2 switches have further evolved to enable definition of a subset of physical ports to a given broadcast domain. Thus, instead of hardwiring hub ports within a broadcast domain or LAN, one could partition which switch ports belong to independent broadcast domains. These independent broadcast domains are referred to as virtual LANs (VLANs), as the switch supports multiple logical LANs on one physical switch device. Broadcast traffic on ports within VLAN 1 will not be broadcast to switch ports associated with VLAN 2 for example. Consider Figure 1-7 for an example of VLAN segmentation. The figure shows the common implementation of associating different VLANs with different

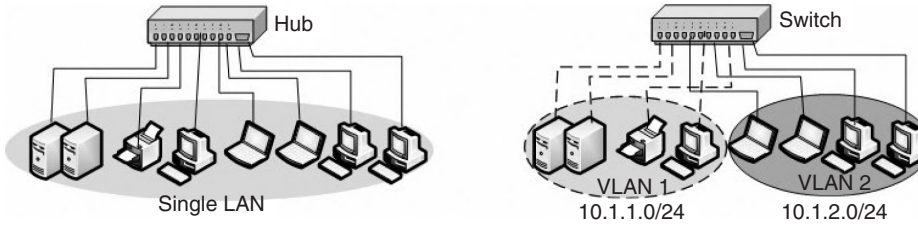

Figure 1-7: Hub vs. switch/VLAN architecture

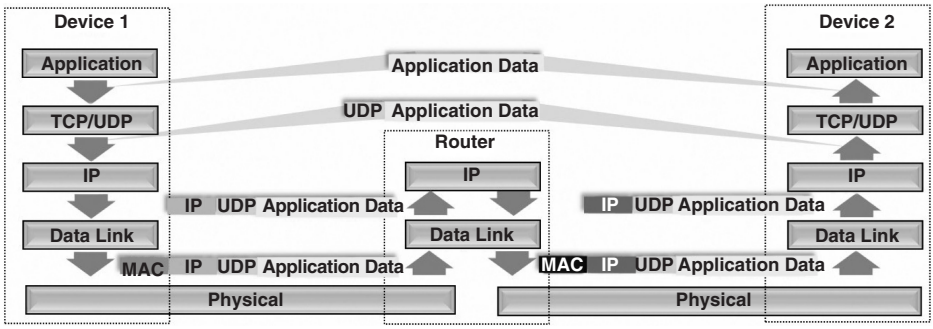

Figure 1-8: Protocol stacks with an intervening router

subnet addresses.* This is contrasted against the single broadcast domain per hub shown on the left of the figure.

Inter-Link Communications. Switches certainly help reduce the scope of collision domains, but there are only so many switch ports! The second method used to limit broadcast domains leverages protocol layering concepts and employs routers to separate layer 2 networks. Consider Figure 1-8, which is a recasting of Figure 1-6 with an intervening router. The left side of the figure is identical to that in Figure 1-6. Each protocol layer adds its respective header as the application data moves down the stack. Finally the Ethernet frame is sent over the physical [layer] network to a router. The router also has a protocol stack, but it consists only of layers 1–3 (routers generally do provide a configuration "application" layer but, for the purposes of this discussion, we'll consider their routing function only). As the router's data link layer checks the frame integrity, it strips off the layer 2 frame header (and footer) then passes the IP packet to the IP layer. The router analyzes the source and destination IP addresses (and potentially other IP header fields) to determine where next to route the packet. That is, the router is only an intermediate point on the way toward the ultimate destination, hence its protocol stack limitation to layers 1–3.

After the router determines where next to route the IP packet, it appends a modified IP header (note the differently shaded IP header in the figure—the source and destination IP addresses remain the same but at least one other field is changed, the Time to Live (TTL) field, which is decremented by each router along the path of the packet so that packets don't wander the Internet aimlessly forever). The router then passes the IP packet to the data link layer. The data link layer encapsulates the IP packet within a link layer frame appropriate to the corresponding link and physical layers over which the message will be routed. The router uses its source link layer (MAC) address based on its chosen outgoing interface and identifies the destination link layer address based on ARP or ARP

*Note that a single subnet address can be shared across multiple VLANs with the use of VLAN IDs in accordance with IEEE 802.1q.

cache mapping of the next hop IP address to its corresponding MAC address. Thus the router may receive the incoming packet on the left over a wireless link, then transmit it as modified appropriately over an Ethernet link.

Upon receiving the link layer frame, Device 2's protocol stack processes each successive layer, passing it up its stack to the intended application. Note that the data link and IP layer frames and packets respectively vary link-by-link. However, the transport layer message and application data itself are identical at both the sending and receiving ends of the connection. Of course, the goal is to transmit the application data intact, and the identical TCP/UDP layer message enables the intended end-to-end processing required of the transport layer.

We can now conclude that the router serves to terminate the data link layer or collision domain on the left side of the figure, only to modify the IP and MAC headers then forward the message to a second collision domain on the right side of the figure. For this reason, routers are also known as gateways, serving as gateways between layer 2 collision domains and IP networks. While switches utilize port VLAN configurations to separate collision domains at layer 2, routers use IP subnets to differentiate collision domains at layer 3.

Worldwide IP Communications. Let's extrapolate our view from Figure 1-8 from two devices interconnected via a single gateway to a scenario where I'm using my PC at home over my broadband connection to access a web server halfway around the world, as shown in Figure 1-9. When I browse the web site hosted on the server, the site address I type or click is captured and formulated into IP packets that are then sent via my broadband router to my service provider's router, Router A.

This transmission follows the same process we reviewed as shown on the left half of Figure 1-8. Router A then routes my data packet via additional intervening routers through the Internet to the ultimate destination, served by Router G. On each link along the way, each router terminates the layer 2 frame and IP packet, determines where next to route it (next hop IP address from a particular outgoing interface), and formulates a corresponding frame for transmission to

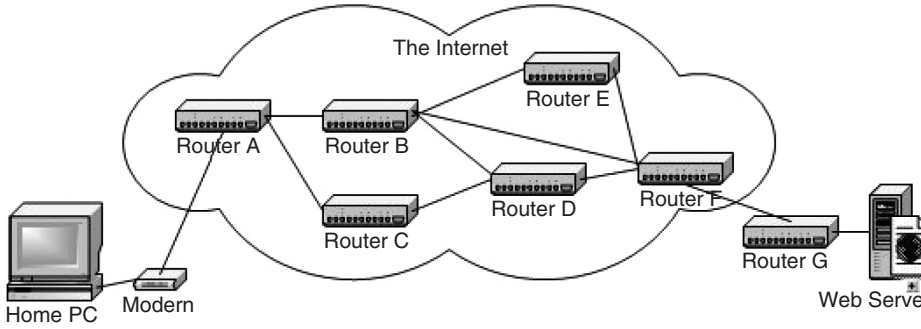

Figure 1-9: Simple view of an IP network such as the Internet

the next hop. Notice that there are multiple paths from my PC to the web server in Figure 1-9. One of my packets may travel from end to end over path A-B-E-F-G, while the next packet may take a different path, say A-C-D-F-G, and so on. Each IP packet is routed independently through the network.

Contrast this with an old-fashioned circuit-switched telephone call that temporarily "nails up" a dedicated connection through the traditional telephone network from my phone to the phone of the person I'm calling. Since our voice conversation requires set-up of a connection that comprises a physical path through the telephone network for the duration of our discussion, this type of connection is referred to as "connection-oriented." At the IP layer, IP does not establish a connection prior to communicating, and each IP packet is routed independently: all packets may happen to follow a common end-to-end path, they may all take different paths, or more likely, somewhere between these two extremes. IP is therefore considered "connectionless."

Connection-oriented communications generally provide a more reliable method of communications at the expense of tying up network resources and not dynamically rerouting around intermediate failure points during the connection session. The term *reliable* in this context means that there are means to detect and recover from dropped packets or packet fragments. "Pardon me?" usually works during a voice conversation and certain protocols such as TCP include an analogous construct with positive acknowledgment. While IP itself is considered an *unreliable datagram service*, the transport layer above IP, namely TCP, can be used to overlay connection-oriented controls to provide reliable communications between two devices or hosts. Without nailing up a physical connection, TCP provides mechanisms to properly order incoming IP packets and to request retransmission of IP packets should one or more get lost along the way. UDP is an alternative connectionless transport layer protocol within the TCP/IP protocol suite that provides unreliable data delivery. Conversely, a pseudo-layer 2 technology, multi-protocol label switching, MPLS, can be used to logically "nail up" a connection path over which IP packets can be transmitted between two endpoints. And finally, an IP static route could be configured along the path of intervening routers to deterministically route packets. Static routing emulates a connection-oriented session, at the expense of the connectionless rerouting advantage.

Dynamic Routing. The advantage of employing a connectionless scheme for IP is that a link outage between routers in the path of communications can be automatically detected and packets rerouted, keeping the lines of communication open. If you've ever been on a phone call and the call dropped, you've experienced this disadvantage of connection-oriented communications: if a link along the path fails, the entire session fails. Connectionless communications provides automated routing around outages, and this resilience was, in fact, one of the key design goals of the Internet Protocol.

Each router along the communications path of an IP packet examines the destination IP address in the IP header to identify whether it directly serves the network on which the destination IP address resides, or failing that, to which

router to forward the packet that is "closer" to the ultimate destination. Each router consults an internal *routing table*, which stores information about where next to route packets destined for various IP networks. The routing table within each router governs routing decisions on each incoming IP packet and generally indicates one or more *next hops* in the path for a given destination network. A next hop is another router to which a given router can forward the packet directly. That is, the next hop is an adjacent or directly connected router, which itself may be directly connected or be multiple (hopefully fewer) hops away from the destination. In this way, each router need not be aware of every other router on the Internet; instead a given router must simply know where next among its directly connected peers to send a packet to get it closer to its ultimate destination.

A *routing protocol* is used by each router to periodically communicate with its neighboring routers to obtain their current routing and reachability information to keep routing tables up-to-date. Hence, dynamic routing makes use of recently updated routing information to make next-hop routing decisions. If a link or router fails, reachability changes will be detected and updated reachability metrics will ripple through the routing infrastructure via the chosen routing protocol. Routing protocols define the format and rules governing this "background" communication among routers, which enable each to maintain its routing table with the latest reachability information.

For example, considering Figure 1-9, Router B will receive advertisements of reachability to the network on which the web server resides from Routers D, E, and F. None of these routers directly serves this network but they offer an intermediate path. Using a simple hop count distance metric, Router F advertises a hop count of 2, while Routers D and E advertise a hop count of 3. Presumably the chosen next hop will be closer to the intended destination, that is, Router F, though other factors such as packet traffic congestion may come into play. More sophisticated metrics beyond hop count are now taken into account with modern routing protocols. Upon receiving the packet from Router B, the chosen next hop router then performs the same basic algorithm to determine whether it directly serves the IP network or if the packet must be sent on to another router, for example, Router G. Ultimately, the packet should reach a router serving the intended destination IP address for delivery.

Two types of routing protocols are generally used by an organization. *Interior* routing protocols enable routers within an organization to communicate subnet reachability. Popular interior routing protocols include Routing Information Protocol (RIP/RIP-2), Enhanced Interior Gateway Routing Protocol (EIGRP from Cisco Systems Inc.), and Open Shortest Path First (OSPF). *Exterior* routing protocols, such as Intermediate System to Intermediate System Protocol (IS-IS) or Border Gateway Protocol (BGP), enable updating of reachability and metric information across organizations or routing domains. Reachability to my network is communicated by my routers (or my Internet Service Provider) using an *exterior* routing protocol. BGP is the de facto Internet standard exterior routing protocol.

Organizations typically run an interior routing protocol on their internal routing network and run BGP on their externally facing router interfaces, for example, those connected to their Internet Service Providers (ISPs). However, BGP is not necessary for organizations with a single ISP connection not providing downstream routing, say, to other organizations. For example, BGP would not be required for a small office with a single-ISP connection. Such an end user is considered single-homed in contrast to a multi-homed organization with multiple Internet connections to one or multiple ISPs. BGP summarizes reachability information for the organization, which is identified via an Autonomous System (AS) number. AS numbers are simply organizational identifiers and are distributed and managed by Regional Internet Registries (see Chapter 2) to uniquely identify an organization or more accurately, a routing domain.

What is meant by the statement that "BGP summarizes reachability information"? When communicating (technically termed *advertising*) subnet reachability to the Internet using an exterior routing protocol, the routers do not list every subnet by hop number. This would create a massive amount of messaging overhead. A process called *summarization* or *aggregation* enables the communication of a single network address on the Internet for each such contiguous set of IP addresses. This is kind of like routing letters to a zip code distribution center. The center at the other end of the country needs only route to the destination postal code center and then allow that center to perform local delivery. Likewise, summarization enables routers to identify a contiguous set of IP addresses as a single network address along with its relative proximity or distance instead of communicating such reachability for every single IP address or subnet. Typically, an organization obtains a set of Internet addresses from an Internet Registry or ISP and simply advertises reachability to all such addresses by network address. It's then up to the organization to carve up this network allocation internally for Internet reachability as needed. This block allocation process is one of the key processes of IP address management as we'll discuss in detail in Chapter 6. A similar summarization process is utilized by interior routing protocols as well.

Routers and Subnets. As routers serve as inter-link gateways and forward or route data based on layer 3 (IP) information, each link needs to be assigned a set of IP addresses, a subnet address. Each device on a given link will require an IP address from the corresponding subnet address associated with the link and will utilize ARP to identify a MAC address corresponding to the IP address to which to transmit IP packets over the link. The process of IP subnetting entails the partitioning of an IP network into contiguous sets or blocks of IP addresses which are then associated with each link or subnet. Based on the subnet plan, a key element of IP address management, routers can be provisioned accordingly.

Provisioning a subnet on a router is akin to the addition of a new housing development for postal delivery. Just as the postal system must be updated to reflect the newly available addresses, Internet routers must likewise be updated. Fortunately, in both cases, this is usually a simple process. In the postal case, most new neighborhoods fall within an existing postal code and as long as the

rest of the "postal system" can continue to deliver letters to the postal distribution center serving this zip code, updating is limited to systems and personnel within the local center or zip code.

In the Internet world, each organization desiring to communicate on the Internet needs a set of Internet-unique IP addresses. An organization's set or block of IP addresses can be likened to a zip code. Any IP communications destined for devices within an organization's set of addresses are routed to the organization's routers, akin to zip code distribution centers. The organization's routers then handle "local delivery" within the organization. Hence, the addition of a subnet to a router's configuration affects only intra-organization routers, which need to be updated via the interior routing protocol to identify which router serves the new subnet.

Referring back to our example network in Figure 1-3(c), the Branch office router advertises direct reachability to the 10.1.1.0/24 network, not a listing of 0-255 IP addresses; this reduces the size of routing tables and update messages from 256 to 1 for each such subnet, thereby reducing overhead and improving router performance. And this is why we don't allow the Retail Store router to contend with the Branch office router for serving the IP address 10.1.1.52 when I'm at the Retail store for a day: the overhead in routing protocol messages would create needless traffic at the expense of end user productivity traffic. Overhead is also minimized by virtue of the fact that the router only analyzes up to the IP layer in the protocol stack. Without having to fully digest each frame, it is able to quickly discern where next to route the message.

In the case of our HQ networks, if these three networks were served by a single router, the router could communicate its proximity to these networks in one statement* instead of $3 \times 256 = 768$ for each IP address, or even just three, one for each network. Organizing your address space to promote this router aggregation is important to keeping routing protocol communications small to speed up communications of updates and routing outages while enabling scalability.

Within an organization, address space planning must consider address capacity needs for the IP device population in the context of the organization's routing topology. As we'll see in later chapters, alignment of address allocations with routing topology produces an efficient address plan that will minimize routing protocol update traffic and routing table sizes.

HIGHLIGHTS OF INTERNET PROTOCOL HISTORY

The Internet Protocol (IP) is the most widely deployed network layer protocol worldwide. Emerging from a U.S. government sponsored networking project for the U.S. Department of Defense begun in the 1960s, the TCP/IP suite was initially implemented in the early 1980s and has evolved and scaled to support

*An example of the sole summarized route in this case would be 10.0.0.0/22, though the mask size in this example network could vary. This will be discussed in detail in Chapter 2.

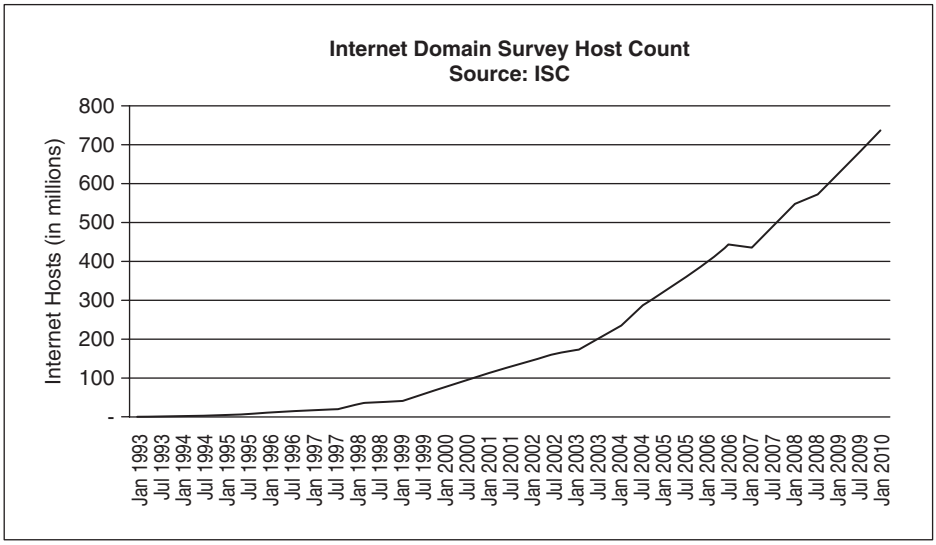

Figure 1-10: Growth of Internet hosts 1994–2009 (2)

networks from hundreds of computers to hundreds of millions today. In fact, according to Internet Systems Consortium (ISC) surveys (2), the number of devices or hosts on the Internet exceeded 730 million as of early 2010, with average annual additions of about 75 million hosts *per year* over each of the last five years (see Figure 1-10). The fact that the Internet has scaled rather seamlessly from a research project to a network of over 730 million computers is a testament to the vision of its developers and robustness of their underlying technology design.

The Internet Protocol was "initially" defined in 1980 in Request for Comments (RFC) 760 (3) and 791 (4), edited by the venerable Jon Postel. I've quoted "initially" because as Mr. Postel pointed out in his preface, RFC 791 is based on six earlier editions of the ARPA (Advanced Research Projects Agency, a U.S. Department of Defense agency) Internet Protocol, though it is referred to in the RFC as version 4 (IPv4). RFC 791 states that the Internet Protocol performs two basic functions: addressing and fragmentation. While this may appear to trivialize the many additional functions and features of the Internet Protocol implemented then and since, it actually highlights the importance of these two major topics for any protocol designer. Fragmentation deals with splitting a message into a number of IP packets so they can be transmitted over networks that have limited packet size constraints, and reassembly of packets at the destination in the proper order. Addressing is of course one of the key topics of this book, so assuring unique addressability of hosts requiring reachability is critical to basic protocol operation.

The Internet has become an indispensable tool for daily personal and business productivity with such applications as email, instant messaging, web

browsing, and wireless, video, and voice communications. The Internet has indeed become a key element of modern society. The term "Internet," evolved from the lower case form of the term used by the early developers of Internet technology to refer to communications among interconnected networks or "internets."

Today, the capitalized "Internet" refers to the global Internet that we use on a daily basis, and it truly is a massive network of interconnected networks. Most enterprise networks also utilize the TCP/IP protocol stack, as do broadband, wireless, and landline service providers. Getting all of these networks and the hosts on them to cooperate and exchange user communications efficiently requires a robust set of rules for such communications. The Internet Protocol has proven remarkably robust, as well as extremely versatile and scalable.

2

INTERNET PROTOCOL ADDRESSING

THE IP HEADER

In Chapter 1, we introduced the concept of protocol layering. The network or IP layer in our case adds a header to the data it receives from the TCP or UDP transport layer. This IP header is analyzed by routers along the path to the final destination to ultimately deliver the IP packet to its final destination, identified by the destination IP address in the header (Figure 2-1). The source IP address is also included in the IP header and may also be used in routing decisions.

Before we analyze the structure of an IP address, let's digress for a brief review of binary arithmetic, which will play an important role as we continue the discussion.

BINARY REVIEW

If base 2 arithmetic excites you, then you'll appreciate this T-shirt slogan: *there are only 10 kinds of people in the world: those that understand binary and those that don't*. For the rest of us, binary arithmetic is a bit perplexing. This section provides a brief review of translating bits into decimal or "normal" numbers.

Introduction to IP Address Management, By Timothy Rooney

Figure 2-1: IP packet

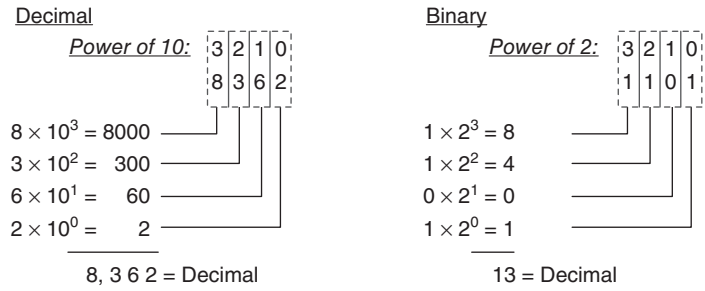

Figure 2-2: Binary to decimal conversion

Computers, network devices, and electronic devices communicate via binary digits, where a digit can have only one of two values: 0 or 1. With one bit, I can have $2^1 = 2$ values, 0 or 1. If I grouped two bits together, I can double that, $2^2 = 4$ values: 00 (0), 01 (1), 10 (2), 11 (3). As with base 10 or decimal arithmetic, where we can determine the quantity or decimal equivalent by adding the ones (10^0), tens (10^1), hundreds (10^2), etc. digits, in binary, we add the appropriate power of 2 for each digit. Instead of counting each digit in powers of 10 as we do with decimal numbers, we use powers of 2. Consider the example in Figure 2-2.

While the algorithm is similar, using powers of 2 versus powers of 10 can be a bit more confusing and hence error-prone. Alas, with practice, it becomes easier. Computing and communications devices typically group these binary digits into groupings of 8 bits called octets or bytes. While the example above shows how to calculate the decimal equivalent of 4-bit grouping, calculation of an 8-bit octet just extends this out to the 7th power of 2. It's left to the reader* to practice extending this algorithm to the full 8 bits. But in case that's too much busywork, there's a handy octet-to-decimal conversion chart at the end of this chapter (Table 2-3) for your reference.

IP ADDRESSING

So let's use your newly honed binary arithmetic skills (or the chart at the end of the chapter!) to look at IP addressing notation. We break our 32-bit IP address into four 8-bit octets, then we convert each octet to decimal and separate each with decimal points or "dots." This is certainly easier than calculating these 32 bits as

*I loved my old textbooks, which always left that statement to derive the most crucial equations! This one's not so crucial since we've provided the cheat sheet in Table 2-3 in the back of this chapter to check your answers.

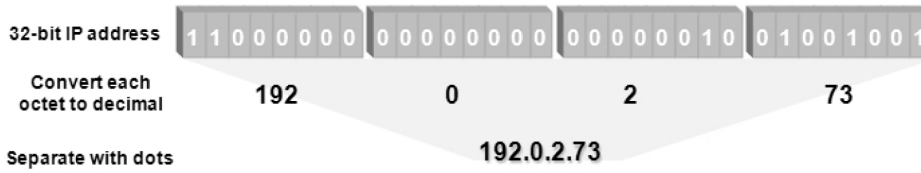

Figure 2-3: Binary to dotted decimal conversion

one huge number! Consider the 32-bit IP address in Figure 2-3. We break this into four octets, convert each octet to decimal, then separate the decimal representation of each octet by "dots." This notation is referred to as "dotted decimal."

Class-Based Addressing

As we introduced in Chapter 1, each IP address consists of a network portion and a host portion. The Internet Protocol specification RFC 791 defined three classes of addresses to define the relative size of these portions per class. These classes, denoted simply as classes A, B, and C, were identified by the initial bits of the 32-bit address as depicted in Figure 2-4. Each class corresponded to a particular fixed size for the network address and local address fields. The local address field could be assigned to individual hosts or further broken down into subnet and host fields as we'll discuss later.

The division of address space into classes provided a means to easily define different-sized networks for different users' needs. At the time, the Internet was comprised of certain U.S. government agencies, universities, and some research institutions. It had not yet blossomed into the de facto world-wide backbone network it is today, so address capacity was seemingly limitless. The other reason to divide address space into classes on these octet boundaries was for easier implementation of network routing. Routers could identify the length of the network portion of the address, also known as the *network prefix* simply by examining the first few bits of the destination address. They would then simply look up the network in their routing table and route each packet accordingly. Computational horsepower in those days was rather limited, so minimizing processing requirements was an important consideration. A side benefit of classful addressing was simple readability. Each dotted decimal number represents one octet in binary. As we'll see later when discussing classless addressing, this is not typically the case today.

Looking at this classful breakdown, we can observe a few key points:

- Class A networks
  - Class A prefixes begin with binary 0 ($[0]_2$) plus 7 additional bits* or 8 network bits total.

*To differentiate a binary 0 (one bit) from a decimal 0 (7–8 bits) in cases where it may be ambiguous, we subscript the number with the appropriate base. Don't worry; we're not digressing into chemistry with discussion of oxygen molecules with the 0_2 notation, simply "zero base 2."

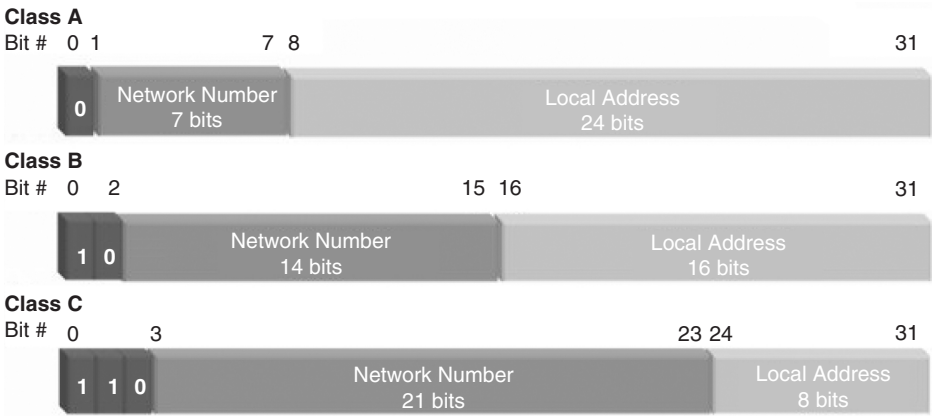

Figure 2-4: Class-based IP addressing

- The network address of all 0's is invalid, though some protocols like DHCP use the all 0's address as a placeholder for "this" address.
- The network address of $[01111111]_2 = 127$ is a reserved address. Address 127.0.0.1 is used for the "loopback address" on an interface.
- This leaves us with a class A network prefix range of $[00000001]_2$ to $[01111110]_2 = 1\text{-}126$ as the first octet.
- The local address field is 24 bits long. This equates to up to $2^{24} = 16{,}777{,}216$ possible local addresses per network address. Generally, the all 0's local address represents the "network" address and the all 1's is a network broadcast, so we typically subtract these two addresses from our local address capacity in general to arrive at 16,777,214 hosts per Class A network. Thus, 10.0.0.0 is the network address, and 10.255.255.255 is the broadcast address for all hosts on the 10.0.0.0/8 network, where "/8" denotes an 8-bit network prefix.

• Class B networks
  - Class B networks begin with $[10]_2$ plus 14 additional bits or 16 network bits total.
  - The range of class B network prefixes in binary is $[10000000\ 00000000]_2$ to $[10111111\ 11111111]_2$ or networks in the range of 128.0.0.0 to 191.255.0.0, yielding 16,384 network addresses.
  - The local address field is 16 bits long for $65{,}536 - 2 = 65{,}534$ possible hosts per class B network.

• Class C networks
  - Class C networks begin with $[110]_2$ plus 21 additional bits or 24 network bits total.
  - The range of class C network prefixes is $[11000000\ 00000000\ 00000000]_2$ to $[11011111\ 11111111\ 11111111]_2$ or networks in the range 192.0.0.0 to 223.255.255.0, yielding 2,097,152 networks.

- The local address field is 8 bits long for 256 – 2 = 254 possible hosts per Class C network.

- Class D networks (not illustrated in Figure 2-4)
  - Class D networks were defined after RFC 791 and begin with $[1110]_2$ to denote multicast addresses. In other words, multiple hosts having a common multicast address would receive all IP traffic sent to the multicast address or group. There is no network and host portion of the multicast network as members of a multicast group may reside on many different physical networks.
  - The range of class D networks is from $[11100000\ 00000000\ 00000000\ 00000000]_2$ to $[11101111\ 11111111\ 11111111\ 11111111]_2$ or the 224.0.0.0 to 239.255.255.255 range, yielding 268,435,456 multicast addresses.
- Class E networks (not illustrated in Figure 2-4)
  - Networks beginning with $[1111]_2$ (Class E) are reserved.

Internet Growing Pains

With seemingly limitless IP address capacity, at least as it seemed through the 1980s, class A and B networks were generally allocated to whomever asked. Network Information Centers (NICs), the organizations responsible for allocating IP address space at the time, had relatively loose application requirements. Recipient organizations would then subdivide or subnet* their class A or B networks along octet boundaries within their organizations.

As email and web browsing burst into the mainstream, more and more companies sought to participate in the Internet by requesting IP address space, and NICs were forced to throttle address allocations. Those requesting IP address space from NICs soon faced increasingly stringent application requirements and were granted a fraction of the address space requested. In having to make do with smaller network block allocations, many organizations were forced to subnet on non-octet boundaries.

Whether on octet boundaries or not, subnetting is facilitated by specifying a *network mask* along with the network address. The network mask is an integer number representing the length in bits of the network prefix. This is sometimes also referred to as the mask length. For example, a class A network has a mask length of 8, a class B of 16, and class C of 24. By essentially extending the length of the network number that routers need to examine in each packet, a larger number of networks can be supported, and address space can be allocated more efficiently. This is illustrated in Figure 2-5.

Routers need to be configured with this mask length for each subnet that they serve. This allows them to "mask" the IP address, to expose only the indicated network+subnet bits within the 32-bit IP address to enable efficient routing

*The term *subnet* is frequently used as a verb as in this context, to mean the act of creating a subnet.

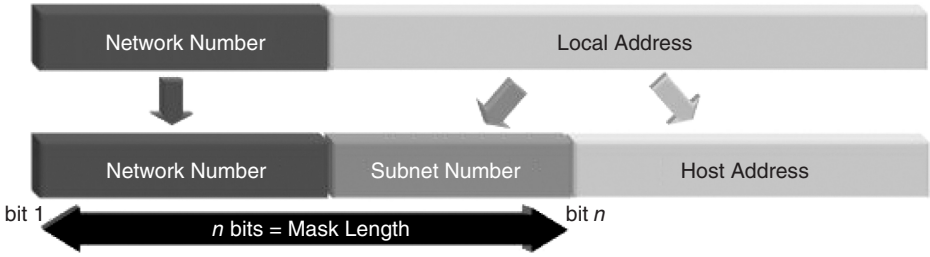

Figure 2-5: Subnetting provides more "networks" with fewer hosts per network

without relying on address class. Based on this extended network number, the router can route the packet accordingly.

The network address and mask length were originally denoted by specifying the 32-bit mask in dotted decimal notation. This notation was derived by denoting the first *n* bits of a 32-bit string as 1's and the remaining 32-*n* bits as 0's, then converting this to dotted decimal.

For example, to denote a network mask length of 19 bits, you would

- Create the 32-bit number with 19 1's and 13 0's: 11111111111111111000 0000000000
- Separate into octets: 11111111.11111111.11100000.00000000
- Convert to dotted decimal: 255.255.224.0.

The final notation for network 172.16.0.0 with this 19-bit mask is 172.16.0.0/255.255.224.0.

Thankfully, this approach was superseded by a simpler notation: the mask is now denoted with the network address as: <network address> / <mask length>. While the notation is easier to read, it does not save us from the equivalent binary exercise! For example, the 172.16.0.0 class B network would be represented as 172.16.0.0/16. The "slash 16" indicates that the first 16 bits, in this case the first two octets, represent the network prefix.

Here's the binary representation of this network:

Network Address	***Network Prefix***	Local Address
172.16.0.0/16	***10101100 00010000***	00000000 00000000

Following our prior example, let's subnet this network further using a 19-bit mask. Expanding this out into binary notation:

Network Address	***Network Prefix***	***Subnet*** Local Address
172.16.0.0/19	***10101100 00010000***	***000***00000 00000000
172.16.32.0/19	***10101100 00010000***	***001***00000 00000000
172.16.64.0/19	***10101100 00010000***	***010***00000 00000000
172.16.96.0/19	***10101100 00010000***	***011***00000 00000000

Network Address	***Network Prefix***	***Subnet*** Local Address
172.16.128.0/19	***10101100 00010000***	***100***00000 00000000
172.16.160.0/19	***10101100 00010000***	***101***00000 00000000
172.16.192.0/19	***10101100 00010000***	***110***00000 00000000
172.16.224.0/19	***10101100 00010000***	***111***00000 00000000

There—we've created eight subnetworks from one network. Notice that the class B network bits are depicted under the Network Prefix column in italic font, and we highlighted the subnet bits in larger bold italic font in the Subnet column. Using this 3-bit subnet mask, we effectively extended the network prefix from 16 bits to 19. By incrementing the binary values of these 3 bits from $[000]_2$ to $[111]_2$ per the highlighted Subnet bits above, we can derive $2^3 = 8$ subnets with this 3-bit subnet mask extension. Routers would then be configured to route using the first 19 bits to identify the network portion of the address by configuring the router serving such a subnet with the corresponding mask length, for example, 172.16.128.0/19, then having the router communicate reachability to this network via routing protocols. This technique, called variable length subnet masking (VLSM), became increasingly more prevalent in helping to squeeze as much IP address capacity as possible out of the address space assigned within an organization.

The two-layer network/subnet model worked well during the first decades of IP's existence. However, in the early 1990s, demand for IP addresses continued to explode, with more and more companies desiring IP address space to publish websites. At the then current rate of usage, the address space was expected to be exhausted before the turn of the century! The guiding body of the Internet, the Internet Engineering Task Force (IETF), cleverly implemented two key policies to extend the usable life of the IP address space, namely, support of private address space (ultimately RFC 1918 (5)) and classless inter-domain routing (CIDR, RFCs 1517-1519 (6)(7)(8)). The IETF also began work on a new version of IP with tremendously enormous address space during this time, IP version 6, which we'll discuss later in this chapter.

Private Address Space

Recall our statement that every "network" within an organization, needs to have a unique network number or prefix to maintain address uniqueness and maintain route integrity. As more and more organizations connected to the Internet, the Internet became a potential vehicle for hackers to infiltrate organizations' networks. Many organizations implemented firewalls to filter out IP packets based on specified criteria regarding IP header values, such as source or destination addresses, UDP vs. TCP, and others. This guarded partitioning of IP address space between "internal" and "external" address space dove-tailed nicely with address conservation efforts within the IETF.

The IETF issued a couple of RFC revisions, resulting in RFC 1918 becoming the standard document that defined the following sets of networks as "private":

- 10.0.0.0–10.255.255.255 (10/8 network)—equivalent to 1 class A
- 172.16.0.0–172.31.255.255 (172.16/12 network)—equivalent to 16 class B's
- 192.168.0.0–192.168.255.255 (192.168/16 network)—equivalent to 1 class B or 256 class C's

The term *private* means that these addresses are not routable on the Internet. However, within an organization, they may be used to route IP traffic on internal networks. Thus, my laptop is assigned a private IP address and I can send emails to my fellow associates, who also have private addresses. My organization in essence has defined a private Internet, sometimes referred to as an intranet. Routers within my organization are configured to route among allocated private IP networks, and the IP traffic among these networks never traverses the Internet. Technically, with the use of virtual private networks (VPNs) or tunnels over the Internet, privately addressed traffic may traverse the Internet, but the tunnel endpoints accessing the Internet on both ends do utilize public IP addresses.

Since I'm using a private IP address, someone external to the organization, outside the firewall, cannot reach me directly. Anyone externally sending packets with my private address as the destination address in the IP header will not be able to reach me as these packets will not be routed by Internet and ISP routers. But what if I wanted to initiate a connection externally to check on how much money I'm losing in the stock market via the Internet? For employees requiring access to the Internet, firewalls employing network address translation (NAT) functionality are commonly employed to convert an enterprise user's private IP address into a public or routable IP address from the corporation's public address space.

Typical NAT devices provide address pooling features to pool a relatively small number of publicly routable (non-private) IP addresses for use on a dynamic basis by a larger number of employees who sporadically access the Internet. The NAT device bridges two IP connections together: the internal-to-NAT device communications utilize private address space, while the NAT device-to-the

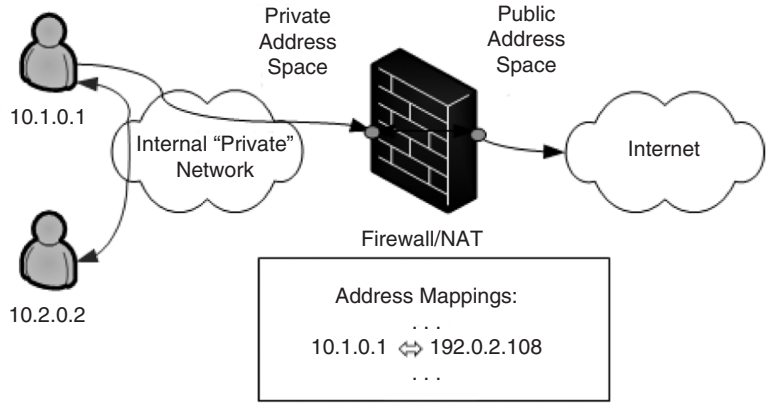

Figure 2-6: Example use of NAT to map private to public addresses

Internet communications utilizes public IP addresses. The NAT device is responsible for keeping track of mapping the internal employee address to the public address used externally.

This is illustrated in Figure 2-6, with the internal network utilizing the 10.0.0.0/8 address space, and external or public addressing utilizing the 192.0.2.0/24 space. Per the figure, if my laptop has the IP address 10.1.0.1, I can communicate to my colleague on IP address 10.2.0.2 via the internal IP network. When I access the Internet, my packets need to be routed via the firewall/NAT device in order to map my private 10.1.0.1 address to a public address, for example, 192.0.2.108. The mapping state is maintained in the NAT device and it modifies the IP header to swap out 10.1.0.1 for 192.0.2.108 for outbound packets and the converse for inbound packets.

From an addressing capacity requirements perspective, my organization only needs sufficient IP address space to support these ad hoc internal-to-Internet connections as well as Internet-reachable hosts such as web or email servers. This amount is generally much smaller than requiring IP address space for every internal and external router, server, and host. Implementation of private address space greatly reduced the pressure on address space, as enterprises required far less public address space.

CLASSLESS ADDRESSING

Implementation of classless inter-domain routing (CIDR) vastly improved network allocation efficiencies. Like variable length subnet masking, which allows subnetting of a classful network on non-octet boundaries, CIDR allows the network prefix for the base address block to be variable. Hence, a contiguous group of four class C's (/24), for example, could be combined and allocated to a service provider as a single /22. This is illustrated in the example below. If the four contiguous blocks shown, 172.16.168.0/24–172.16.171.0/24, are available for allocation, they could be allocated as a single /22, that is, 172.16.168.0/22 (Figure 2-7).

Notice that the darker shaded bits represent the network number, that is, the first 22 bits. These bits are identical on all four networks. The remaining 10

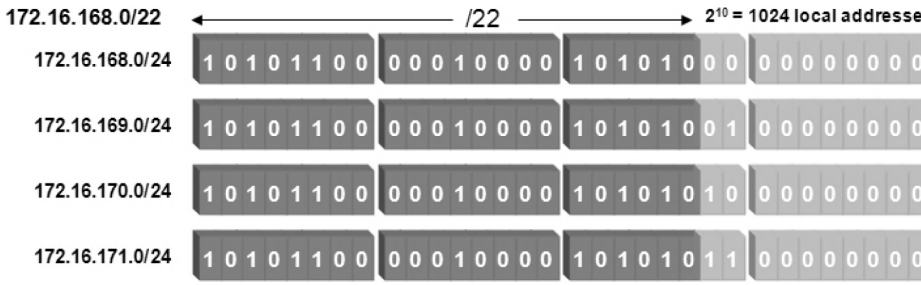

Figure 2-7: CIDR allocation example

bits represent the local address space for host assignment. Since the network address is indicated with all 0's in the local address field, this /22 network is identified as the bit string at the top, namely 172.16.168.0/22. As you can see, CIDR is very similar to VLSM in terms of the decimal to binary arithmetic required to calculate network addresses on non-octet boundaries. The extra step of filling in zeroes for local addresses outside non-octet boundary masks creates an additional opportunity for error. In addition, VLSM can be applied to a CIDR allocation to further increase the chance of error. But this is the price paid for more flexibility. CIDR and VLSM tore down the class walls to provide truly flexible network allocations and subnetting.

SPECIAL USE IPv4 ADDRESSES

In addition to private space, IPv4 address allocations include certain reservations for special use IP addresses, which are summarized in Table 2-1 below and defined in RFC 3330 (9) and the RFC 3330*bis* internet draft (10).

TABLE 2-1: IPv4 Address Allocations (9–10)

Address Space	Special Use
0.0.0.0/8	"This" network; 0.0.0.0/32 denotes this host on this network
10.0.0.0/8	Private IP address space, not routable on the public Internet per RFC 1918
127.0.0.0/8	Assigned for use as the Internet host loopback address, i.e., 127.0.0.1/32
169.254.0.0/16	The "link local" block used for IPv4 auto-configuration for communications on a single link
172.16.0.0/12	Private IP address space, not routable on the public Internet per RFC 1918
192.0.0.0/24	Reserved for IETF protocol assignments
192.0.2.0/24	Assigned as "Test-Net-1" for use in documentation and sample code
192.88.99.0/24	Allocated for 6to4 relay anycast addresses (6to4 is an IPv4-IPv6 co-existence technology discussed in Chapter 8)
192.168.0.0/16	Private IP address space, not routable on the public Internet per RFC 1918
198.18.0.0/15	Allocated for use in benchmark tests of network interconnect devices
198.51.100.9/24	Assigned as "Test-Net-2" for use in documentation and sample code
203.0.113.0/24	Assigned as "Test-Net-3" for use in documentation and sample code
224.0.0.0/4	Allocated for IPv4 multicast address assignments (formerly Class D space)
240.0.0.0/4	Reserved for future use (formerly Class E space)
255.255.255.255/32	Limited broadcast on a link

IP VERSION 6 (IPv6)

Version 6 of the Internet Protocol* is an evolution from version 4 but is not inherently compatible with version 4. The primary objective for version 6 was essentially to redesign version 4 based on the prior 20 years of experience with IPv4. Real world application support added to the IPv4 protocol suite over the years was designed into IPv6 from the outset. This included support for security, multicast, mobility, and auto-configuration.

The most striking difference in the evolution from IPv4 to IPv6 is the tremendous expansion of the size of the IP address field. Whereas IPv4 uses a 32-bit IP address field, IPv6 uses 128 bits. A 32-bit address field provides a theoretical maximum of 2^{32} addresses or 4.2 billion addresses. A 128-bit address field allows 2^{128} addresses or 340 trillion trillion trillion addresses or 340 undecillion (3.4×10^{38}, using the American definition of undecillion, 10^{36}, not the British definition which is 10^{66}). To put some context around this tremendously large number, consider that this quantity of IP addresses:

- Averages to 5×10^{28} IP addresses per person on Earth based on a 6.5 billion population.
- Averages to 4.3×10^{20} IP addresses per square inch of the Earth's surface.
- Amounts to about 14 million IP addresses per *nanometer* to the nearest galaxy, Andromeda, at 2.5 million light years.

Like IPv4, not nearly every address will be usable due to subnetting inefficiencies, but a few undecillion of wasted addresses won't have much impact! Beyond this seemingly incomprehensible number of IP addresses, there are a number of similarities between IPv6 and IPv4. For example, at a basic level, the "IP packet" concept we discussed in Chapter 1 applies equally well for IPv6 as IPv4 in terms of the concept of the packet header and contents, as does the basic concept of protocol layering, packet routing, and CIDR allocations.

IPv6 Addressing

Three types of IPv6 addresses have been defined. Like IPv4, these addresses apply to interfaces, not nodes. Thus, a printer with two interfaces would be addressed by either of its interfaces. The printer can be reached on either interface, but the printer node does not have an IP address per se. Many router and server products support the concept of a "box address" via a software loopback address. This loopback address, not to be confused with the 127.0.0.1 or ::1 loopback addresses, enables reachability to any one of the device's interfaces. Of

*IP version 5 was never implemented as an official version of IP. The version number of "5" in the IP header was assigned to denote packets carrying an experimental real-time stream protocol called ST, the Internet Stream Protocol. If you'd like to learn more about ST, please refer to RFC 1819 (121).

course, for end users attempting to access a node, DNS can hide this subtlety by enabling a host name to map to one or more interface addresses.

- **Unicast**—the IP address of a single interface. This is analogous to the common interpretation of an IPv4 host address (non-multicast/non-broadcast /32 IPv4 address).
- **Anycast**—an IP address for a set of interfaces usually belonging to different nodes, any one of which is the intended recipient. An IP packet destined for an anycast address is routed to the nearest interface (according to routing table metrics) configured with the anycast address. The concept is that the sender doesn't necessarily care which particular host or interface receives the packet, but that one of those sharing the anycast address receives it. Anycast addresses are assigned from the same address space from which unicast addresses have been allocated. Thus, one cannot differentiate a unicast address from an anycast address by sight.

  Anycast in IPv4 networks for DNS has proven successful in providing *closest routing to the intended service* by deploying a unicast IPv4 address on multiple DNS servers. This provides benefits in simplifying client configuration in always using the same [anycast] IP address to perform a DNS query, regardless of where on your network the client is connected.
- **Multicast**—an IP address for a set of interfaces typically belonging to different nodes, all of which are intended recipients. This of course is similar to IPv4 multicast. Unlike IPv4, IPv6 does not support broadcasts. Instead, applications that utilized broadcasts in IPv4, such as DHCP, use multicast to a well-known (i.e., pre-defined) DHCP multicast address in IPv6.

A device interface may have multiple IP addresses of any or all address types. IPv6 also defines a link-local scope of IP addresses to uniquely identify interfaces attached to a particular link, such as a LAN. Additional scoping can be administratively defined per site or per organization for example.

IPv6 Address Notation

Recall that IPv4 addresses are represented in dotted decimal format where the 32-bit address is divided into four 8-bit segments, each of which are converted to decimal, then separated with "dots." If you thought remembering a string of four decimals was difficult, IPv6 will make life a little tougher. IPv6 addresses are not expressed in dotted-decimal notation; they are represented using a colon-separated hexadecimal format. Jumping down to the bit level, the 128-bit IPv6 address is divided into eight 16-bit segments, each of which is converted to hexadecimal, then separated by colons. Each hexadecimal "digit" represents four bits per the mapping of each hex digit (0-F) to its 4-bit binary mapping below. Each hex digit corresponds to 4 bits with possible values of:

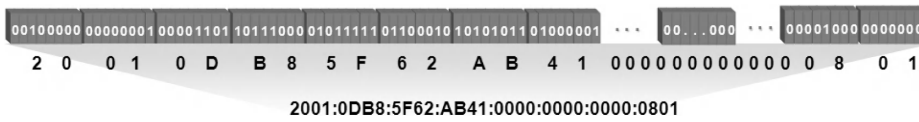

Figure 2-8: IPv6 address representation

0 = 0000	4 = 0100	8 = 1000	C = 1100
1 = 0001	5 = 0101	9 = 1001	D = 1101
2 = 0010	6 = 0110	A = 1010	E = 1110
3 = 0011	7 = 0111	B = 1011	F = 1111

After converting a 128-bit IPv6 address from binary into hex, we group sets of four hex digits and separate them with colons. We'll use the term *nibble* to represent a grouping of four hex digits or 16 bits; thus, we have eight nibble values separated by colons, rendering an IPv6 address as illustrated in Figure 2-8.

Instead of dealing with four decimal values, each between 0 and 255, separated by dots in IPv4, IPv6 addresses consist of up to eight hexadecimal values, each between 0 and FFFF, separated by colons. There are two acceptable abbreviations when writing IPv6 addresses. First, leading zeroes within a nibble section, that is, between colons, may be dropped. Thus, the address above could be abbreviated:

2001:DB8:5F62:AB41:0:0:0:801

The second form of abbreviation is the use of a double colon to represent one or more consecutive sets of zero nibbles. Using this form of abbreviation, the address above can be further abbreviated as:

2001:DB8:5F62:AB41::801

Isn't that much better?! Note that only one double colon may be used within an address representation. Since there are always eight nibble segments in the address, one can easily calculate how many of them are zero with one double-colon notation; however it would be ambiguous with more than one.

Consider the address: 2001:DB8:0:56FA:0:0:0:B5. We can abbreviate this address as either:

2001:DB8::56FA:0:0:0:B5 *or* 2001:DB8:0:56FA::B5

We can easily calculate that the double colon denotes one nibble segment (8 total minus 7 nibble segments shown) in the first case and three (8 minus 5 shown) in the second notation. If we attempted to abbreviate this address as 2001:DB8::56FA::B5, we could not unambiguously decode this, as it could represent any of the following possible addresses:

2001:DB8:0:56FA:0:0:0:B5

2001:DB8:0:0:56FA:0:0:B5

2001:DB8:0:0:0:56FA:0:B5

Thus the requirement holds that only one double colon may appear in an IPv6 address.

Address Structure

As with IPv4, the IPv6 address is composed of a network and a host portion; however, experience with VLSM and subnetting in general has led to addition of a subnet field. The IPv6 address is divided into three fields, shown in Figure 2-9.

The global routing prefix is akin to an IPv4 network number and is used by routers to forward packets to router(s) locally serving the network corresponding to the prefix. For example, a customer of an ISP may be assigned a /48-sized global routing prefix and all packets destined to this customer would contain the corresponding global routing prefix value. In this case, n = 48 per Figure 2-9. When denoting a network, the global routing prefix is written, followed by slash, then the network size, called the prefix length. Assuming our example IPv6 address, 2001:DB8:5F62:AB41::801, resides within a /48 global routing prefix, this prefix address would be denoted as 2001:DB8:5F62::/48. As with IPv4, the network address is denoted with zero-valued bits beyond the prefix length (bits 49-128 in this case) as denoted by the terminating double colon.

The subnet ID provides a means to denote particular subnets within the organization. Our ISP customer with a /48 may choose to use 16 bits for the subnet ID, providing 2^{16} or 65,534 subnets. In this case m = 16 per Figure 2-9. This leaves 128 – 48 – 16 = 64 bits for the interface ID. The Interface ID denotes the interface address of the source or intended recipient for the packet. As we'll discuss a bit later, the global unicast address space that has been allocated for use so far requires a 64-bit interface ID field.

One of the unique aspects of this IPv6 address structure in splitting a network ID consisting of the global routing prefix and subnet ID, from an Interface ID, is that a device can retain the same Interface ID independently of the network to which it is connected, effectively separating "who you are," your interface ID, from "where you are," your network prefix. As we'll see, this convention facilitates address autoconfiguration, though not without privacy concerns. But we're getting a little ahead of ourselves, so let's jump back up to the macro level and

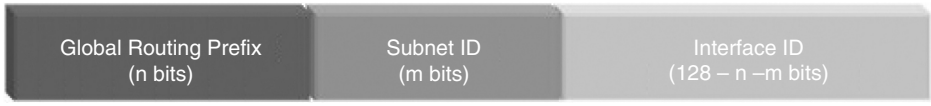

Figure 2-9: IPv6 address structure (11)

consider the IPv6 address space allocated so far by the Internet addressing authority, the Internet Assigned Numbers Authority (IANA).

IPv6 ADDRESS ALLOCATIONS

The address space that has been allocated so far by IANA is highlighted in bold in Table 2-2 and is discussed in the ensuing text. These allocations represent less than 14% of the total available IPv6 address space.

2000::/3—Global Unicast Address Space

The global unicast address space allocated so far, 2000::/3, represents 2^{125} or 4.25×10^{37} IP addresses. Given the 64-bit Interface ID requirement defined in the IPv6 addressing architecture (RFC 4291 (11)), the global unicast address format as formally defined in RFC 3587 (13) is depicted in Figure 2-10.

TABLE 2-2: IPv6 address allocations (12)

IPv6 Prefix	Binary Form	Relative Size of IPv6 Space	Allocation
0000::/3	000	1 / 8	Reserved by IETF—The "unspecified address" (::) and the loopback address (::1) are assigned from this block.
2000::/3	**001**	**1 / 8**	**Global unicast address space**
4000::/3	010	1 / 8	Reserved by IETF
6000::/3	011	1 / 8	Reserved by IETF
8000::/3	100	1 / 8	Reserved by IETF
A000::/3	101	1 / 8	Reserved by IETF
C000::/3	110	1 / 8	Reserved by IETF
E000::/4	1110	1 / 16	Reserved by IETF
F000::/5	1111 0	1 / 32	Reserved by IETF
F800::/6	1111 10	1 / 64	Reserved by IETF
FC00::/7	**1111 110**	**1 / 128**	**Unique Local Unicast**
FE00::/9	1111 1110 0	1 / 512	Reserved by IETF
FE80::/10	**1111 1110 01**	**1 / 1024**	**Link Local Unicast**
FEC0::/10	1111 1110 11	1 / 1024	Reserved by IETF
FF00::/8	**1111 1111**	**1 / 256**	**Multicast**

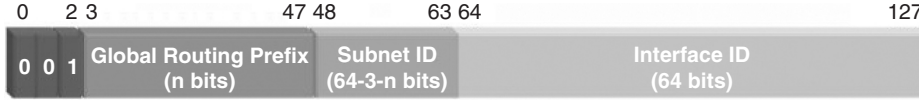

Figure 2-10: Global unicast address format

The first three bits are $[001]_2$ to indicate global unicast address space. The following 61 bits comprise the global routing prefix, and subnet ID followed by the 64-bit Interface ID.

FC00::/7—Unique Local Address Space

The unique local address (ULA) space, defined in RFC 4193 (14), is intended to provide locally assignable and routable IP addresses, usually within a site. RFC 4193 states that "these addresses are not expected to be routable on the global Internet." Thus, while not as stringent as RFC 1918 in defining private IPv4 address space, the unique local address space is essentially private addressing, providing "local" addressing with a high probability of still being globally unique. The format of unique local address space is shown in Figure 2-11.

The first seven bits, bits 0-6, are $[1111\ 110]_2$ = FC00::/7, which identifies a unique local address. The eighth bit, the "L" bit is set to "1" if the Global ID is locally assigned; setting the "L" bit to "0" is currently undefined. The 40-bit Global ID field is intended to represent a globally unique prefix and must be allocated using a pseudo-random algorithm, not sequentially. The subnet ID is a 16-bit field to identify each subnet, while the interface ID is a 64-bit field.

FE80::/10—Link Local Address Space

Link local addresses are used only on a particular link, such as an Ethernet link; packets with link local destination addresses are not routed. That is, packets having link local addresses will not be routed and will therefore not reach beyond the corresponding link. The format of link local addresses is shown in Figure 2-12.

The FE80::/10 link local prefix is followed by 54 zero bits and the 64-bit interface ID (11).

FF00::/8—Multicast Address Space

Multicast addresses identify a group of interfaces typically on different nodes. Think of multicast addresses as a scoped broadcast. All multicast group members

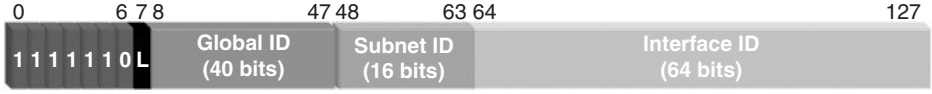

Figure 2-11: Unique local address format

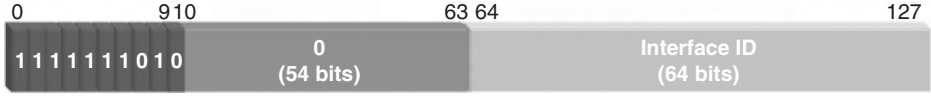

Figure 2-12: Link local address format

Figure 2-13: Multicast address format

share the same group ID and hence all members will accept packets destined for the multicast group. An interface may have multiple multicast addresses; that is, it may belong to multiple multicast groups. The basic format of IPv6 multicast addresses is illustrated in Figure 2-13.

The prefix FF00::/8 identifies a multicast address. The next field is a 4-bit field called "Flags." The format of the multicast packet is dependent on the value of the flags. The Scope field indicates the breadth of the multicast scope, whether per node, link, global, or other scope values. Some example multicast addresses follow:

- FF01::2 = node-local all routers address
- FF02::1 = link-local all nodes address
- FF05::1:3 = site-local all DHCP servers address
- FF08::101 = organization-local NTP server address

IPv6 ADDRESS AUTOCONFIGURATION

Leading among the advertised benefits of IPv6 is the ability for devices to automatically configure their own IPv6 address that will be unique and relevant to the subnet to which it is presently connecting.* After all, who wants to type in a massive hex string! In a nutshell, an IPv6 address is autoconfigured by concatenating the network prefix of the subnet to which the device is connected with the node address, derived from its corresponding network interface (MAC) address.

The first address autoconfiguration ingredient, the network prefix, is derived through a process called *neighbor discovery*. Neighbor discovery enables IPv6 nodes to discover the presence of other IPv6 nodes, to identify their link layer addresses, to discover routers serving the subnet, and to perform duplicate address detection.

In the specific case where the neighbor is a router, the router advertises the network prefix (i.e., subnet address) served on the link including corresponding prefix length and valid address lifetime, as well as other configuration parameters. The router advertisement also indicates whether a DHCPv6 server is available for address assignment or other configuration. Thus, a device may truly

*Note that some IPv4 protocol stacks, such as those provided with Microsoft Windows 2000 and XP, among others, perform address autoconfiguration utilizing the IPv4 "link local" address space, 169.254.0.0/16.

autoconfigure its address (second ingredient coming up in a moment!), use DHCPv6 or both.

Technically there are three basic forms of IPv6 address autoconfiguration:

- Stateless—This process is "stateless" in that it is not dependent on the state or availability of external assignment mechanisms, for example, DHCPv6. The device attempts to configure its own IPv6 address(es) without external or user intervention.
- Stateful—The stateful process relies solely on an external address assignment mechanism such as DHCPv6. The DHCPv6 server assigns the 128-bit IPv6 address to the device in a manner similar to DHCP for IPv4 operation. This process will be described in Chapter 3.
- Combination Stateless and Stateful—This process involves a form of stateless address autoconfiguration used in conjunction with stateful configuration of additional IP parameters. This commonly entails a device autoconfiguring an IPv6 address using the stateless method, then utilizing DHCPv6 to obtain additional parameters or options such as which Network Time Protocol (NTP) servers to query for time resolution on the given network.

Interface IDs

The second half of an autoconfigured (stateless or combination) address consists of an interface identifier. The IPv6 addressing architecture stipulates that all unicast IPv6 addresses, other than those beginning with binary $[000]_2$, must use a 64-bit Interface ID derived using the modified EUI-64 algorithm. The "unmodified" EUI-64 algorithm entails concatenating the 24-bit company identifier issued by the IEEE to each network interface hardware manufacturer (e.g., the initial 24 bits of an Ethernet address) with a 40-bit extension identifier. For 48-bit Ethernet addresses, the company identifier portion of the Ethernet address (first 24 bits) is followed by a 16-bit EUI label (defined as hexadecimal FFFE) followed by the 24-bit extension identifier, that is, the remaining 24 bits of the Ethernet address.

The modification required to convert an unmodified into a modified EUI-64 identifier calls for inverting the "u" bit (universal/local bit) of the company identifier field. The "u" bit is the seventh most significant bit in the company identifier field. Thus the algorithm for a 48-bit MAC address is to invert the "u" bit and insert the hexadecimal value FFFE between the company identifier and the interface identifier. This is illustrated in Figure 2-14 using a MAC address of AC-62-E8-49-5F-62. The resulting interface ID is AE62:E8FF:FE49:5F62. This interface ID can then be appended to network prefixes on the corresponding network interface. Hence, given the link-local prefix of FE80::/10, the link-local address for this interface is FE80::AE62:E8FF:FE49:5F62.

For non-Ethernet MAC addresses, the algorithm calls for use of the link layer address as the Interface ID, with zero padding (from the "left"). For cases

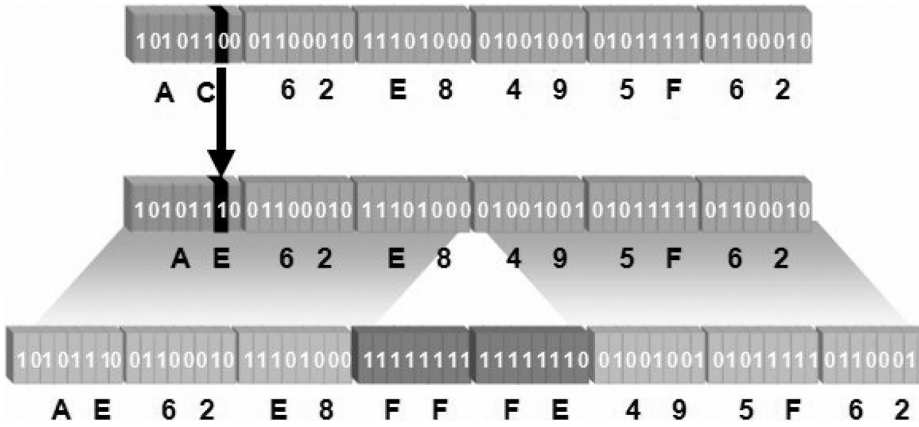

Figure 2-14: Modified EUI-64 derivation

where no link layer address is available, for example, on a dial-up link, a unique identifier utilizing another interface address, a serial number, or other device-specific identifier is recommended.

However, due to the fact that interface IDs may not be unique, especially if not derived from a unique 48-bit MAC address, the device must perform duplicate address detection (DAD) prior to committing and using the new address. Hence, prior to completing the DAD process, the address is considered tentative.

Duplication Address Detection (DAD)

DAD is performed using the neighbor discovery process we mentioned earlier and enables the identification of another host on the subnet which is already using the desired IPv6 address. The basic process entails the device sending an IPv6 Neighbor Solicitation packet to its desired IPv6 address in order to identify a pre-existing occupant of the IP address. After a slight delay, the device also sends a Neighbor Solicitation packet to the *solicited node multicast address* associated with the address. The solicited node multicast address is formed by appending the low-order (rightmost) 24 bits of the solicited node's Interface ID to the well-known FF02::1:FF00:0/104 prefix.

For example, let's say a node wishes to use the IP address: 2001:DB8:4E:2A:3001:FA81:95D0:2CD1. Using the low-order 24 bits, D02CD1 in hex, the device would address its neighbor solicitation request to FF02::1:FFD0:2CD1 as illustrated in Figure 2-15.

If another device is already using the IP address or if the low-order bits of its interface ID match, it will respond with a Neighbor Advertisement packet, and the autoconfiguration process will cease; that is, manual intervention or configuration of the device to use an alternate interface ID is required. If a Neighbor Advertisement packet is not received, the device can assume uniqueness of the address and assign it to the corresponding interface. Participation in

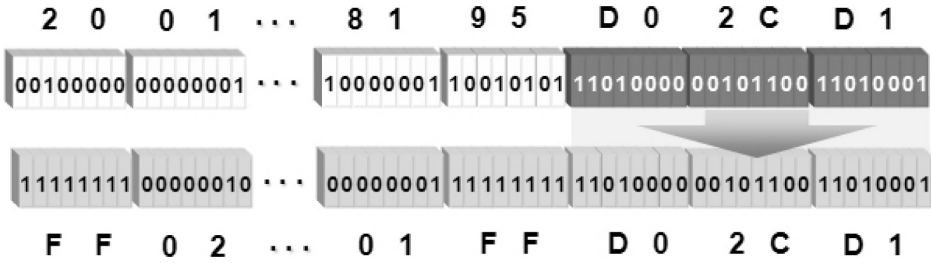

Figure 2-15: Solicited node Multicast address (11)

this process of Neighbor Solicitation and Advertisement is required not only for autoconfigured addresses but even for those statically defined or obtained through DHCPv6.

After the link local address has been assigned, the device attempts to configure a global address. Periodic Router Advertisement messages from router(s) on the subnet advertise network prefixes and associated configuration parameters for locally connected devices. A device may issue a Router Solicitation message to explicitly request a Router Advertisement message. These Router Advertisement messages are used by the device to identify the global unicast network prefix configured for this subnet. The device appends its Interface ID to the network prefix and issues a Neighbor Solicitation packet to this tentative address to initiate the DAD process. If no Neighbor Advertisement reply is received, the device can assign the address to its corresponding interface.

IPv6 Address Lifetimes

IPv6 addresses have a lifetime during which they are valid. In some cases the lifetime is infinite, but the concept of address lifetime applies to both DHCPv6 leased addresses as well as autoconfigured addresses. This is useful to ease the process of network renumbering, should it become necessary. Routers are configured with and advertise a preferred lifetime and a valid lifetime value for each network prefix in their Router Advertisement messages. Addresses that have successfully proven unique through the duplicate address detection process described above can be considered either preferred or deprecated. In either state, the address is valid, but this provides a means for upper layer protocols (e.g., TCP, UDP) to select an IP address that will likely not change during the session. The values of preferred lifetime and valid lifetime are included with each Router Advertisement message.

A device refreshes the preferred and valid time with each Router Advertisement message in accordance with the values advertised. When time expires on a preferred prefix, the associated address(es) will become deprecated, though still valid. Thus the deprecated state provides a transition period during which the address is still functional but should not be used to initiate new communications. Once the valid lifetime of the address expires, the address is no longer valid for use. Should a subnet be reassigned a different network prefix,

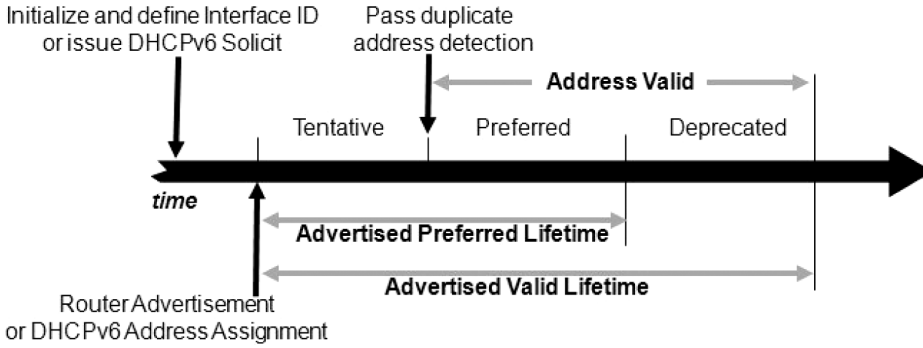

Figure 2-16: IPv6 address lifetimes (Figure based on (15))

the router will advertise the new prefix, and devices on the network would undergo the autoconfiguration process using the new prefix. The relationship among these address states is illustrated in Figure 2-16.

REQUIRED HOST IPv6 ADDRESSES

RFC 4294 (16) summarizes the requirements for IPv6 nodes, a device that implements IPv6, and for IPv6 routers. In terms of required addresses, all IPv6 nodes must be capable of recognizing the following IPv6 addresses for itself:

- The loopback address (::1)
- Its link-local unicast address (FE80::<interface ID> as configured via autoconfiguration)
- The all-nodes multicast address (FF0*s*::1 where *s* = scope)
- Unicast and anycast addresses configured automatically or manually on each interface
- The solicited node multicast address for each of its unicast and anycast addresses
- Multicast addresses for each multicast group to which the node belongs

A router node is required to support the above addresses plus the following addresses:

- The subnet router-anycast address (<subnetwork prefix>::/128, that is interface ID = 0)
- The all-routers multicast address (FF0*s*::2 where *s* = scope)
- Anycast addresses configured on the router

Other device types such as DHCP and DNS servers must recognize scoped multicast addresses corresponding to Group IDs assigned by IANA (i.e., when Flags = 0).

REGIONAL INTERNET REGISTRIES

IP addresses must be unique on a given network for proper routing and communication. As with every seemingly authoritative statement, there are exceptions! Anycast addresses are typically assigned to multiple hosts, and multicast addresses likewise are shared. Private addresses are also commonly duplicated. This statement applies to unicast addresses.

How is this uniqueness assured across the global Internet? The Internet Assigned Numbers Authority, IANA, is responsible for global allocation of IP address space for both IPv4 and IPv6, as well as other parameters used within the TCP/IP protocol, such as application port numbers. In fact, you can view these top-level allocations by browsing to www.iana.org/ipaddresses/ip-addresses.htm and selecting "Internet Protocol v4 Address Space" or "IPv6 Address Space."

IANA is, in essence, the top-level address registry, and it allocates address space to Regional Internet Registries (RIRs). The RIRs, listed below, are organizations responsible for allocation of address space within their respective global regions from their corresponding space allotments from IANA.

- AfriNIC (African Network Information Centre)—Africa Region
- APNIC (Asia Pacific Network Information Centre)—Asia/Pacific Region
- ARIN (American Registry for Internet Numbers)—North America Region
- LACNIC (Regional Latin-American and Caribbean IP Address Registry)—Latin America and some Caribbean Islands
- RIPE NCC (Réseaux IP Européens)—Europe, the Middle East, and Central Asia

The goals of the RIR system are as follows:

- Uniqueness—Each IP address must be unique world-wide for global Internet routing.
- Aggregation—Hierarchical allocation of address space assures proper routing of IP traffic on the Internet. Without aggregation, routing tables become fragmented which could ultimately create tremendous bottlenecks within the Internet.
- Conservation—With IPv4 in particular but also for IPv6 space, address space needs to be distributed according to actual usage requirements.
- Registration—A publicly accessible registry of IP address assignments eliminates ambiguity and can help when troubleshooting. This registry is called the *whois* database. Today, there are many whois databases, operated not only by RIRs but by LIR/ISPs as well for their respective address spaces.
- Fairness—Unbiased address allocation based on true address needs and not long term "plans."

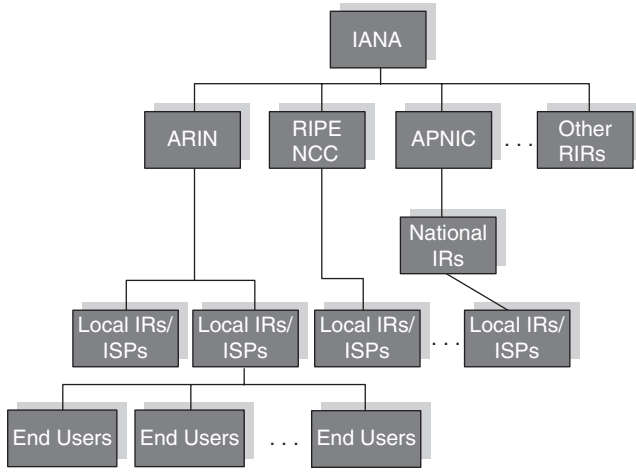

Figure 2-17: IP Address allocation from the top down (17)

The general address allocation hierarchy is depicted in Figure 2-17. National Internet Registries are akin to Local Internet Registries, but are organized at a national level.

Back in the 1980s and early 1990s many corporations (end users per Figure 2-17) obtained address space directly from NICs. However, the RIR/LIR/ISP layer was inserted during the transition to CIDR addressing to provide further delegation of address allocation responsibility. Today, most organizations obtain address space from LIRs or ISPs. The process for obtaining such address space is generally dictated by the LIR/ISP with whom you conduct business, though RIRs recommend use of consistent policies to maximize efficiency.

After an RIR allocates a given address space to an ISP, the ISP may advertise the address space on the Internet. The ISP may then assign portions of its given address space to its customers as they subscribe to the ISP's service. Customers in turn then allocate and assign their portion of the address space within their respective organizations. The ISP need not modify its address space advertisement because customer-assigned IP address space "rolls up" within the bounds of this advertised space. Going back to our postal analogy in Chapter 1, the allocation of space to an ISP is analogous to deployment of a new zip code to a postal region. Mail from outside this region destined to the new as well as former zip codes served by the region are forwarded to the corresponding distribution center. Likewise, as IP packets destined for an ISP's allocated space traverse the Internet, they are routed to the ISP for subsequent routing to the corresponding customer.

Thus, this insertion of the LIR/ISP layer helps aggregate route advertisements on the Internet. Multiple customers served by the ISP can be summarized in one route on the Internet. If business is good and the LIR/ISP requires more address space, the LIR/ISP can request additional space from their RIR. Each RIR generally has its own defined process for making address requests, so please consult the RIR in your region for details.

RIR Address Allocation

From an RIR perspective, RIRs *allocate* space to LIR/ISPs, and LIR/ISPs *assign* address space to their customers. The term *allocate* technically refers to the provision of an IP address block to serve as a "pool" of address space that can be drawn upon for *assignment* to customers. Customers can then use the assigned address space by themselves allocating blocks and subnets from this space, then assigning IP addresses from allocated subnets to individual hosts. The mechanics of this allocation and assignment are based on procedures we'll discuss in Chapter 6.

RIRs differentiate allocations from assignments because assignments comprise addresses in use, while allocations are pools for assignment which begin unused but in theory grow in usage with a number of assignments from the allocation over time. Technically, RIRs count both allocations and assignments as in-use, but leave open the ability to audit allocated space for actual address utilization as needed to process additional allocation requests from each LIR/ISP.

To obtain address space in the first place, the LIR/ISP must demonstrate the need for utilization of 25% of the allocation immediately and 50% within one year. Requests for additional address space require justification via demonstration of 80% utilization of the LIR/ISP's current allocations. In order to keep track of LIR/ISP allocations, the RIRs have each implemented electronic update mechanisms. As the LIR/ISP assigns address space, the assignment information can be communicated to the RIR using the corresponding form of electronic update. Theoretically, by the time additional address space is requested, the RIR and LIR/ISP have common allocation information against which the 80% threshold can be confirmed and approved. All RIRs allow emailing of information in specific template formats to convey allocation information. ARIN refers to this process as the Shared Whois Process or SWIP. Email templates vary by RIR and do occasionally change, so it's best to contact the RIR serving your geographic region to obtain the latest version of the template you require.

Address Allocation Efficiency

During the development of IPv6, much thought went into deriving the 128-bit address size. While IPv4 provides a 32-bit address field, which provides a theoretical maximum of 2^{32} addresses or over 4.2 billion addresses, in reality the theoretical maximum is much less than 4.2 billion. This is due to the hierarchical allocation of address space for multiple layers of networks, then subnets and finally hosts. RFC 1715 (18) provides an analysis of address assignment efficiency, in which a logarithmic scale was proposed as a measure of allocation efficiency, defined as the H ratio:

$$H = \frac{\log_{10}(number\ of\ objects)}{number\ of\ available\ bits}$$

With over 730 million hosts on the Internet today, today's H ratio is 0.277. The H ratio for 100% utilization of 4.2 billion IP addresses is 0.301 (which is equal to $\text{Log}_{10}(2)$, is the maximum value of the H ratio) so today's H ratio is high (92% of the maximum value).

Assignment efficiency measurements for IPv6, with its massive amount of address space, is not calculated based on the H ratio; a different ratio, the HD ratio (19), is used:

$$HD = \frac{\log_{10}(number\ of\ allocated\ objects)}{\log_{10}(maximum\ number\ of\ allocatable\ objects)}$$

The "objects" measured in the HD ratio for IPv6 are the IPv6 site addresses (/48s) assigned from an IPv6 prefix of a given size. The /48 address blocks are those expected to be assigned to each end user by the LIR/ISP. So an LIR/ISP with a /32 allocation which has allocated 100 /48s would have an HD ratio of log(100)/log(65,536) = 0.415.

KEY IP ADDRESSING MANAGEMENT CHALLENGES

Now that we've covered the basics of IPv4 and IPv6 addressing, let's summarize the key management challenges organizations face in effectively assigning and tracking IP address space. We'll discuss techniques for addressing these challenges beginning in Chapter 5.

- Obtain public IPv4 and IPv6 address space from an ISP or RIR.
- Derive private and ULA address space for internal use.
- Define the internal or customer hierarchy by geography, business unit, application, routing topology, or combination thereof and allocate top-level blocks of required types accordingly.
- Perform successive address space allocation at each hierarchy level as needed down to the subnet level where individual hosts will be assigned IP addresses.
- Define and track individual IP address assignments of each version and type, including devices with multiple addresses and interfaces. Some devices may have both IPv4 and IPv6 addresses. We'll discuss IPv4-IPv6 address co-existence in Chapter 8.
- Tracking address assignments and capacity to feed back to the allocation process if necessary to supplement IP address capacity.
- Keep the IP block, subnet, and address repository accurate with respect to actual assignments within your network. This is the IP address "plan of record" on which troubleshooting and future allocations and assignments will be made.

TABLE 2-3: Decimal to binary conversion

Decimal to Binary Conversion Chart							
Decimal	Octet	Decimal	Octet	Decimal	Octet	Decimal	Octet
0	00000000	43	00101011	86	01010110	129	10000001
1	00000001	44	00101100	87	01010111	130	10000010
2	00000010	45	00101101	88	01011000	131	10000011
3	00000011	46	00101110	89	01011001	132	10000100
4	00000100	47	00101111	90	01011010	133	10000101
5	00000101	48	00110000	91	01011011	134	10000110
6	00000110	49	00110001	92	01011100	135	10000111
7	00000111	50	00110010	93	01011101	136	10001000
8	00001000	51	00110011	94	01011110	137	10001001
9	00001001	52	00110100	95	01011111	138	10001010
10	00001010	53	00110101	96	01100000	139	10001011
11	00001011	54	00110110	97	01100001	140	10001100
12	00001100	55	00110111	98	01100010	141	10001101
13	00001101	56	00111000	99	01100011	142	10001110
14	00001110	57	00111001	100	01100100	143	10001111
15	00001111	58	00111010	101	01100101	144	10010000
16	00010000	59	00111011	102	01100110	145	10010001
17	00010001	60	00111100	103	01100111	146	10010010
18	00010010	61	00111101	104	01101000	147	10010011
19	00010011	62	00111110	105	01101001	148	10010100
20	00010100	63	00111111	106	01101010	149	10010101
21	00010101	64	01000000	107	01101011	150	10010110
22	00010110	65	01000001	108	01101100	151	10010111
23	00010111	66	01000010	109	01101101	152	10011000
24	00011000	67	01000011	110	01101110	153	10011001
25	00011001	68	01000100	111	01101111	154	10011010
26	00011010	69	01000101	112	01110000	155	10011011
27	00011011	70	01000110	113	01110001	156	10011100
28	00011100	71	01000111	114	01110010	157	10011101
29	00011101	72	01001000	115	01110011	158	10011110
30	00011110	73	01001001	116	01110100	159	10011111
31	00011111	74	01001010	117	01110101	160	10100000
32	00100000	75	01001011	118	01110110	161	10100001
33	00100001	76	01001100	119	01110111	162	10100010
34	00100010	77	01001101	120	01111000	163	10100011
35	00100011	78	01001110	121	01111001	164	10100100
36	00100100	79	01001111	122	01111010	165	10100101
37	00100101	80	01010000	123	01111011	166	10100110
38	00100110	81	01010001	124	01111100	167	10100111
39	00100111	82	01010010	125	01111101	168	10101000
40	00101000	83	01010011	126	01111110	169	10101001
41	00101001	84	01010100	127	01111111	170	10101010
42	00101010	85	01010101	128	10000000	171	10101011

TABLE 2-3: *Continued*

Decimal to Binary Conversion Chart							
Decimal	Octet	Decimal	Octet	Decimal	Octet	Decimal	Octet
172	10101100	193	11000001	214	11010110	235	11101011
173	10101101	194	11000010	215	11010111	236	11101100
174	10101110	195	11000011	216	11011000	237	11101101
175	10101111	196	11000100	217	11011001	238	11101110
176	10110000	197	11000101	218	11011010	239	11101111
177	10110001	198	11000110	219	11011011	240	11110000
178	10110010	199	11000111	220	11011100	241	11110001
179	10110011	200	11001000	221	11011101	242	11110010
180	10110100	201	11001001	222	11011110	243	11110011
181	10110101	202	11001010	223	11011111	244	11110100
182	10110110	203	11001011	224	11100000	245	11110101
183	10110111	204	11001100	225	11100001	246	11110110
184	10111000	205	11001101	226	11100010	247	11110111
185	10111001	206	11001110	227	11100011	248	11111000
186	10111010	207	11001111	228	11100100	249	11111001
187	10111011	208	11010000	229	11100101	250	11111010
188	10111100	209	11010001	230	11100110	251	11111011
189	10111101	210	11010010	231	11100111	252	11111100
190	10111110	211	11010011	232	11101000	253	11111101
191	10111111	212	11010100	233	11101001	254	11111110
192	11000000	213	11010101	234	11101010	255	11111111

3

DYNAMIC HOST CONFIGURATION PROTOCOL (DHCP)

INTRODUCTION

In the early days of the Internet's existence, when hosts numbered in the hundreds, assigning an IP address to a device was fairly trivial. It was simply one of the configuration parameters entered manually on each host. This "once and done" or *static* address assignment process using a hard-coded IP address certainly was simple, but it inhibited the host's mobility among different networks or subnets, which at the time when a "personal computer" was the size of a television, was a non-issue. Enabling mobility required the cumbersome task of reconfiguring the host with a new IP address based on the present location or network to which connection is desired, as we illustrated in Chapter 1.

Nonetheless, you will likely have a set of static addresses for devices on your network that do not require mobility, such as routers, servers, IP PBXs, etc. It's a good idea to keep track of which IP addresses on allocated subnets are statically assigned, which are assigned to address pools for dynamic assignment, and which are free or reserved for future use. Maintaining a subnet IP inventory is a recommended practice to maintain a record of addresses assigned on each subnet to minimize duplicate or otherwise erroneous IP address assignments. Just make

Introduction to IP Address Management, By Timothy Rooney

sure the inventory of static addresses matches what's actually been provisioned on the router, server, or statically addressed device. Performing periodic baselines of address assignments through various forms of discovery or ping sweeps can help identify any mismatches as we'll discuss in Chapter 6.

For those devices that obtain addresses dynamically, two main strategies are available to automate this address assignment process: client-based or network server-based. In Chapter 2, we introduced the concept of client-based address assignment in the form of IPv6 address autoconfiguration, a process whereby a client can determine its location based on router advertisements and automatically calculate its interface identifier to derive an address. However, if during the requisite process of duplicate address detection, the host determines that its autoconfigured address is already in use, it must rederive another address or await manual intervention. Network server-based address assignment such as DHCP enables a host to "announce" its presence in requesting an IP address, among other parameters, from a server in the network.

DHCP OVERVIEW

The Dynamic Host Configuration Protocol (DHCP) defines a client-server protocol for devices connecting to an IP network to automatically obtain an IP address. DHCP has been a tremendous time saver for IP network administrators. It enables a device to request an IP address, and have one or more DHCP servers within the IP network service the request without user intervention. For most end user devices such as laptops, VoIP phones, PDAs, and others, the DHCP process transpires "behind the scenes" upon device boot-up or connection to a wireline or wireless network without user intervention.

DHCP is supported as part of both IPv4 and IPv6. We'll discuss the IPv4 version first. DHCP is built on the foundation of an older protocol, the Bootstrap Protocol, referred to as BOOTP. BOOTP provides automation of address assignment but is restricted to pre-assigning a given IP address to a particular device, identified by its network interface (MAC) address. Thus a BOOTP server is configured with a list of MAC addresses and corresponding IP addresses. DHCP incorporates this functionality with the added capability of assigning IP addresses to clients without requiring a priori knowledge of each client's hardware address.

DHCP supports three types of IP address allocation:

1. Automatic allocation—the DHCP server assigns a permanent IP address to the client; we'll refer to this form of allocation as A-DHCP.
2. Manual allocation—like BOOTP, the DHCP server assigns a "fixed" IP address based on the particular device's hardware address; we'll refer to this form of allocation as M-DHCP.
3. Dynamic allocation—the DHCP server assigns an IP address for a limited time period, after which it can be reassigned, perhaps to a different device; we'll refer to this form of allocation as D-DHCP.

Automatic allocation may be useful for a particular set of users or devices requiring a permanent IP address assignment via DHCP, where there's no requirement for a particular user or device to have a particular IP address. In other words, you may want to set aside a number of "permanent" addresses without directly associating each IP address with a particular hardware address. This is in contrast to Manual DHCP, which associates a particular hardware address with a corresponding IP address.

Dynamic allocation is commonly used to set up address pools in DHCP servers in order to "reuse" IP addresses. In this way, the DHCP server can assign an IP address to a particular client for a given time period referred to as the lease time; then when the IP address becomes available due to the expiration of the lease or the client relinquishing the address, reassign the same address to a different client. Under dynamic allocation, the DHCP server leases its IP addresses to clients for a fixed period of time. The lease time is a configurable parameter within the DHCP server.

Regardless of the DHCP address allocation type, the process by which a DHCP client obtains a lease is the same. The basic process begins with a DHCP client broadcasting a DHCPDISCOVER (Discover) packet. Since the client does not have an IP address, nor generally any information about the IP network, it inserts the all-zeroes address as the source address and the broadcast (all-ones) address as the destination address within the IP header. Let's assume that a DHCP server has been deployed on the same subnet to which the DHCP client is connected. Upon receiving the Discover packet, the DHCP server will determine if it has an address available on this subnet on which the Discover was received.

If an address is available in the pool, the DHCP server will send a DHCPOFFER (Offer) packet to the client, offering an IP address and associated configuration parameters, called *options*. The DHCP client may receive more than one Offer if multiple DHCP servers are servicing this subnet. The client will select one configuration set and broadcast a DHCPREQUEST (Request) packet, specifying the selected DHCP server whose offer it has accepted. The selected DHCP server will acknowledge the Request with a DHCPACK (Ack) once it has recorded the lease information in non-volatile storage, thereby binding the IP address to the DHCP client. This basic message flow is sometimes referred to as the "DORA" process—Discover, Offer, Request, and Ack (Figure 3-1).

In this simple example, the DHCP server resides on the same subnet as the DHCP client. The client broadcasts the Discover packet on the network. Since the DHCP server resides on the same network, it receives the broadcast and processes the packet. Knowing the network from which the broadcast originated, the DHCP server can assign an available IP address on the network. But do you have to deploy a DHCP server on every subnet? Fortunately, no; the DHCP server simply must be reachable from the subnet via the IP routing infrastructure. The router(s) receiving the Discover broadcast packet will not propagate the broadcast as this would create excessive and needless IP packet traffic. Instead, the router will forward or *relay* the packet via unicast directly to the intended

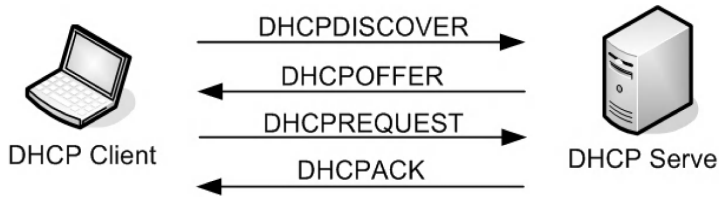

Figure 3-1: DHCP "DORA" process

DHCP server. Each router configured to perform this relay function is referred to as a *relay agent*. Each relay agent must be configured with the IP addresses of each DHCP server serving the subnet. This configuration parameter, commonly referred to as the DHCP Relay address, enables the router to accept the Discover broadcast, look up the DHCP server(s) configured for DHCP Relay, and then route the Discover packet via unicast directly to each DHCP server as illustrated in Figure 3-2.

In the process, the router modifies the DHCP Discover packet to insert the IP address of the interface on which the Discover was received into the Relay Agent (Gateway) Interface Address field. This parameter enables the DHCP server to identify the subnet on which an address assignment has been requested. Note that when the Gateway Interface Address (GIAddr) field is zero, the DHCP server assumes the subnet on which to assign the IP address is the same as that on which the Discover was received (via direct broadcast).

DHCP Servers and Address Assignment

Each DHCP server can be configured with multiple address pools serving several different subnets in various locations. In fact, for some DHCP server implementations, the same address pool can be configured on multiple DHCP servers for redundancy. The DHCP server keeps track of the state of all IP addresses across all of its configured address pools. When an address is leased to a client, the server generally tracks not only the lease time for the IP address, but an identifier for the client leasing the IP address. This identifier is typically the client

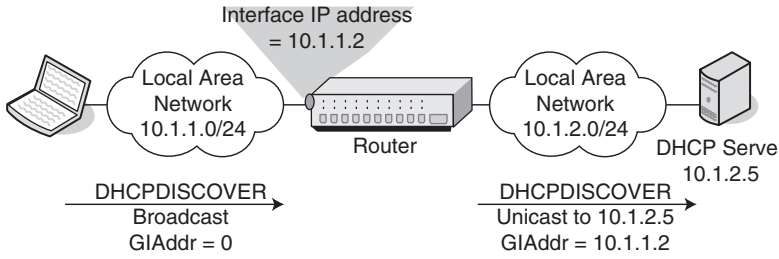

Figure 3-2: DHCP relay

hardware address (chaddr) of the client, though the client-identifier field, option 60, may also be used.

The use of the client-identifier (client ID) option over the chaddr field was suggested to maintain an identifier for the device even if the link interface hardware is moved to another device. But in practice, most devices either do not provide a client ID or copy the value of the chaddr field into the client ID option.

The basic decision process used by DHCP servers in offering an address is based on the following.

- If the client has a leased address as recorded in the DHCP server, the server will assign this address.
- If the client previously had an address that is now expired or released but is still available, the server will assign this address.
- If the client includes an address in the Requested IP Address option, option 50, and the address is available, the server will assign this address.
- The server will assign an available address from a pool on the same subnet on which the Discover broadcast was received if the GIAddr field is zero, or on the subnet indicated by the GIAddr value if nonzero. Additional criteria based on parameters within the Discover packet may dictate from which pool the address gets assigned, if there are multiple pools serving the subnet in question. These parameters are generically referred to as client class parameters and are discussed next.

Device Identification by Class

Client class parameters provide a means for the DHCP client to provide additional information to the DHCP server, and for the DHCP server to recognize clients requiring unique IP address or parameter assignments. For example, you may want to dedicate one address pool for VoIP devices and a separate pool for data devices. This may be motivated by administrative concerns or by source routing policies for voice vs. data packets from the respective devices. Most DHCP servers, including those available from the Internet Systems Consortium (ISC) or Microsoft, enable specification of vendor class or user class values to match to perform such categorization. The DHCP server can be configured to associate a particular vendor class or user class value or set of values as criteria in assigning an address from an address pool, along with associated options.

Let's consider an example: assume we need to define one address pool within a 10.32.128.0/23 subnet for laptops and a different pool for VoIP devices. In Figure 3-3, we'd like to configure our DHCP server to discriminate between the VoIP phones and laptops and assign addresses from respective address pools. The first step is to determine what information in the DHCP packet can be used to uniquely identify each class of device as VoIP phone vs. laptop. Typically, your VoIP phone provider will inform you that there is a particular string in the vendor class identifier option (option 60); let's say this string is "voip." We can

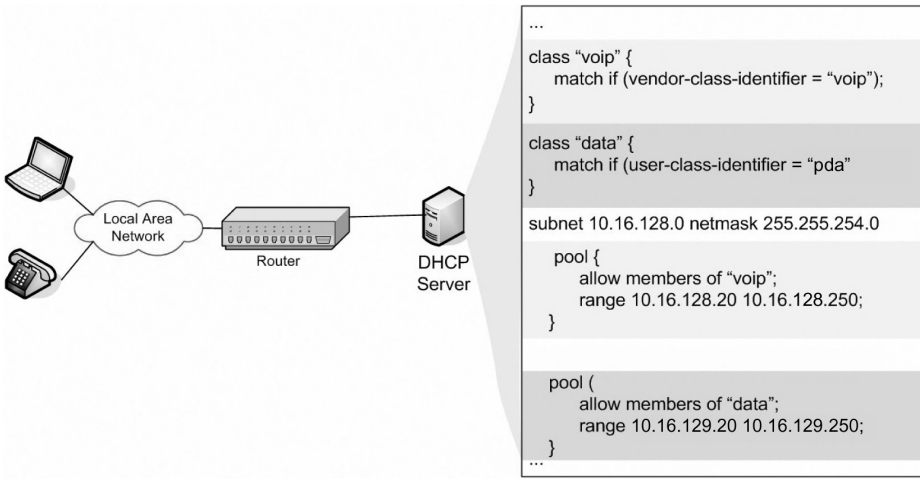

Figure 3-3: Client classing example using DHCP configuration pseudo-code based on (20)

define a client class in the DHCP server per the example in Figure 3-3, though the syntax will depend on your DHCP server vendor (or IPAM tool). In this example, we configure the DHCP server to categorize devices sending DHCP packets with option 60 = "voip" as devices of the voip class. Likewise, we can define a data class by defining the user class identifier option (option 77) having a value of "pda" per the match-if clause in Figure 3-3. Alternatively, this pool could be setup as the "default" pool for clients not matching other defined client classes.

Now that we've defined our two classes, enabling the DHCP server to identify packets as originating from devices belonging to one class or the other, we can now instruct the server how to handle these requests. We can define two address pools within the respective subnet declaration, as we'd like to separate the address assignments for these two classes of devices. Within a 10.16.128.0/23 subnet, we define one address pool for voip class devices as containing addresses 10.16.128.20–10.16.128.250 and a second address pool for data class devices as containing 10.16.129.20–10.16.129.250, shaded to map to 10.16.129.20–10.16.129.250 the class definition in the figure.

When configured per Figure 3-3, the DHCP server will now examine each DHCPDISCOVER packet from devices on the 10.16.128.0/23 subnet to discern the class of device, then assign voip devices an address from the 10.16.128.20–10.16.128.250 pool and assign data devices from the 10.16.129.20–10.16.129.250 pool. Note that there may be additional parameters or option settings you may wish to define within each of these pool statements to provide configuration information accordingly to each class of device.

Depending on the vendor of the DHCP server you deploy, there are various menu interfaces or text file editors that can be used for managing the configura-

tion of address pools and server behavior as well as criteria you can specify to dictate address assignment logic. For example, Microsoft DHCP servers can be configured through a Windows graphical user interface (GUI), while ISC DHCP servers can be configured via text editor. However, the ISC DHCP provides more flexibility in defining client classes beyond user class and vendor class; any parameter in the packet can be examined and filtered upon for client class processing, including the chaddr field for MAC address filtering, or any other parameter present. For mixed ISC and Microsoft environments, the use of a centralized IPAM system could help abstract the individual vendor interfaces and enable configuration of both with a single interface.

DHCP Options

Clients can request settings for particular options, and servers can assign these and other parameters based on the DHCP server configuration. DHCP administrators can define groupings of options to be assigned to all or certain DHCP clients based on the client's hardware address, client class value, or other DHCP packet parameter.

In the prior section, we set up two client classes corresponding to VoIP vs. data devices. Devices of these types will likely require different configuration parameters. For example, Cisco VoIP devices typically require option code 66 or 150, while Avaya VoIP devices require option 172, and data devices require neither of these. We've already described how client classes can be used to configure the DHCP server to distinguish different DHCP clients. We can now add options for each pool, which will be provided to clients receiving addresses in the corresponding pool. Alternatively, Manual DHCP address reservations enable mapping of a hardware address to a specific IP address, and associated DHCP options can also be defined for the device.

Please refer to Appendix A for a complete list of currently defined DHCP options.

DHCP FOR IPv6 (DHCPv6)

DHCP for IPv6 addresses is referred to as DHCPv6 and is defined in RFC 3315 (21). As defined, DHCPv6 is not integrated with DHCP for IPv4. This means that DHCPv6 will support IPv6 addresses and configurations only, not additionally IPv4 addresses and parameters. It's left to future development to define this should demand dictate.

DHCP Comparison IPv4 vs. IPv6

DHCPv6 uses different message types and packet formatting than DHCP for IPv4 but is similar in many ways. Table 3-1 highlights these similarities and differences.

TABLE 3-1: Comparison of DHCP for IPv4 and IPv6

Feature	DHCPv4	DHCPv6
Destination IP address of initial message from the DHCP client	Broadcast (255.255.255.255)	Multicast to link-scoped address: All-DHCP-Agents address (FF02::1:2)
DHCP Relay Support	Yes, using pre configured relay agent addresses	Yes, using All_DHCP_Servers site-scoped multicast address (FF05::1:3) or pre-configured
Relay Agent forwarding	Same message type code but inserts giaddr and unicasts to DHCP server(s)	Encapsulates client message in RELAY-FORW to DHCP server(s) and RELAY-REPL from server(s)
Message to locate server to obtain IP address and configuration	DHCPDISCOVER	SOLICIT
Server message to engage client	DHCPOFFER	ADVERTISE
Client message to accept parameters	DHCPREQUEST	REQUEST
Server acknowledgment of lease binding	DHCPACK	REPLY
Client message to leasing DHCP server to extend lease	DHCPREQUEST (unicast)	RENEW
Client message to any DHCP server to extend lease	DHCPREQUEST (broadcast)	REBIND
Client message to relinquish a lease	DHCPRELEASE	RELEASE
Client message to indicate that an offered IP address is already in use	DHCPDECLINE	DECLINE
Server message to instruct client to obtain a new configuration	DHCPFORCERENEW	RECONFIGURE
Request IP configuration only, not address	DHCPINFORM	INFORMATION-REQUEST

DHCPv6 Address Assignment

The DHCPv6 process begins with a client issuing a SOLICIT message, in essence requesting a "bid" from DHCP servers that can provide an IP address on the particular subnet to which the client is connected. Instead of broadcasting this initial packet as in IPv4, the SOLICIT message is sent by the client to the All_Relay_Agents_and_Servers multicast address, FF02::1:2.

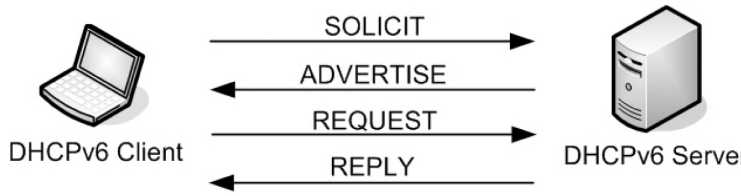

Figure 3-4: DHCPv6 address assignment

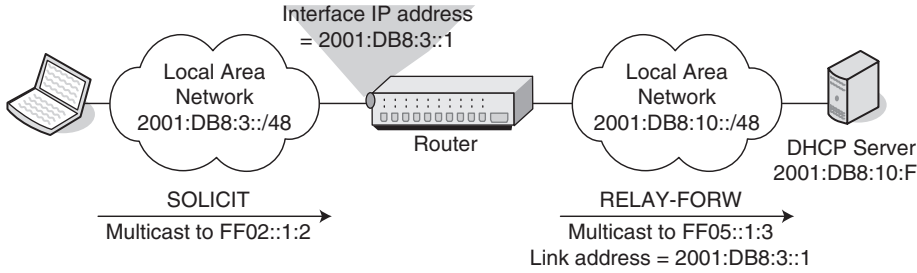

Figure 3-5: DHCPv6 relay

Any routers on the link configured as relay agents will receive the SOLICIT packet and will relay the packet to a DHCPv6 server(s). IPv6 relay agents do not require configuration of DHCP Relay addresses as in the IPv4 case, though they may enable such configuration. Instead, relay agents encapsulate the original Solicit packet within a Relay-Forw packet, which is then transmitted to the site-scoped All-DHCP-Servers multicast address (FF05::1:3). The Link Address field of the Relay-Forw packet indicates the link on which the client requesting an IP address currently resides. This information is used by the DHCP server in assigning an appropriate IP address for this link, in a manner similar to the DHCPv4 GIAddr field. This process is illustrated in Figures 3-4 and 3-5.

DHCPv6 servers on this subnet will receive the SOLICIT packet directly, and others responding to the site-scoped All_DHCP_Servers multicast address will receive the SOLICIT packet encapsulated within a RELAY-FORW packet. In either case, the DHCPv6 server may respond with an ADVERTISE packet, indicating a preference value. The preference value is intended to enable the client to select the server advertising the highest preference as configured by administrators. The server will also indicate if it has no addresses available on the subnet. The ADVERTISE packet will be unicast to the client if the SOLICIT had been received directly using the client's source IP address from the SOLICIT packet (most likely the client's link local address). If the SOLICIT had been received by the server via a RELAY-FORW packet from a relay agent, the ADVERTISE message will be encapsulated in a RELAY-REPL packet and unicast to the corresponding relay agent.

The client analyzes the advertisements received, and selects a server from which to request an IP address, typically with the highest preference, and issues a REQUEST message to the server. The server will then record the address assignment and reply to the client with a REPLY message.

When the client receives a Reply packet to confirm the address assignment, the client must perform duplicate address detection to assure no other device is already using the IP address due to autoconfiguration or manual configuration. If another device is using the assigned IP address, the client would send a Decline message to the DHCP server, indicating that the address is in use. The client can then reinitiate the DHCP process to obtain a different IP address.

In addition to the four-packet exchange outlined above, DHCPv6 features a rapid commit option. This halves the messaging requirements by enabling the server to simply REPLY to a SOLICIT packet. The client would include the rapid commit option in its SOLICIT message. Servers responding with an address assignment would issue a REPLY packet directly, also including the rapid commit option. Note that each server responding will assume the address it assigned is leased, so rapid commit should be used with either short lease times or for support by a limited number of servers if normally there are many serving the same subnet.

As in IPv6 autoconfiguration described in Chapter 2, each nontemporary IPv6 address assigned via DHCP has a preferred lifetime and a valid lifetime. After the preferred lifetime expires, the address is considered valid but deprecated. No new IP communications sessions should utilize the address while deprecated.

DHCPv6 Prefix Delegation

DHCPv6 is used not only to assign individual IP addresses and/or associated IP configuration information to hosts, but can also be used to delegate entire networks to requesting router devices. This form of delegation via DHCPv6 is called *prefix delegation*. The original motivation for prefix delegation arose from broadband service providers seeking to automate the process of delegating IPv6 subnets (e.g., /48 to /64 networks) to broadband subscribers in a hierarchical manner. A requesting router device at the edge of the service provider network, facing subscribers, would issue a request for address space via the DHCPv6 protocol to a delegating router. Note the terminology: this is intended to be an inter-router protocol though a DHCPv6 server could perform the functions of the delegating router.

The prefix delegation process utilizes the same basic DHCPv6 message flow described above for address assignment per Figure 3-4: Solicit, Advertisement, Request, and Reply. Additional information within the corresponding DHCPv6 messages can be used to determine an appropriate network for delegation. Like IP addresses, prefixes have preferred and valid lifetimes. The requesting router can request a lifetime extension via the DHCPv6 Renew and Rebind messages.

DHCPv6 Support of Address Autoconfiguration

When we discussed IPv6 autoconfiguration in Chapter 2, we defined three types of autoconfiguration:

- Stateless—This process is "stateless" in that it is not dependent on the state or availability of external assignment mechanisms, for example, DHCPv6.
- Stateful—The stateful process relies solely an external address assignment mechanism such as DHCPv6, as described above.
- Combination Stateless and Stateful—This process involves a form of stateless address autoconfiguration used in conjunction with stateful configuration of additional IP parameters.

This third combined form of autoconfiguration leverages DHCPv6 not for IPv6 address assignment, but for assignment of additional parameters, encoded as DHCPv6 options. The client can request configuration parameters via the Information-Request message, indicating which option parameter values it is seeking. A server(s) that is able to supply the desired configuration parameters will respond with a Reply message with the corresponding option parameters.

Device Unique Identifiers (DUIDs)

Like DHCPv4, DHCPv6 servers must track the availability and assignment of IP addresses within its configured address pools, and identify requestors and holders of IP addresses. DHCPv6 utilizes the Device Unique Identifier, or DUID, to identify clients. DUIDs are used not only for servers to identify clients, but for clients to identify servers. The DUID is analogous to the client-identifier concept in that DUIDs are intended to be globally unique for a device, not an interface. DUIDs should not change over time, even if the device undergoes changes in hardware. DUIDs are constructed in various manners automatically by IPv6 nodes. They consist of a two octet type code followed by a variable number of octets based on the type. The following DUID type codes are currently defined:

- Type = 1—Link layer address plus Time (DUID-LLT)
- Type = 2—Vendor-assigned unique ID based on Enterprise Number (DUID-EN)
- Type = 3—Link-layer based DUID (DUID-LL)

For those based on link layer address, they are to be used for *all* device interfaces, even if the hardware from which the link layer address was obtained is removed. The DUID is a device identifier, not an interface identifier.

Identity Associations (IAs)

While DUIDs are associated with all interfaces of a device, and IP addresses are assigned to interfaces, you may be wondering how the device and server identify particular interfaces for a given DUID. The concept of the Identity Association (IA) provides this linkage between a DHCPv6 server and a client interface for individual address assignment. IAs are differentiated by type between those for temporary addresses (IA_TA), which are short-leased, non-renewable addresses, those for non-temporary addresses (IA_NA), and those for prefix delegation (IA_PD).

Temporary address assignments assuage privacy concerns associated with auto-configured addresses based on hardware addresses (i.e., modified EUI-64 interface IDs), which do not change over time. The concern is that a given Interface ID within an IPv6 address does not change unless the underlying hardware interface changes. Thus, even if the network upon which a device is connected changes from day-to-day, the interface ID does not. The ability to track the location of a device, and thus its user, becomes relatively easy; hence the concern with privacy. The use of short-lived, nonrenewable address assignments via DHCP can be one approach to address this concern, hence the concept of temporary addresses. Please see RFC 3041 (22) for more background on this privacy issue.

For individual address assignment, temporary or nontemporary, each client interface has an IA, identified by an IA Identifier, or IAID. The IAID is represented as four octets in client-server DHCPv6 communications and is chosen by the client. The IAID must be unique among all IAIDs associated with the client and must be stored persistently across client reboots or consistently derivable upon each reboot. The client specifies its DUID and IAID for which an address is being requested from the DHCPv6 server. The DHCPv6 server assigns an IPv6 address to the IAID, along with the corresponding T1 (renew) and T2 (reboot) timer values.

IA_PDs are not necessarily associated with a device interface. Recall that the requesting router is using DHCPv6 to obtain an IPv6 network delegation. The requesting router must derive one or more IA_PDs for use within DHCPv6, and it must be persistent across reboots or consistently derivable.

DHCPv6 Options

As with DHCP for IPv4, additional configuration parameters may be defined on a DHCPv6 server for assignment to corresponding DHCPv6 clients. Please refer to Appendix B for a complete listing of currently defined DHCPv6 options.

DHCP SERVER DEPLOYMENT CONSIDERATIONS

Whether you are planning a new or refreshed deployment of DHCP servers or just want to validate your current strategy, here are some key considerations when formulating the DHCP server deployment design.

- Response time requirements—Do your clients have stringent response time requirements? Most popular clients tolerate response times in the seconds, but certain applications may be more demanding. The more stringent your requirements, the more important will be server performance and perhaps client proximity.
- Load requirements—Do you have certain load conditions that must be handled? For broadband service providers utilizing DHCP as a customer premises equipment initialization technology, load spikes may occur upon recovery from a residential power outage or equipment installation or reboot. For enterprise environments, such a spike could occur at the start of the work day if several associates arrive at or near the same time, though many devices will simply attempt to renew an IP address previously used by default.
- Traffic expectations—Do you employ short lease times to minimize overbooking, which causes more frequent renewal attempts? Generally the shorter the lease time, the shorter the interval between obtaining the lease and subsequent lease renewal attempts. This drives increasing traffic on the network to and from the DHCP server(s) and must be considered when designing to the aforementioned response time and load requirements for server quantities and associated bandwidth.
- Availability requirements—Do your clients positively have to be able to obtain an IP address or configuration via DHCP 24X7 or is the service "best effort" based? Most will answer that high availability is critical, but with devices growing increasingly multi-networked, as long as one network's address assignment mechanism is available, this may be acceptable (of course this statement assumes different DHCP services serve these different interfaces, which may not be the case). Mean time to repair (MTTR) is another consideration in meeting DHCP services availability objectives. Having a spare server locally can shorten MTTR while having to order a replacement will delay this process.

The first three considerations above relate to deploying sufficient quantities of servers of given lease distribution rate to meet respective performance objectives. A good starting point is to identify the number of expected DHCP clients at each site on your network. This number should account for all devices requiring DHCP, including data devices, voice devices, and all IP devices requiring DHCP at each site. Don't forget to account for "peak" quantities of users and devices so that everyone, even associates visiting on temporary basis, may obtain a valid lease.

After accounting for peak quantities of DHCP clients, consider the frequency of DHCP transactions. This will be dependent on your lease times, as well as client lease/release configuration. Most clients will "remember" a prior lease and attempt to request it upon power-up, for example, when an employee returns to work the next day, though this is not always the case.

The fourth consideration listed above relates to providing high-availability DHCP services for DHCP clients. Given the general importance of providing highly available DHCP service, deploying for high availability is typically recommended. Once you've designed your deployment based on performance requirements, total or selective high availability may be planned. Based on server technologies you plan to deploy, implementation of high availability will impact not only the number of servers required, but potentially your address space plan.

The popular ISC and Microsoft DHCP implementations utilize vastly different approaches. The ISC server employs a failover protocol such that for a given address pool, one DHCP server will act as the primary, while a second DHCP server will act as the backup or failover server. In this manner, if a given DHCP server is down, a client may obtain an address from the same address pool from a failover DHCP server. Load balancing among the two servers may also be configured.

The Microsoft approach is to split the pool or scope into two and deploy one portion (80% of the pool) on a local DHCP server and the other portion (20%) on a remote DHCP server. This approach enables most of the clients to obtain an IP address from the local DHCP server while providing a backup in the event the local server is down. Respective pool sizings in this "split scope" approach for each server should be increased if possible up to 100% on each, which doubles the effective pool size but provides full scope redundancy in the event of a lengthy outage.

DHCP Deployment on Edge Devices

Most router vendors provide a DHCP service as a component of their router platforms. This may lead one to question whether a separate server is needed to support DHCP services. As with most design questions, the answer is, "it depends." Small environments with a few sites with local routers serving up to 100 or so monolithic clients each may be well served by configuring the router to provide DHCP services. However, larger organizations or those requiring more advanced DHCP services, for example, for discriminating voice vs. data clients for address and option parameter assignment, would be better served deploying discrete (non-router-integrated) DHCP servers.

The advantages of running DHCP on a router device include:

- Lower hardware cost—no need to procure a server or set of servers.
- Single user interface—the same command line interface can be used to configure the router and the DHCP server, and no relay agent configuration is needed.
- "Fewer moving parts"—one less communication link and server required to perform DHCP functions.

The main disadvantages of running DHCP on a router are:

- Options support—most router based DHCP servers are primitive, supporting address assignment but little in the way of options support.
- Client class support—major vendors do not support client classes, which is required for discriminatory address/option assignment to different devices, for example, VoIP vs. data devices.
- No failover—if a router fails, you've probably lost connectivity in any case, but if there are two routers serving a subnet for redundancy a split scopes approach would have to be employed, increasing management complexity.
- No centralized management—router-based DHCP services are configured via command line and unless a centralized tool is employed, each router DHCP server must be configured manually with respect to the IP addressing plan; less likely support is possible if multiple router vendor products are in use.

OTHER MEANS OF DYNAMIC ADDRESS ASSIGNMENT

While DHCP provides a means for network administrators to preallocate dynamic address pools on a number of subnets and provide a mechanism to discern different device types for discriminatory assignment of an IP address and configuration parameters, there are other methods, albeit less popular, for dynamic address assignment. A common alternative method besides address auto-configuration is the use of a Radius server to assign an IP address. Radius, or its successor protocol, Diameter, provides an Authentication, Authorization, and Accounting (AAA) service for IP hosts attempting to access a network. The connection from a client to a Radius server is commonly performed via a Point-to-Point (PPP) or Extensible Authentication Protocol (EAP) connection, for example, when the client is attempting to access a network edge device or dial pool. The Radius server challenges the client to enter a user name and password, authenticates the entered information against its internal or external database, then provides access to the network by providing an IP address to the client.

While vastly simplifying the Radius protocol, the relevant concept here is that some Radius servers or even edge router devices, can be configured with address pools from which individual IP address assignments can be made to authorized clients. In some cases, Radius servers can be configured to actually utilize the DHCP protocol to obtain an address from a DHCP server. In this case, the Radius server acts as a DHCP proxy client to obtain an IP address on behalf of, and for assignment to, the requesting client.

KEY DHCP MANAGEMENT CHALLENGES

DHCP management challenges are summarized below. We'll discuss techniques for addressing these challenges beginning in Chapter 5.

- Address pool sizing to meet capacity requirements for clients of IPv4 and IPv6 and by class
- Identifying class-specific configuration requirements and mapping this to client identifying parameters (e.g., vendor class identifier option) and corresponding client configuration parameter requirements
- Configuring the appropriate DHCP server(s) with the allocated address pools, client class definitions, and associated general and class-specific pools and option values
- High-availability planning and management
- Monitoring DHCP servers as well as address pool capacity to assure availability of DHCP services and addresses for clients

4

THE DOMAIN NAME SYSTEM (DNS)

DNS OVERVIEW—DOMAINS AND RESOLUTION

DNS is the third cornerstone of IPAM and a foundational element of IP communications. DNS provides the means for improved usability of IP applications insulating end users from communicating by entering IP addresses. Certainly, to communicate over an IP network, an IP device needs to send IP packets to the intended destination IP device; and as we have seen, the IP packet headers require source and destination IP addresses. DNS provides the translation from a user-entered named destination, for example, web site address, to its IP address.

As a network service, DNS has evolved from simple host name-to-IP address lookup utility to enabling very sophisticated "look-up" applications supporting voice, data, multimedia, and security applications. DNS has proven extremely scalable and reliable for such lookup functions. We'll discuss how this lookup process works after first introducing how this information is organized.

Domain Hierarchy

The global domain name system is effectively a distributed hierarchical database. Each "dot" in a domain name indicates a boundary between tiers in the hierarchy, with each name in between dots denoted as a *label*. The top of the hierarchy,

Introduction to IP Address Management, By Timothy Rooney

the "." or root domain provides references to top-level domains, such as .com, .net, .us, .uk, which in turn reference respective subdomains. Each of these top-level domains, or TLDs, is a child of the root domain. Each TLD has several children domains as well, such as ipamworldwide.com with the ipamworldwide domain beneath the com domain. And these children may have children domains and so on.

As we read between the dots from right to left, we can identify a unique path to the host we are seeking. The text left of the leftmost dot is generally the host name (some environments allow dots within hostnames, which is relatively uncommon though permissible), which is located within the domain indicated by the rest of the domain name. A *fully qualified domain name* (FQDN) refers to this unique full [absolute] path name to the node or host within the global DNS data hierarchy. Figure 4-1 illustrates a fully qualified domain name mapping to the tree-like structure of the DNS database. Note that the trailing dot after .com. explicitly denotes the root domain within the domain name, rendering it fully

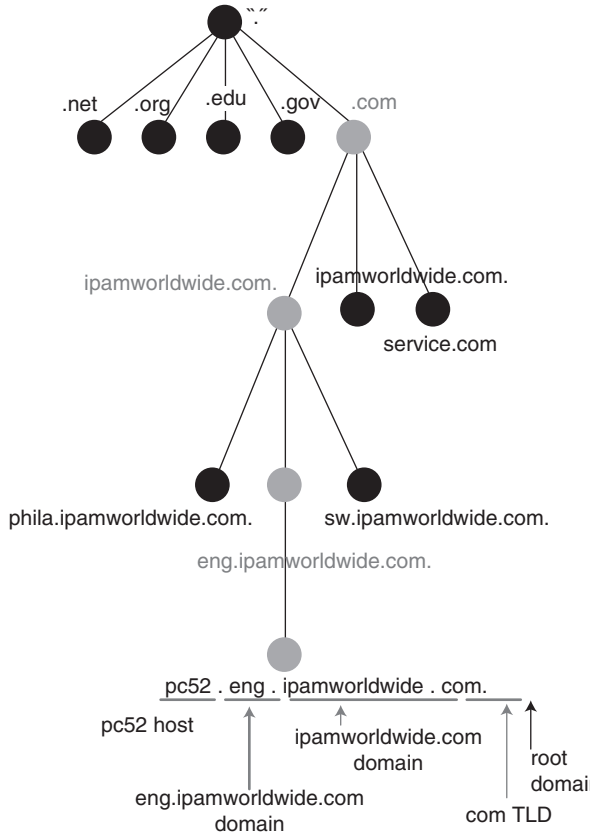

Figure 4-1: Domain tree mapping to a fully qualified domain name

qualified. Keep in mind that without this explicit FQDN trailing dot notation, a given domain name may be ambiguously interpreted as either fully qualified or relative to the "current" domain. This is certainly legal and easier shorthand notation, but just be aware of the potential ambiguity.

NAME RESOLUTION

To illustrate how domain information is organized and how a DNS server leverages this hierarchical data structure, let's take a look at an example of name resolution. Let's assume I'd like to connect to a device named `pc52` per the example in Figure 4-1. Thus I enter the host domain name, `pc52.eng.ipamworldwide.com.` as my intended destination. The application into which I type this domain name (e.g., email client, web browser, etc.) utilizes the sockets* application programming interface (API) to communicate with a portion of code within the TCP/IP stack called a *resolver*. The resolver's job in this instance is to translate the web server name I typed into an IP address that may be used to initiate IP communications.

The resolver issues a query for this host name to my local DNS server, requesting that the server provide an answer. The IP address of this local DNS server is configured either manually** or via DHCP using the domain servers option (option 6 in DHCP and option 23 in DHCPv6). This DNS server will then attempt to answer the query by looking in the following areas in the specified order and as illustrated in the Figure 4-2.

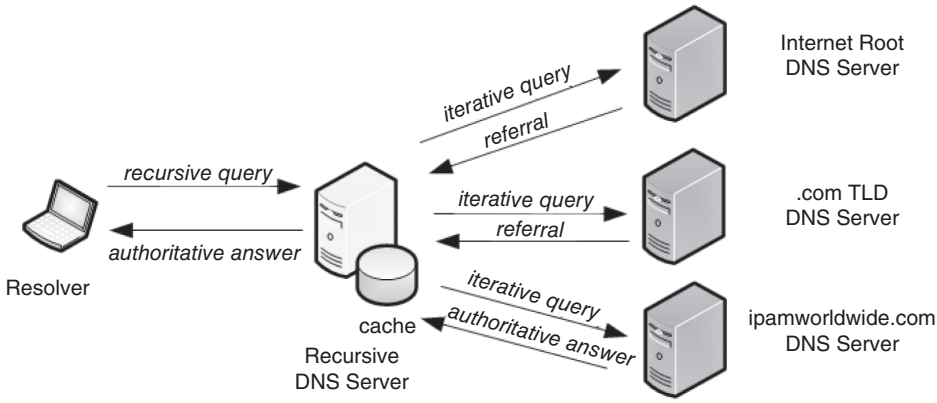

Figure 4-2: Recursive and iterative queries in name resolution

*This API call is from the application to the TCP/IP layer of the protocol stack. The gethostbyname sockets/Winsock call initiates this particular process.

**We'll review how to perform this manual configuration a little later in this chapter.

We often refer to this DNS server to which the resolver issues its query as a *recursive server*. "Recursive" means that the resolver would like the DNS server to try to find the answer to its query if it does not know itself. From the resolver's viewpoint, it issues one query and expects an answer. From the recursive DNS server's perspective, it attempts to locate the answer for the resolver. The recursive server is the resolver's "portal" into the global domain name system. The recursive server accepts recursive queries directly from client resolvers and performs the steps outlined below to obtain the answer to the query on behalf of the resolver.

Beginning on the left of Figure 4-2, the resolver initiates a recursive query to the recursive DNS server. The queried server will first search its configured data files. That is, the DNS server is typically configured with configuration and resolution information for which it is *authoritative*. This information is typically configured using text files, a Windows interface, or an IPAM system. For example, your company's DNS servers are likely configured with resolution information for your company's IP devices. As such, this is authoritative information. If the answer is found, it is returned to the resolver and the process stops.

If the queried server is not authoritative for the queried domain, it will access its cache to determine if it recently received a response for the same or similar query from another DNS server during a prior resolution task. If the answer for `pc52.eng.ipamworldwide.com.` resides in cache,* the DNS server will respond to the resolver with this nonauthoritative information and the process stops. The fact that this is not an authoritative answer is generally of little consequence, but the server alerts the resolver to this fact in its response.

If the queried DNS server cannot locate the queried information in cache, it will then attempt to locate the information via another DNS server that has the information. There are three methods used to perform this "escalation."

a. If the cache information referenced above indicates a partial answer to the query, it will attempt to contact the source of that information to locate the ultimate source and answer. For example, a prior query to another DNS server, server A, may have indicated that DNS server A is authoritative for the `ipamworldwide.com` domain. The initially queried DNS server may then query DNS server A for information leading to resolution of `pc52.eng.ipamworldwide.com.`
b. If the cache does not provide useful information, and the recursive server is configured to forward queries, the recursive server will forward the query as configured in its configuration or zone file. Forwarding can be used to funnel Internet-bound queries through a set of caching DNS servers as we'll discuss later in the chapter.

*Cache entries are temporary and are removed by DNS servers based on user configuration settings as well as advertised lifetime of a resource record.

If no information is found in cache, the server cannot identify a referral server, or forwarding did not provide a response* or is not configured, the DNS server will access its *hints* file. The hints file provides a list of root name servers to query in order to begin traversing down the domain hierarchy to a DNS server that can provide an answer to the query.

Note that by issuing queries to other DNS servers to locate resolution information, the recursive server itself performs a resolver function to execute this lookup. The term *stub resolver* is commonly used to identify resolvers, like those within end user clients, that are configured only with name servers to query.

Upon querying either a root server or a server further down the tree based on cached information, the queried server will either resolve the query by providing the IP address(es) for `pc52.eng.ipamworldwide.com.` or will provide a referral to another DNS server further down the hierarchy "closer" to the sought fully qualified domain name. For example, upon querying a root server, you are guaranteed that you will not obtain a direct resolution answer for `pc52.eng.ipamworldwide.com.` However, the root name server will refer the querying DNS server to the name servers that are authoritative for `com.` The root servers are "delegation-only" servers and do not directly resolve queries, only answering with delegated name server information for the queried TLD.

The recursive server *iterates* additional queries based on responses down the domain tree until the query can be answered. These point-to-point iterative queries are illustrated to the right of Figure 4-2. Upon querying the name server that is authoritative for `com.`, the com TLD server will respond with a referral to the name server that is authoritative for `ipamworldwide.com.`, and so on down the tree.

Ultimately, a DNS server that is authoritative for the zone of relevance to the query should be located. The authoritative DNS server will read the corresponding zone information for a *resource record* of the type being queried. The server will pass the resource record(s) to the querying (recursive) DNS server. When the answer is received, the recursive DNS server will provide the answer to the resolver and also update its cache and the process ends.

In summary, the resolution process entails a) finding a name server with authoritative information to resolve the query in question and b) querying that server for the desired information. In our example, the desired information was the IP address corresponding to the domain name `pc52.eng.ipamworldwide.com.` This "translation" information mapping the queried domain name to an IP address is stored in the DNS server in the form of a resource record. We'll discuss resource records later in the chapter.

The bottom line is that DNS servers are configured at all levels of the domain tree as authoritative for their respective domain information, including where to direct queriers further down the domain tree. In many cases, these servers at

* If the `forward only` option is configured, the resolution attempt will cease if the forwarded query returns no results; if the `forward first` option is configured, the process outlined in this paragraph ensues, with escalation to a root server.

different levels are administered by different organizations. Not every level or node in the domain tree requires a different set of DNS servers as an organization may serve multiple domain levels within a common set of DNS servers.

While the top three layers of the domain tree typically utilize three sets of DNS servers under differing administrative authority, the support of multiple levels or domains on a single set of DNS servers is a deployment decision. This decision hinges primarily on whether administrative delegation is required or desired. For example, the DNS administrators for the `ipamworldwide.com` domain may desire to retain administrative control of the `eng.ipamworldwide.com.` domain, but to delegate `sw.ipamworldwide.com` to a different set of administrators and name servers. This leads us to a discussion regarding the distinction between zones and domains.

ZONES AND DOMAINS

The term *zone* is used to differentiate the level of administrative control with respect to the domain hierarchy. In our example, the `ipamworldwide.com` zone contains authority for the `ipamworldwide.com` and `eng.ipamworldwide.com` domains, while the `sw.ipamworldwide.com` zone retains authority for the `sw.ipamworldwide.com` domain as illustrated in Figure 4-3.

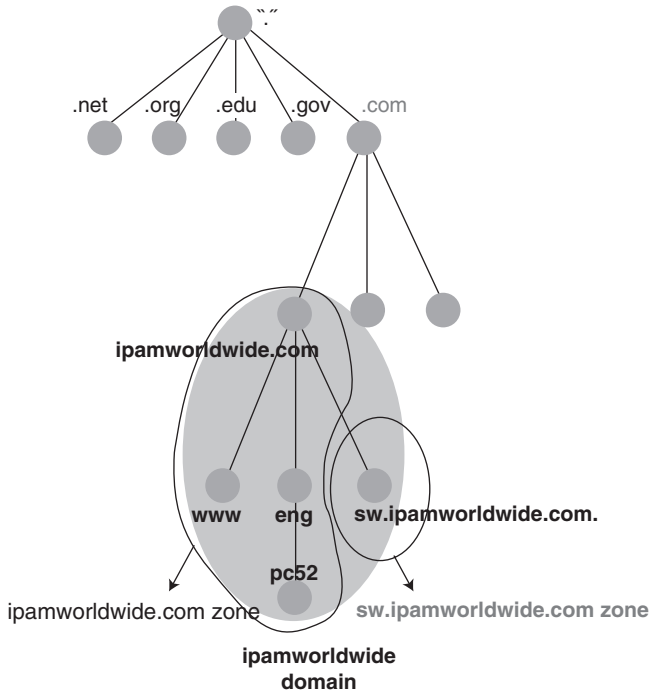

Figure 4-3: Zones as delegated domains

By delegating responsibility for sw.ipamworldwide.com, the DNS administrators for ipamworldwide.com agree to pass all queries for sw.ipamworldwide.com (and below in the domain tree for any subdomains of sw.ipamworldwide.com) to DNS servers administered by personnel operating the sw.ipamworldwide.com zone. These sw.ipamworldwide.com administrators can manage their domain and resource records and any children autonomously; they just need to inform the parent domain administrators (for ipamworldwide.com) where to direct queries they receive as resolvers or other DNS servers attempt to traverse down the domain tree seeking resolutions.

Thus administrators for the ipamworldwide.com zone must configure all resource records and configuration attributes for the ipamworldwide.com zone, including subdomains within the ipamworldwide.com zone such as the eng.ipamworldwide.com domain. At the same time, ipamworldwide.com administrators must provide a delegation linkage to any child zones, such as sw.ipamworldwide.com. This delegation linkage is supported by entering name server (NS) resource records within the ipamworldwide.com zone file, which indicate which name servers are authoritative for the sw.ipamworldwide.com delegated zone. These NS records provide the continuity to delegated child zones by referring resolvers or other name servers further down the domain tree. Corresponding A or AAAA records called *glue records* are also usually defined to glue the resolved NS host domain name to an IP address for further queries.

The shading in Figure 4-3 indicates that the ipamworldwide.com domain contains the ipamworldwide.com node plus all of its children, highlighting this level of responsibility for ipamworldwide.com and "below." The ipamworldwide.com DNS administrators are responsible for maintaining all DNS configuration information for the ipamworldwide.com zone as well as referrals to DNS servers serving delegated child zones. Thus when other DNS servers around the world are attempting to resolve any name ending in ipamworldwide.com on behalf of their clients, their queries will require traversal of the ipamworldwide.com DNS servers and perhaps other DNS servers, such as those serving the sw.ipamworldwide.com zone.

The process of delegation of the name space provides autonomy of DNS configuration while providing linkages via NS record referrals within the global DNS database. As you can imagine, if the name servers referenced by these NS records are unavailable, the domain tree will be broken at that point, inhibiting resolution of names at that point or below in the domain tree. If the sw.ipamworldwide.com DNS servers are down, authoritative resolution for sw.ipamworldwide.com *and its children* will fail. This illustrates the requirement that each zone must have at least two authoritative DNS servers for redundancy.

DNS Server Redundancy

Given the criticality of the DNS service in resolving authoritatively and maintaining the domain tree linkages, DNS server redundancy is a must. Different

DNS server vendors take different redundancy approaches. Microsoft replicates DNS information among a set of domain controllers when DNS information is integrated into Active Directory, an architectural foundation of the Windows Server products. The ISC BIND implementation supports DNS information replication through a hub-and-spoke model. Configuration changes are made to a *master* DNS server. Redundant DNS servers are configured as *slaves* or *secondaries*, and they obtain zone updates by the process of *zone transfers*. Zone transfers enable a slave server to obtain the latest copy of its authoritative zone information from the master server. Microsoft Active Directory–integrated DNS servers also support zone transfers for compatibility with this standard process.

Versions of zone files are tracked by a zone serial number, which must be changed every time a change is applied to the zone. Slaves are configured to periodically check the zone serial number set on the master server; if the serial number is larger than its own value defined for the zone, it will conclude that it has outdated information and will initiate a zone transfer. Additionally, the server that is master for the zone can be configured to *notify* its slaves that a change has been made, stimulating the slaves to immediately check the serial number and perform a zone transfer to obtain the updates more quickly than awaiting the normal periodic update check.

Zone transfers may consist of the entire zone configuration file, called an absolute zone transfer (AXFR) or of the incremental updates only, called an incremental zone transfer (IXFR). In cases where zone information is relatively static and updated from a single source, for example, an administrator, the serial number checking with AXFRs as needed works well. These so-called *static zones* are much simpler to administer than their counterpart: *dynamic zones*. Dynamic zones, as the name implies, accept dynamic updates, for example, from DHCP servers updating DNS with newly assigned IP addresses and corresponding domain names. Updates for dynamic zones can utilize IXFR mechanisms to maintain synchronization among the master and multiple slave servers. With BIND 9, journal files on each server provide an efficient means to track dynamic updates to zone information. Many server implementations load the zone file information into memory along with incremental zone updates, which are also loaded into memory for fast resolution.

Reverse Domains

Until now, we've introduced the common name-to-IP address resolution process, locating a DNS server authoritative for a name resolution, which then responds authoritatively to the query. Another popular form of query is for IP address-to-name resolution. This "reverse" form of resolution is commonly used as a security check when establishing virtual private network (VPN) connections or for general IP address-to-hostname lookups. Given an IP address, how does a DNS server traverse the domain tree to find a host domain name? Special top-level domains are defined for IP address-based domain trees within the *Address and Routing*

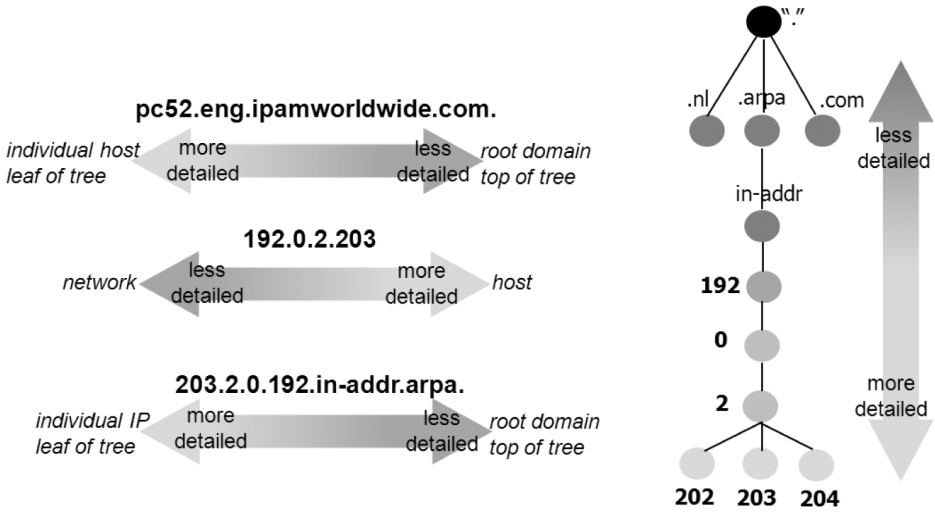

Figure 4-4: IP address (reverse) domain tree mapping

Parameter Area (arpa) domain: `in-addr.arpa.` is defined for IPv4 address-to-name resolution and `ip6.arpa.` is for IPv6 address-to-name resolution.

The only wrinkle in organizing IP addresses within a domain tree results from mapping an IP address, which reads left-to-right as less detailed (network) to more detailed (IP host), while reading a domain name left-to-right reads more specific (specific host, domain) to less specific (root). Therefore, the IP address is reversed to enable representation within the domain hierarchy, reading left-to-right as more specific to less specific. This is illustrated in Figure 4-4.

IPv6 Reverse Domains

IPv6 reverse domain mapping is a bit more cumbersome. As with IPv4, the IPv6 address must be reversed, maintaining its hexadecimal format. But the IPv6 address must first be "padded" to the full 32-hex digit representation; that is, the two forms of abbreviation discussed in Chapter 2 must be removed by including leading zeroes between colons and filling in double colon-denoted implied zeroes. Figure 4-5 illustrates an example of the process for the IPv6 address 2001:DB8:B7::A8E1. The address must be expanded or padded and the digits reversed. Then, this result must be "domain-ized" by removing the colons, inserting dots between each digit, and appending the ip6.arpa. upper level domains.

Figure 4-6 illustrates the logic in reversing the IPv6 address in order to be represented in a domain hierarchy as read left-to-right as more specific to less specific. This is directly analogous to Figure 4-4, which illustrates this concept for IPv4 addresses. The full 32-hex digit representation used in Figure 4-6 provides a unique, though lengthy, traversal down the `ip6.arpa.` domain tree (not shown).

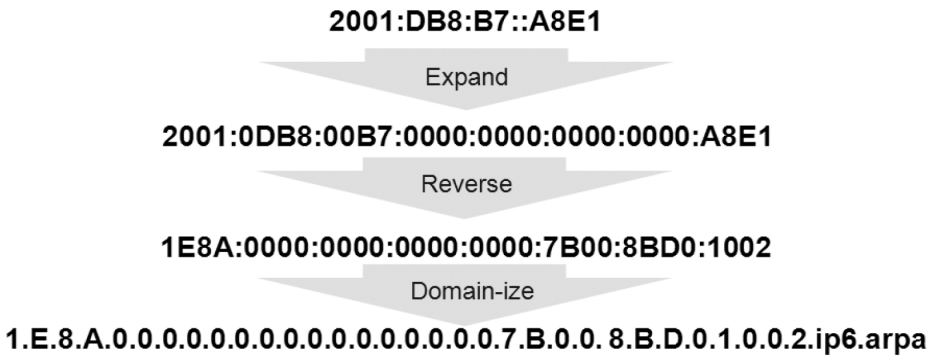

Figure 4-5: IPv6 address to reverse domain mapping

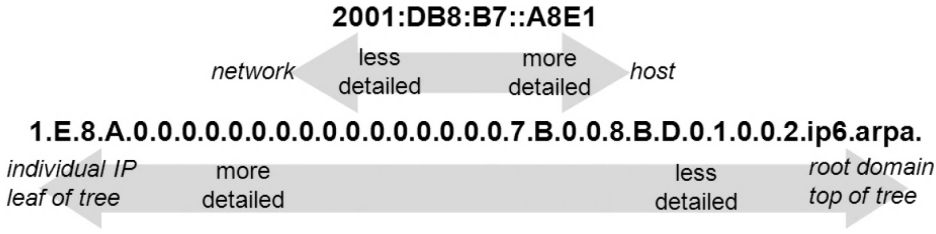

Figure 4-6: The IPv6 reverse domain notation

Additional Zones

Root Hints. We mentioned a *hints file* during the overview of the resolution process. This file should provide DNS server names and addresses that the server should query if the resolver query cannot be resolved via authoritative, forwarded, or cached data. The hints file will typically list the Internet root servers, which are authoritative for the root (.) of the domain tree. Querying a root server enables the querying server to start at the top to begin the traversal down the domain tree in order to locate an authoritative server to resolve the query. The contents of the hints file for Internet root servers may be obtained from www.internic.net/zones/named.root, though BIND and Microsoft DNS server implementations include this file with their distributions.

Some environments may require use of an internal set of root servers, where Internet access is restricted by organizational policy. In such cases, an internal version of the hints file can be used, listing names and addresses of internal root servers instead of the Internet root servers. The organization itself would need to maintain the listing of internal root servers, as well as their requisite root zone configurations.

Localhost Zones. Another zone file that proves essential is the localhost zone. The localhost zone enables one to resolve "localhost" as a hostname on the given server. A corresponding in-addr.arpa. zone file resolves the 127.0.0.1 loopback address. A single entry within the 0.0.127.in-addr.arpa zone maps address 1 to the host itself. This zone is required as there is no upstream authority for the 127.in-addr.arpa domain or subdomains. Likewise, the IPv6 equivalents need to be defined for the corresponding IPv6 loopback address, ::1. The localhost zone simply maps the localhost hostname to its 127.0.0.1 or ::1 IP address using an A and AAAA record, respectively.

RESOLVER CONFIGURATION

Like DHCP, DNS resolution occurs behind the scenes and involves a client and server. Ideally, end users don't even know it happens; they type in a web address and connect! The resolver software must be configured with which DNS server(s) to query for resolution. Thus, unlike DHCP, which requires no initial client configuration (since it simply broadcasts or multicasts to a well-known address), DNS does require some basic client configuration prior to use. This initial configuration may be performed manually or by obtaining this information from a DHCP server.

Figure 4-7 illustrates the configuration of a Microsoft Windows resolver in terms of manually defining the DNS server to query or the use of DHCP to obtain DNS server addresses automatically.

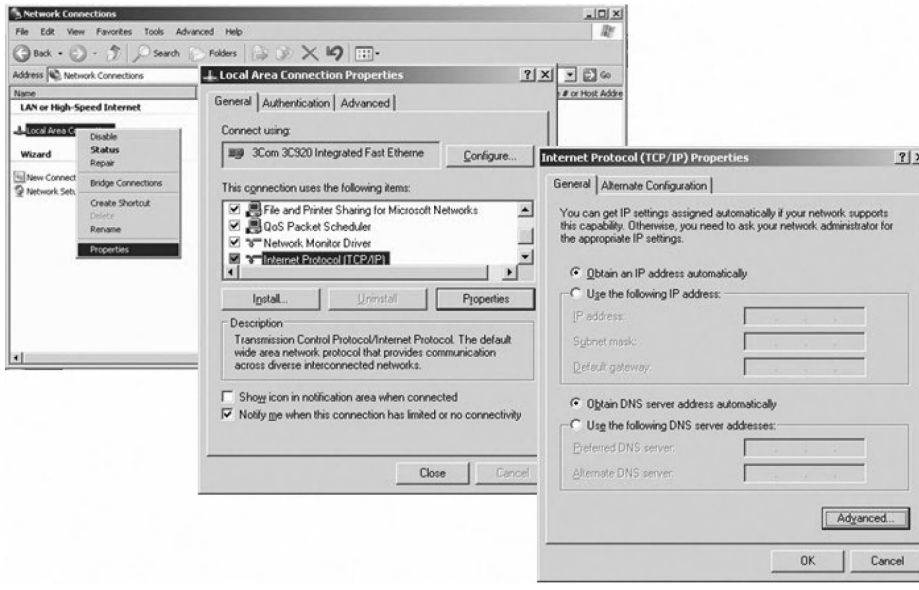

Figure 4-7: Microsoft windows configuration of IP address DNS servers to query

Microsoft Windows enables entry of multiple DNS servers to query within its graphical interface. Notice there are two entries in the "brute force" method shown on the screen on the right of Figure 4-7, one for preferred and another for alternate. Clicking the Advanced tab enables entry of more than two and in particular order. You should have *at least* two DNS servers configured within the resolver, so in the event a DNS server is out of service, the resolver will automatically query an alternate server. If the "Obtain DNS server address automatically" radio button is selected, as shown in Figure 4-7, the resolver will obtain a list of DNS servers via DHCP.

On Unix- or Linux-based systems, the `/etc/resolv.conf` file can be edited with resolver configuration information. The key parameter in this file is one or more `nameserver` statements pointing to DNS servers, but a number of options and additional directives enable further configuration refinement.

DNS UPDATE

DNS Update messages enable a client, DHCP server, or other source to perform an update (add, modify, or delete) of one or more resource records within a zone. The DNS update message enables specification of a prerequisite that defines required conditions attached to the update. Prerequisite conditions include whether a set of resource records for a given lookup value (e.g., hostname) exists and matches (or doesn't exist) in the zone being updated. Such prerequisites can be used to minimize the overwriting of a given resource record by a dynamic client. For example, if "host-A" has a resource record within a given zone, a prerequisite can be used to prevent another host from using the name "host-A."

When the ISC DHCP server performs dynamic updates of DNS data upon assigning an IP address, it can perform a DNS update with the hostname-IP address record (A or AAAA) along with an additional resource record, identifying the particular client to which the hostname is associated. This additional record, a DHCP Identifier (DHCID) or Text (TXT) record, includes a hash of the host's hardware address receiving the IP address to uniquely identify the host. The prerequisite condition for updating the address record provides a means to ensure that only the original holder of this A or AAAA record can modify it, minimizing naming duplication.

RESOURCE RECORDS

This chapter has covered the organization of DNS data and traversal of the domain tree. Once we've navigated the tree and located a DNS server authoritative for the information for a given domain, resource records configured within the domain provide the means to map the question to an answer. The type of resource record defines the desired result type; for example, the A resource record type will provide an IPv4 address as an answer while the AAAA type will

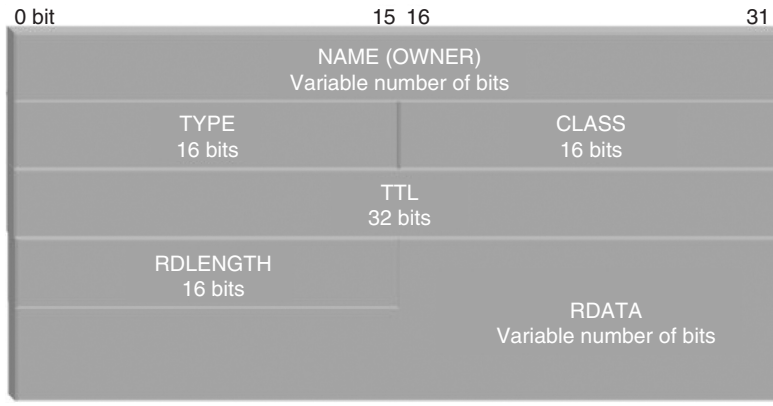

Figure 4-8: DNS resource record wire format (23)

provide an IPv6 address. The answer may be "the final answer" or information that can be used to obtain the desired answer via additional queries or other means. When responding to a query for information, a DNS server will place the resource record information in the Answer section of a DNS message. The "on-the-wire format" dictated by the DNS protocol is illustrated in Figure 4-8.

When representing resource records in zone files, all of these fields may be entered except the RDLength field, which is inserted when the resource record information is placed in a DNS message by the DNS server. The textual representation of a resource record generally follows a common convention shown below. Most resource records are defined with the following general fields, though many have subfields within the RData field.

Owner	Time to Live	Class	Type	RData

- **Owner** (Name)—This field contains the information being queried.
- **Time to Live**—The number of seconds for which the information contained in this resource record is valid for servers and resolvers caching this information. After the TTL expires, the resource record information must be removed from the name server and resolver cache. The TTL can be specified on a per resource record basis or if omitted, a zone level default TTL value is used.
- **Class**—The Class of the resource record, usually IN for Internet.
- **Type**—The type of resource record corresponding to the type of information being sought.
- **RData**—The "record data" or answer portion corresponding to the information being sought by matching the Owner (Name), class, and type field contents.

Please consult Appendix C for a listing of currently defined resource record types and the applications they support.

DNS SERVER DEPLOYMENT CONSIDERATIONS

Beyond the availability and performance considerations key to designing a DHCP server deployment, additional design attributes required for DNS relate to deploying sets of DNS servers in support of particular *roles* in resolving domain names. This relates primarily to deploying DNS servers to address the following queries for your name space. Each type of query is labeled by letter in Figure 4-9 for reference.

A. Queries from internal clients for internal hosts
B. Queries from internal clients for external or Internet hosts
C. Queries from external or Internet clients for internal hosts

Queries from internal hosts for destinations within the internal name space, for example, email servers, and intranet servers, printers, should be resolved authoritatively by internal DNS servers configured with internal zone and resource record information. Queries from internal hosts for external destinations, like web sites, would also be directed to internal DNS servers. Internal resolvers don't distinguish and simply issue queries to one set of DNS servers. These servers, performing recursion on behalf of internal clients may query Internet root servers directly using the resolution escalation path we discussed earlier.

Alternatively, these internal DNS servers may forward queries destined for external resolution to a discrete set of *caching servers* as shown in Figure 4-9. These servers are called caching servers because they are the funnel points through which all external resolutions flow, and they cache this resolution information for use in resolving future queries for similar information. Deployment

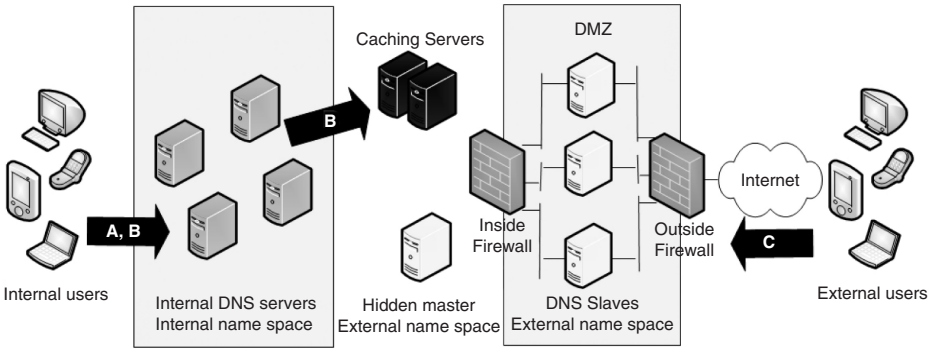

Figure 4-9: Basic DNS deployment

of caching servers decreases the number of servers querying the Internet, enabling a degree of firewall traffic control, and improves resolution time for internal clients. As the caching servers build up an extensive cache, resolution data is more likely to be readily available in cache, obviating the need for an external query. As we'll discuss in Chapter 9, a common DNS threat is cache poisoning so steps must be taken to mitigate these threats on caching servers, whether deployed as dedicated servers per Figure 4-9 or whether internal name space servers query Internet servers directly.

External DNS servers should be configured with resolution information for hosts that are accessible from the Internet, for example, your web, email, and other public applications. DNS queries from the Internet should not be permitted beyond the DMZ. The slave DNS servers in the DMZ should be authoritative for all externally published destinations. The master server should be "hidden" and inside the interior firewall (not queried by Internet servers). The master server is hidden by not publishing it among the NS records in the parent zone. Recall that delegation down the domain tree is facilitated by configuring NS records corresponding to authoritative DNS servers serving the delegated zone; by publishing only the slave DNS server NS records, the master may effectively be hidden. The additional step of changing the Master Name (MNAME) field within the zone's SOA record should also be performed for such static external zones to more completely "hide" the master server.

The external servers must be configured to ignore recursive queries; they should respond only to other DNS servers, not end user resolvers. These servers should also be secured to minimize vulnerability, including possibly signing your zone information via DNSSEC. DNS security including DNSSEC is discussed in Chapter 9. As we discussed, internal servers should be secured as well, though signing internal zones is generally of lesser concern.

This separation of servers and zone information is highly recommended, though there are many other role types that are possible, as we discuss in *IP Address Management Principles and Practice* (24).

Some general DNS server deployment principles to keep in mind include the following.

- Deploy a master and at least two slave servers as authoritative for any given zone or set of zones.
- For Microsoft Windows DNS server deployments, deploy multiiple domain controllers for each domain.
- Deploy servers authoritative for the same zones on different subnets and, ideally, different locations for site-diverse high availability. If a subnet is unreachable, resolvers will be able to query a server on a different subnet.
- Deploy authoritative servers "close" to clients/resolvers for better performance and less network overhead. For external servers, deploy close to Internet connections; for internal servers, deploy nearer to higher density employee areas.

- For master servers, consider deploying a redundant hardware solution.
- Separate DNS servers should be deployed to handle external queries vs. internal queries. By external queries, we mean those queries origininating from outside the organization, for example, the Internet. Internal queries are those originating from within the organization.
- Consider separating servers responsible for resolving authoritative data fom those responsible for resolving recursive queries on behalf of stub resolvers.

More details and deployment scenarios are available in *IP Address Management Principles and Practice* (24).

KEY DNS MANAGEMENT CHALLENGES

DNS management challenges are summarized below. We'll discuss techniques for addressing these challenges beginning in the next chapter.

- Identify and obtain a public domain name (or perhaps several) from your ISP or domain name registrar.
- Define internal name space including subdomains and subzones; identify administrators and DNS servers to support subzone delegations.
- Enumerate hosts and corresponding resource record information to expose externally for resolution from external or Internet queries; configure corresponding external DNS servers.
- Enumerate hosts and corresponding resource record information to expose internally for resolution from internal queries; configure corresponding internal DNS servers.
- Define DNS update mechanisms and configure servers accordingly; in general, external DNS servers should not support dynamic updates.
- Configure DNS server parameters based on the server vendor attributes for security, scalability, master/slave configurations, etc.
- Monitor DNS servers for status and query rates to detect outages, denial of service attacks, or query capacity issues.

5

IP ADDRESS MANAGEMENT OVERVIEW

KEY ELEMENTS OF IP ADDRESS MANAGEMENT

The adept practice of IP address management consists of effectively managing the three fundamental aspects of IP networking we covered in the prior three chapters:

1. IP address subnetting and tracking (IPv4/IPv6 addressing)—Maintenance of a cohesive IP address allocation plan that promotes route summarization, maintains accurate IP address inventory, and provides an automated individual IP address assignment and tracking mechanism. This also requires tracking individual IP address assignments on each subnet, some assigned by hard-coding, for example, routers or servers, and others assigned dynamically, such as, laptops and VoIP phones.
2. DHCP—Automated IP address and parameter assignment relevant to location and device type. This requires tracking address assignments configured on devices and setting aside dynamically allocated address pools. These address pools can be configured on DHCP servers in order to enable devices to request an IP address and receive a location-relevant address in reply.

Introduction to IP Address Management, By Timothy Rooney

3. DNS—Lookup or resolution of host names, for example, www entries to IP addresses. The third key aspect of IP address management deals with simplifying IP communications for humans through the use of names, not IP addresses, to establish IP communications. Because DNS enables this critical mapping of host names to IP addresses and vice versa to simplify human use of the Internet, it is considered a key element of the practice of IP address management. After all, the mapped IP addresses must be consistent with the IP address plan.

Given the dynamic nature of IP address assignment for mobile devices, it follows that updating the name for each dynamic address assignment is generally necessary. This enables those seeking to connect to my laptop to connect using its hostname, say, *laptop-abc*, provided that DNS is updated to translate the name *laptop-abc* to its current IP address. This also enables *laptop-abc* to communicate using certain applications, such as virtual private networks (VPNs), which typically require *laptop-abc* and its IP address to be retrievable from DNS.

Hence, when laptop-abc obtains an IP address from a DHCP server, whose address pools are provisioned based on corresponding subnets defined in the IP address plan, the DHCP server can update a DNS server via dynamic DNS (DDNS). DDNS defines the mechanism for updating DNS upon dynamic address assignment using DNS update messages. While DDNS provides a linkage between DHCP and DNS within the scope of IP management, static IP devices such as routers or servers generally require accessibility by name as well, so from the perspective of associating a host or domain name with an IP address, the linkage to static and dynamic IP address inventory also exists.

Recalling our IP networking overview in Chapter 1, possession of an IP address is a prerequisite for communicating on an IP network. Each IP address must be unique and relevant to each user's location within the network to effectively communicate using IP applications like email and web browsing. Organizations have a finite set of IP addresses for allocation and assignment throughout their networks. Those responsible for operating the IP network need to assure proper allocation of these valuable IP address resources to provide adequate address capacity. By "proper", we mean that the allocation must be large enough to meet the IP address demands of those IP devices at the site, but not excessively large in terms of needlessly reducing the availability of IP addresses for allocation elsewhere in the network.

You may be wondering, sure, it may be complicated to get the network set up in terms of IP address allocations, assignments, and DHCP and DNS server configurations; but once it's set up, you're done, right? Depending on your business, this is rarely the case. Most IP networks grow and shrink with the demands of the business: new stores are opened, offices are closed or moved, companies are acquired, and new devices and device types need IP addresses. These and other less substantial changes impacting the IP network can have major repercussions on the existing IP address plan. As the number of users and IP addresses increases, along with the number of subnets or sites, the task of tracking and

managing IP address allocations, individual assignments, and associated DNS and DHCP server configurations grows in complexity. The practice of IP address management seeks to bring order and discipline to managing this complexity. Before discussing this practice in detail, let's reiterate its importance by first examining the network and user applications enabled by IPAM technologies.

APPLICATIONS OF ADDRESS MANAGEMENT TECHNOLOGIES

In reviewing these applications, we can more clearly see the importance of effectively configuring and managing DHCP and DNS servers performing their respective roles in conjunction with an overarching IP address plan.

DHCP-Driven Applications

Automated Address Assignment. The most fundamental application for DHCP is automated address assignment. We take for granted that DHCP works when we connect to an IP network. In Chapter 1, we discussed the application of DHCP services for automated IP address assignment. This automation streamlines the end user experience and requires no manual involvement from IT or Operations staff. End users need not call the help desk to obtain and enter IP addresses into their devices. Customer satisfaction with no additional cost! DHCP not only automates IP address assignments, it also enables network administrators to retain control of what IP addresses may be assigned to certain clients, even up to denying access, as we'll discuss in Chapter 9.

Service Class–Based Address and Parameter Assignment. The key component in supporting various services and applications with DHCP is the ability of the DHCP server to classify a device requesting an address and to supply an appropriate IP address and additional configuration information. This classification of clients into *client classes* enables the DHCP administrator to identify a parameter value within a particular DHCP packet field or option to match on a per-DHCP transaction basis. When a client is classified, the DHCP server may then determine:

- The IP address pool from which to assign an address to the client (if any)
- What additional option parameter values to provide to the client

Leading DHCP reference implementations from the Internet Systems Consortium (ISC) and Microsoft support both the vendor class identifier (option 60 for IPv4 and 16 for IPv6) and the user class identifier (option 77 for IPv4 and 15 for IPv6) options as class parameters. When these options are included in the Discover or Solicit packet, the server can use this information to identify the type of device that is requesting its configuration.

The most common class-based example application is that of multimedia device initialization, such as VoIP devices. In many cases, the multimedia vendor manufacturer encodes a given vendor class identifier option value. VoIP phones are a prime example of this application. Most vendors supply a model number and/or manufacturer name within the vendor class identifier option field. Configuring the DHCP server to recognize this particular value enables the server to supply particular DHCP options required by the client and to assign an IP address from a specific address pool. Other application-specific DHCP clients requiring particular configuration parameters may likewise be identified and configured based on the value of the vendor class option.

The user class identifier option is another candidate for determining client configuration. However, since user class identifier is typically end user settable, it is considered less reliable. Should a user outside of the user class group discover the value or setting, he or she could program his or her device accordingly. For example, using Microsoft's ipconfig utility with the /setclassid argument, it's quite easy to set the value of the user class identifier option.

The ISC DHCP server supports filtering on additional class parameters, in fact up to any packet parameter from MAC address, a subset of the MAC address, or any option value. This is convenient if a given MAC address (interface card) or MAC prefix (manufacturer) needs to be filtered and assigned certain parameters.

Broadband Subscriber Provisioning. The cable industry defined a standard for data transmission over cable, referred to as Data Over Cable Service Interface Specifications (DOCSIS®). The DOCSIS specifications, authored by CableLabs, require the use of DHCP for provisioning of customer premises equipment (CPE), such as cable modems and telephony devices. A cable operator that offers cable data or broadband Internet services must deploy DHCP servers to support the CPE provisioning process. Other broadband technologies such as digital subscriber line (DSL) and fiber may also use DHCP or Bootp, though other techniques such as PPP (Point to Point Protocol) are also used by these technologies.

The incorporation of DHCP into the provisioning process affords the broadband operator control over IP address assignments and capacity, as well as additional configuration parameters used by CPE for initialization. DHCP can also be used to assign IP addresses from address pools corresponding to various service levels based upon the customer's subscription. Assigning an address from a given pool requires the network routing infrastructure be configured to route IP packets with such addresses only to certain networks, permit access to certain destinations, and treat packets with corresponding levels of priority and queuing.

Let's consider an example to illustrate these concepts. In Figure 5-1, three subscribers are connected to a common broadband gateway via the broadband access network. The figure depicts each subscriber with various levels of service as indicated by different shading, connected to individual ports on the broadband

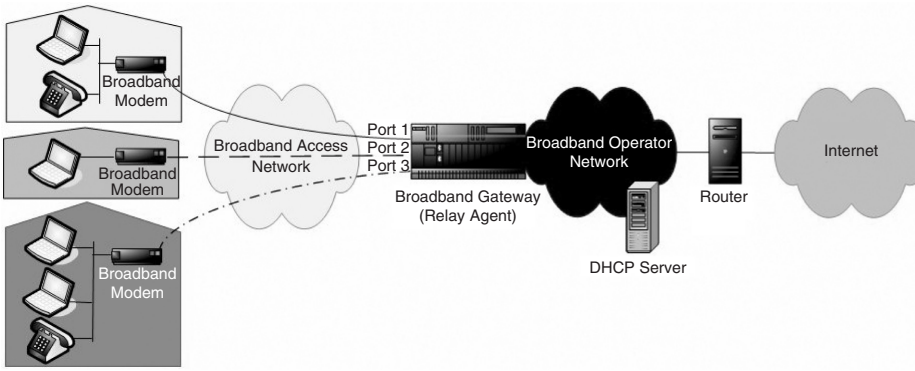

Figure 5-1: Broadband access scenario

gateway. Depending on the broadband access technology, these may be physical ports or logical ports for shared network access.

Regardless of the broadband access technology, service providers using DHCP need to base address and parameter assignment on known or trusted information. Instead of relying on the client hardware address field of the DHCP packet, which can be spoofed, service providers rely on information from the broadband gateway, which resides in the service provider's network and is considered trustworthy.

The broadband gateway, acting as a DHCP relay agent, unicasts the DHCP packet to the appropriate DHCP server(s), inserting the GIAddr field within the DHCP packet header. The gateway also inserts the relay agent information option parameter as the last option before the null option terminator. The relay agent information option provides information such as the subscriber device hardware address or subscriber virtual circuit identifier to help the DHCP server identify the subscriber client that issued the DHCPDISCOVER packet.

This enables the DHCP server to provide, based on its configuration, an appropriate number of IP addresses and/or option parameters for a given subscriber, as identified by the relay agent information option. This option in IPv4 (option 82) is a single option comprised of one or more suboptions, while with DHCPv6, two analogous options have been defined: Option_remote_id and Option_subscriber_id.

Related Lease Assignment or Limitation Applications. The use of lease limiting and parameter setting based on relay agent information is not exclusive to broadband environments. Other applications may use the same technique assuming relay agents support the population of the relay agent information option. In such cases, use of the ISC DHCP server enables address and parameter assignment as well as lease limiting based on defined classes and relay agent information parameters. This technique may be employed to throttle address assignments on certain subnets or to provide configuration parameters to devices in factory or similar applications.

Pre-Boot Execution Environment (PXE) Clients. PXE ("Pixie") clients are devices that boot up relying on network servers instead of a co-resident hard disk. Such diskless servers and other such devices typically use DHCP to obtain an IP address as well as boot parameters including boot server addresses and boot file names. DHCP provides a convenient mechanism to initialize these devices without manual intervention. Historically, DHCP servers had to be configured with the MAC address of each PXE client to provide configuration information specific to the device, even if multiple PXE clients of the same "type" could leverage exactly the same boot information.

RFC 4578 (25) is an informational RFC defining a means whereby a PXE client can identify its type or architecture. This information can be used by the DHCP server to identify and provide appropriate device initialization parameters. The DHCP server would need to be configured to match on particular client-provided PXE option values, then map these results to a corresponding set of configuration parameters or options to return to the client. Naturally this is accomplished using processing similar to that for client classes.

PPP/Radius Environments. The RADIUS (Remote Access Dial In User Service) protocol provides a means to authenticate end users attempting to connect to a network. Radius is a vital component of 802.1X, a popular layer 2 media access control protocol proposed within leading network admission control (NAC) offerings. Radius also plays a role at layer 3, especially when used in conjunction with point-to-point protocol (PPP) connections, commonly used with dial-up or DSL connections.

When operating at layer 3, some Radius servers can be configured to assign IP addresses to each client at the other end of the PPP connection. This address assignment process can be performed by configuring an address pool directly on the server or by configuring the Radius server to obtain an address via a DHCP server. In the latter scenario, the Radius server functions as a DHCP proxy on behalf of the client. The Radius server initiates the DHCP D-O-R-A process, issuing a DHCPDISCOVER packet. One caveat with this approach is that the Radius server must generate a hardware address or client identifier on behalf of each client to uniquely identify each. Otherwise, by using the Radius server's hardware address, the DHCP server would assume that the same client is continually rebooting and assigns the same IP address on all requests! The Radius server can spoof the client's hardware address using an internal mechanism but needs to map the derived address to the end client to process subsequent lease transactions like Renews and Releases.

Mobile IP. Mobile IP provides a mechanism for an IP device to retain network connectivity while moving about a local or remote IP network. This movement may occur during a communications session, not only when conducting then terminating a session, for example, from a headquarters meeting to opening a new session at a branch office. The mobile device has a home address, corresponding to its home network, as well as a care-of address, which is obtained on

the serving network depending on where the mobile device is presently connected for location-relevance. For example, if I power up my personal digital assistant (PDA) device while out of town, I may obtain wireless service from a different service provider than that which I normally use when "at home." As long as my home provider has a service agreement with the provider I'm visiting, I should be able to obtain an address manually, via DHCP, or via autoconfiguration.

IP mobility support differs somewhat between IPv4 and IPv6 but both protocols leverage the concept of a mobile node possessing a home address, the node's address on the "home" network and a care-of address, its address on the visited network. While not strictly a DHCP "application," we mention it here as an area for consideration with respect to address allocation and assignment strategies, not to mention access security for visiting nodes on your network, as we'll discuss in Chapter 9.

DNS-Driven Applications

Information Translation. DNS inherently lends itself well to "translating" a given piece of information into another related piece of information. This resolution process is the very reason for the invention of DNS, and it has been extended beyond resolving hostnames into IP addresses and vice versa to support a broad variety of applications. Virtually any service or application that requires translation of one form of information into another can leverage DNS.

Each resource record configured in DNS enables this lookup function, returning a resolution answer for a given query. The query indicates what "name" is sought and what "type" of information about the sought name is requested. The "class" is also included, though Internet (IN) is the most common class. The DNS server parses the query from the DNS message, seeking a match within the respective zone file for the queried name, class, and type. In fact, each resource record has a Name (aka Owner), Class, and Type field. Upon matching these three fields, the corresponding Rdata field contains the answer to the query!

The resource record type defines the data type and format of the question (owner/name field) and corresponding answer (Record data or Rdata field). In some instances, multiple resource records may match the queried name, type, and class. In such cases, the set of all matching records, called a *Resource Record Set (RRSet)*, is returned in the Answer section of the response message.

As new information translation applications are developed, new resource record types can be defined through the Internet standardization process. But not all new applications require new resource record types to enable definition of application-specific information. The various forms of information that are stored in DNS along with the applications they support are summarized in Appendix C.

Telephone Number Resolution. When we think of DNS resolution, we commonly think of mapping domain names to IP addresses. DNS can also be used to map telephone numbers into IP addresses, which is useful for VoIP

applications or related telephony over IP applications (e.g., Fax, etc.). The ENUM (E.164 telephone number mapping) service has been defined to support such resolution. ENUM supports the mapping of telephone numbers, in ITU E.164 format, into uniform resource identifiers (URIs). A URI is an Internet identifier consisting of a uniform resource name (URN) and a uniform resource locator (URL). A simple example: for URL `http://ipamworldwide.com`, and URN `file.txt`, the corresponding URI is `http://ipamworldwide.com/file.txt`. This mapping of E.164 telephone numbers into URIs is performed primarily using the Naming Authority Pointer (NAPTR) resource record type.

Note that most enterprise IP PBX systems provide their own directories to map intra-PBX phone numbers to destination phones' IP addresses, so ENUM is not commonly implemented in such environments. However, VoIP service providers have a vested interest in assuring calls remain on their or their partners' IP and access networks to the maximum extent, to reduce call handling costs paid to nonpartner network providers, or worse, competitors. And ENUM is key to enabling such call routing by virtue of telephone number mapping or resolution. That is not to say that you won't see ENUM within enterprise networks. ENUM provides resolution to multiple destinations with preference settings, which may find use within reachability or contact management–type applications.

As just mentioned, the NAPTR resource record provides translation of a string or telephone number information into destination URIs. Currently defined in RFC 3403 (26), NAPTR records were initially defined to provide a means to iteratively resolve an arbitrary string into a URI for the Dynamic Delegation Discovery System (DDDS). Some background on DDDS is provided in RFC 3402 (27), but it initially stemmed from the desire to define a resolution process that could enter with a resource name (e.g., a particular application or piece of data), which in itself contains no network location information, and resolve it to a destination resource identifier by applying a series of iterative rules from a database. This separation of the specification of the resource name from the process to locate or resolve it facilitates making of changes and redelegations of resources without impacting the end user application's naming convention.

This effort expanded beyond resolving resource names to supporting resolution of generic lookup strings, and evolved into the DDDS, using DNS as one form of the rules database. The NAPTR record enables the specification of such rules within DNS, sometimes using multiple NAPTR records to fully complete the resolution process. Each NAPTR record translates a given entry string, that is, a valid DNS domain name, into a rule that can be applied to the string to derive the next string to lookup. This process iterates until a terminal rule is reached and the final result is returned to the requesting application.

A NAPTR record can be used to lookup a destination telephone number, and resolve the number to a destination, for example, a Session Initiation Protocol (SIP) server, email address, or other URI-formatted destination. NAPTR records also support the ability to define regular expressions, which supply logical rules as "next steps" for the resolver to locate the intended destination.

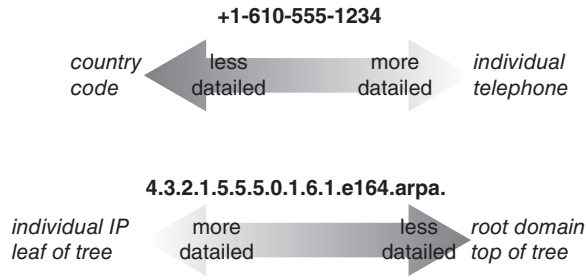

Figure 5-2: Telephone number mapping to domain structure

E.164 is an International Telecommunications Union (ITU) standard for formatting telephone numbers. "Fully qualified" telephone numbers, meaning they are globally unique given the country code prefix followed by a country-specific telephone number format, are represented with a plus sign prefix, such as +16105551234. Much like reverse domains for IP addresses, formatting a telephone number requires a similar convention of reading the resource record from left to right as more specific to less specific. This convention requires reversal of the fully qualified telephone number (dropping the plus sign) and separating each digit with "dots" as illustrated in Figure 5-2.

Note the use of the .arpa top level domain. Similar to ip6.arpa and in-addr.arpa domain structures, the e164.arpa domain is a "reverse" domain in that it enables lookup of a structured numerical value, a phone number. Like other .arpa lookups, the domain structure is organized top-to-bottom as generalized-to-specific, or country code-to-telephone line number. Thus, the fully formatted E.164 telephone number is reversed, each digit is separated with dots, and the e164.arpa. domain suffix is appended.

This structure lends itself well to segmentation of telephone number space. For example, the domain 1.e164.arpa refers to all country code 1 telephone numbers and could be delegated to such a number authority. Likewise 44.e164.arpa could be delegated to the U.K. telephone numbering authority. Within each of these domains, further delegation may be accomplished in accordance with the numbering plan for the country. For example, within the U.S., an area code represents the next logical administrative delegation point, followed by exchange. Thus the administrators for 1.e164.arpa may delegate the 0.1.6.1.e164.arpa zone to the numbering administrator for the 610 area code, who may in turn delegate 5.5.5.0.1.6.1.e164.arpa to those responsible for the 555 exchange within the 610 area code.

Network Services Location. IP devices booting on a network often need to find specific services for device initialization. While DHCP provides some level of service location via specification of certain option values such as TFTP server IP addresses, DNS provides a services location mechanism using the services location resource record type (SRV). The SRV record provides a means for

non-DHCP clients or for clients seeking services after initialization to locate servers providing requested services.

If you've worked with Microsoft clients and domain controllers since the introduction of Windows 2000, you're probably very familiar with SRV records. When Windows domain controllers boot up, they perform a dynamic DNS (DDNS) update for their A and SRV records, enabling them to effectively advertise services availability. These records are also easily recognized by underscores within the owner field. The SRV record owner field is comprised of a concatenation of a particular service, which is available via a particular protocol (TCP or UDP), for a given domain. The service name is prefixed with an underscore, as is the protocol value. The underscores were added to eliminate collisions with valid domain names. While technically not a valid host domain name character per the DNS RFC 1035 (23), Microsoft and BIND servers can be configured to tolerate the underscore character via the check-names option parameter. This is a common example of the use of SRV records, though SRV records are certainly not limited to Windows applications.

Email and Anti-Spam Management. Spam email, or unsolicited bulk email, has been a nuisance since the dawn of the Internet, though in the early days it was highly frowned upon. In fact, a husband and wife team from Arizona was taken to court in the mid-1990s under a spam complaint. Alas, had only the precedent been set then! Nevertheless, with the explosive growth of the Internet, the volume of spam emails has seemingly grown even faster. A variety of techniques exist to combat spam, many of which involve the use of DNS. To understand how DNS can help reduce spam, we'll first look at the anatomy of an email transmission including the role of DNS in email delivery, then review the use of DNS in various anti-spamming solutions.

An email typically originates from one person and is sent to one or more recipients. Each email address is formatted as a mailbox@maildomain. The mailbox commonly refers to the name of the person or owner of a mailbox or email account, while the maildomain, typically the company or Internet provider name, is the destination domain for delivery to the corresponding mailbox or mail exchanger. Emails are composed using an email client, such as Microsoft Outlook and Eudora, or web-based clients like Yahoo and Google. Regardless, when sent by the originator, the client connects to a Simple Mail Transfer Protocol (SMTP) server to send the email. The SMTP server must resolve the maildomain to an IP address for transmission of the message. Naturally this is done using DNS with a lookup for the Mail Exchanger (MX) record type, as well as the corresponding A or AAAA record types.

Recipient mail servers may also query DNS to verify the validity of the sender as an anti-spam check. Lookups of known spammers in block lists or the converse provides the server information in identifying SPAM email. DNS queries to the email sender's originating domain and name servers may also be used to determine whether information within the received email matches the policies of the sending domain. Such policies as Sender Policy Framework (SPF) or Sender

ID can be encoded in SPF or TXT resource records. Domain Keys Identified Mail (DKIM) also uses DNS to validate the signature on a received email with the corresponding sender's signing policies and domain keys. If you're interested in more detail, please consult *IP Address Management Principles and Practice*.

DNS provides numerous other lookup functions for such things as geographic locations, SSH fingerprints, host or domain aliases, and more. The resource record table in Appendix C summarizes currently defined resource record types.

POTENTIAL IMPACTS OF INADEQUATE IP ADDRESS MANAGEMENT

Having summarized some key applications of IPAM technologies, let's consider the impacts of failure to effectively manage them. To illustrate this, let's consider the potential impacts of inadequately managing this complexity. Network managers responsible for maintaining accessibility to an IP network need to keep this information organized, accurate, and secure. The ramifications of not properly managing IP addresses include the inability of users to connect to the network due to:

- Router subnet configuration misalignments with the IP topology and plan
- Inaccurate IP inventory inhibiting troubleshooting of issues related to particular IP addresses
- Duplicate assignment of IP addresses to multiple devices
- Improper or inaccurate configuration of DHCP servers resulting in an inability to properly assign addresses dynamically
- Improper or inaccurate configuration of DNS services resulting in an inability to reach web servers, email servers, or other IP application servers by name
- Lack of availability of IP addresses due to fully assigned address pools
- Misalignments with actual IP network assignments due to "localized" address assignment ("Oh, I'll just connect my own printer to this Ethernet cable and assign it address X.")
- Lack of history tracking inhibiting change control, auditing, reporting and troubleshooting

IP ADDRESS MANAGEMENT AS NETWORK MANAGEMENT

The discipline of network management has long offered technical and business benefits to organizations with the centralization of the monitoring, control, and provisioning of distributed network elements such as routers and application or services databases. These benefits include holistic management of the entire network from a centralized location where appropriate resources are concentrated for troubleshooting, resolution, and escalation. The centralized "top

down" approach also lends itself well to supporting structured network change control procedures.

It's a small leap to consider DNS and DHCP servers as network elements, as they provide critical IP services to clients on an IP network. While not in-band or on the data path for user IP traffic like traditional network elements, they provide necessary services required to make such in-band data paths possible and usable. From a telephony intelligent network analogy, DNS and DHCP are akin to Network Control Points in providing look-up and addressing information. So it follows that centralized management of these servers is equally wise and beneficial. A blip on a network manager's screen could ultimately have been caused by an IP address misconfiguration, so equipping him or her with the corresponding capabilities for monitoring and control of DHCP and DNS services in the network can reduce overall troubleshooting and resolution time. We'll focus the next chapter on mapping IPAM functions to the standard FCAPS model for network management. FCAPS is defined in ITU standard M.3400 as part of the Telecommunications Management Network (TMN) framework for managing data networks.

IP ADDRESS MANAGEMENT BUSINESS BENEFITS

We've dedicated Chapter 10 of this book to helping you quantify IPAM business benefits for your organization, but we provide a brief overview here. Many of the business benefits of IP address management relate to *reducing time* and *reducing errors* in performing certain tasks. If you run your network like a service provider, whether offering commercial services to customers or providing services to internal or external constituents, time reduction in provisioning and trouble resolution can translate into actual dollars saved if service level agreements (SLAs) are in place, or to customers saved by promoting constituent goodwill through rapid provisioning and resolution of issues.

Other business benefits relate to readiness and accuracy of information. Regulatory requirements may dictate tracking of IP address assignments over time. Network audits may require tracking of who made certain changes in key network elements including DHCP and DNS servers. And reports for external authorities such as industry oversight organizations or Internet registries, for example, may be required on an ad hoc or periodic basis. And certainly having this required information readily available, including IP address allocations and assignments as well as DHCP/DNS server configurations, can reduce time required to satisfy these requirements as well as reduce troubleshooting time. These and other key business benefits are summarized below.

Reducing Outages

The fact that the services managed under the umbrella of "IP address management" are so fundamental to the very operation of the IP network for all of these IP applications, yet are provided in an automated manner that yields little visibil-

ity, leads many users and IP planners to take them for granted. Of course, once an outage occurs due to IP address misassignments, or DHCP to DNS services failure, the visibility can become blinding. If an end user cannot obtain a relevant IP address that is unique, or cannot resolve host names, he or she will likely consider the network down and call the help desk. This event effectively is an outage since there is at least one end user attempting to use the network that cannot.

A considerable business benefit of effective IPAM is the reduction of these outages. Forrester Research (28) estimated that up to 15 percent of network outages are related to DHCP and DNS issues. Hence, employing IPAM discipline can help reduce network outages, thereby reducing fire-fighting and associated costs while improving end user satisfaction with increased network uptime.

Rapid Trouble Resolution

Should a site or portion of the network become unreachable for direct diagnosis, a centralized IP database may prove indispensible in identifiying occupied and available IP addresses, perhaps to configure a "back door" to the isolated site. The very moment the information is needed may be when it may not be directly attainable from the source; hence a centralized IP address database is critical to facilitating rapid trouble resolution.

Streamlining Configuration

Another key benefit is automation. The close inter-relationship among IP inventory, DHCP, and DNS affords the opportunity to leverage common information across these key elements. This information sharing can reduce data entry requirements by multiple users, perhaps in multiple systems. Automation reduces manual effort and associated data entry errors, and can reduce the time required to perform certain tasks. Automation benefits may also extend beyond IPAM functions, as many IPAM solutions can automate across a broader spectrum as we'll discuss in Chapter 7.

Accurate Inventory and Reporting

Maintaining a centralized repository of IP address information eases the process of tracking inventory for configuration, management, and reporting. Of course, accuracy of the information is key, so mechanisms to periodically reconcile inventory information with the network are recommended. Audit tracking concerning who had which IP address and who made particular network changes is also important to meet accountability and possibly regulatory requirements.

Deploying Additional IP Services

Historically, most organizations implemented IP in order to support an external web site. IP implementations expanded to internal IP network deployments in the form of intranets and ultimately to support other IP data applications. Today,

with the proliferation of wireless networks, private and public, as well as voice and multimedia over IP technologies, the reliance on your IP network is growing ever more prominent. Given these and other advanced IP services rollouts, a solid foundation of accurate and effective IP address inventory and planning as well as accurate DHCP and DNS server configuration is paramount.

Distributing Administration

While centralized management is a common network management approach, the ability to delegate responsibility for certain functions and/or subpartitions of the network is important as well. Enabling other administrators to manage subsets of the network enables the centralized team to handle issues requiring a higher level of expertise such as troubleshooting escalation. Even empowering end users for simple tasks such as requesting an IP address can reduce staff time requirements and improve responsiveness through self-service.

Enhanced Security

Implementing IP address management systems and procedures can enable you to maintain and secure IP network information, and related user information. Auditing of network access attempts and access control mechanisms also contribute to your overall network security plan. Securing access to your IP network as well as securing DNS and DHCP information and communications need also be considered. We've devoted Chapter 9 to the topic of security.

IPv6 Deployment

While many have reacted to recent announcements of impending IPv4 address depletion with disinterest if not apathy, certain industry segments are beginning to implement IPv6. Large organizations seeking to IP-enable everything from appliances, RFID-tagged commodities, wireless devices, sensors, and so on are also making similar plans. As demand for IP addresses continues to grow, a move to IPv6 will ultimately become a necessity for many organizations, especially those serving as Internet Service Providers (ISPs). Every Regional Internet Registry (RIR) has issued notifications to the Intenet community at large that IPv4 space availability is limited and will be exhausted within "a few years." RIRs are responsible for IP address allocation to ISPs, who in turn allocate space to enterprises, service providers, and any organization requiring IP address space. Ultimately, this exhaustion will impact organizations requiring public IP address space.

Even if your organization currently has more than ample public and private IPv4 address space for the foreseeable future, consideration of IPv6 is probably worthwhile. Such consideration should include planning for IPv6 implementation on external resources such as web and email servers. After IPv4 space has been depleted, organizations requiring address space will only be able to obtain IPv6 space, thereby enabling them to communicate on the Internet via IPv6 only. To

enable these organizations to access web and email services on your site, implementation of IPv6, at least externally, will be required. An IP address management solution that incorporates IPv6 capabilities in addition to IPv4 is an important consideration for such organizations. We'll discuss IPv6 implementation in Chapter 8.

COMMON APPROACHES AND IPAM EVOLUTION

Early History

With the growth in prominence of the use of TCP/IP within enterprise networks and for service provider Internet offerings starting in the mid-1990s, organizations initially managed the three cornerstones of IP management independently. Smaller organizations maintained a paper log, spreadsheet, or in-house database for tracking IP address space and subnet assignments. DHCP and DNS configuration files for the few DHCP and DNS servers operating on the network were generally configured manually using text editors. Larger organizations built home-grown systems or procured commercial software solutions for all or portions of these three areas to provide some degree of automation and consistency among these key areas. The focus historically had been on simply providing a repository of information for IP address tracking and usually for some level of DHCP and DNS configuration and tracking.

Most currently available IP management tools provide a repository and in some cases automated creation of DHCP and DNS configuration information. Indeed there are many tools available in the market today, each providing varying levels of integration and functionality. Thus, some organizations continue to use spreadsheets or databases for IP inventory, while utilizing a software tool with a graphical user interface (GUI) for DHCP and/or DNS management. Even the use of spreadsheets or databases has historically proven adequate for "first generation" monolithic IP networks.

Today's IP Networks and IP Management Challenges

Today, many organizations have deployed VoIP and multimedia services, and more IP service offerings are coming to market. These organizations commonly configure their routers to provide a higher "class of service" to VoIP devices than to data devices such as laptops or PCs. This translates to IP packets for voice traffic garnering higher priority queuing than data traffic, for example. The delay sensitivity requirements of voice communications often necessitate such a configuration. For example, if my email takes an additional 30 seconds to reach my email server, I would rarely even notice. But if portions of my voice conversation are delayed by even half a second, the communication will be rendered totally useless. But how can each router distinguish VoIP vs. data traffic? As we discussed earlier in this chapter, routers can use information in the IP header to differentiate these packets, including in some cases the source IP address field.

Considering use of the source IP address field, the routers must be configured with which source IP addresses represent VoIP phones and which represent data devices. Of course, this maps back directly to the IP addressing plan afforded by the IP inventory and assignment aspect of IP address management.

This address segmentation may require the organization to partition its address space essentially into two parallel address spaces—parallel in that they each service the same physical networks, albeit for different applications. With the relative scarcity of IP address space, most organizations do not have the luxury of deploying a separate overlay or parallel IP address space for VoIP to keep it administratively separate from the data network. They must resort to carving out a portion of the data network for allocation for VoIP devices. This slice of IP address space must then be reflected in the router configurations for packet processing and in each subnet providing VoIP and data services to end users. In addition, VoIP phones utilize DHCP to obtain an IP address and configuration, so configuration of the data and VoIP address pools within DHCP servers is required. Along with the pool configuration, the DHCP server must be configured to recognize a VoIP phone from a data device while deciding what address to assign. This is generally performed using client classes as discussed earlier. Lastly, many organizations desire to have VoIP devices reside on a different DNS domain for administrative purposes. Whether on a separate domain or one common to other devices on the subnet, generally DNS requires updating the relevant name to address information as well.

The upshot of this discussion is that the introduction of a new IP service, VoIP, introduces a substantial ripple effect on how IP address space is managed. It affects IP inventory, in tracking and allocating what amounts to a parallel address space across the organization's sites, as well as DHCP and DNS configurations. While the overall address space within the organization may not have changed, the complexity and granularity of configuration required has essentially doubled. The introduction of additional IP services requiring "special treatment" in address assignment and configuration, such as video conferencing, would stimulate the requirement for a third parallel address space and the further increasing of management complexity.

Beyond the management complexity around the three basic cornerstones of IP address management, an additional set of requirements has arisen. As more IP services are deployed, it's easy to see that the reliance on the IP network grows linearly if not exponentially. With more intra- and even inter-organizational communications relying on the integrity, performance, and availability of the IP network, the need to effectively manage it grows. IP address management is a key element of the overall IP network management strategy. The increasing reliance on the IP network increases the reliance on the IP management processes, which must be reliable, highly available, accurate, and ideally integrated into the broader IP network management processes and systems.

6

IP ADDRESS MANAGEMENT PRACTICES

In the previous chapter, we introduced the concept of "IP address management as network management." Because IP addresses and associated DHCP and DNS functions are so foundational to services and applications running over an IP network, these functions must be prudently managed, much as other critical network infrastructure elements are managed. The most commonly applied network management approach is that of the FCAPS model from a functional perspective and ITIL® from a service management perspective. We'll discuss common IP address management tasks within the context of the FCAPS model, then relate functional mapping of these tasks to ITIL® process areas towards the end of the chapter.

FCAPS SUMMARY

The FCAPS model covers the following key functions within the practice of network management:

- F = Fault Management—Involves monitoring and detection of network faults with the ability to diagnose, isolate, and resolve them. As network

Introduction to IP Address Management, By Timothy Rooney

elements such as routers, servers, and switches are monitored to detect faults or outages, DHCP and DNS services should likewise be monitored. Appropriate workaround mechanisms such as providing for high availability services may also be implemented.

- C = Configuration Management—Entails accurate configuration and backups of network elements, including DHCP and DNS servers. Accurate and timely configuration of network elements reduces provisioning errors and time intervals within change management windows.
- A = Accounting Management—Involves tracking and policing of usage of network resources with respect to business quotas or customer entitlements. Aspects of IP management regarding access control policies, address utilization with respect to business parameters, and monitoring service level agreement (SLA) compliance fall within accounting management.
- P = Performance Management—Deals with tracking performance of network elements and services, along with resource utilization. Tracking of IP address utilization and DHCP/DNS server performance are key requirements for effective IP address management.
- S = Security Management—Includes the securing of information regarding the network and its users, providing access controls, as well as audit logging and security breach detection. Security management for IP address management includes IP address access policies, DNS and DHCP security, and rogue or illicit device detection on the network.

COMMON IP MANAGEMENT TASKS

Leveraging the FCAPS framework, we'll discuss the most common IP management tasks, starting with "configuration," then move on to the other categories. Some functions may likely require the use of multiple management systems depending on your IP management system capabilities. For example, if your IP management system consists of a spreadsheet, you'll need another tool to perform fault management functions. Likewise for commercial IP management systems, varying subsets of functions and tasks will be available natively within the system, while others will require supplemental systems. We'll come back to this topic relating key functions and the associated costs of performing them in Chapter 10.

CONFIGURATION MANAGEMENT

When most people think of IPAM, they primarily consider it a configuration management mechanism. Early IPAM systems in fact focused solely on configuration management, though many have expanded into other aspects of FCAPS. Nevertheless, configuration management is a fundamental IPAM function. In this section, we'll discuss common tasks required when managing IP address

space and DHCP/DNS server configurations. These tasks relate to the day-to-day activities of an IP address planner with respect to moves, adds, and changes for IP addresses, subnets, address pools, domains, and other aspects of DHCP and DNS configuration.

Configuration management within the context of IPAM entails the configuring of DHCP and DNS servers for lease and parameter assignment and name resolution respectively within the scope of the overall address plan. This involves at minimum, configuration of IPAM related information, that is, address pools and associated parameters and DNS configuration and zone files. The configuration process may also entail configuration of high-availablilty deployments and server-level configuration parameters, for server-based or appliance-based DHCP/DNS servers.

The result of the configuration management function is that each of the DHCP and DNS servers within the network is configured with its files or parameters necessary to perform its respective role in the network; e.g., primary DHCP server for a set of address pools, failover DHCP configuration, DNS zones, parameters, and options. From this perspective, the goal is to base each DHCP and DNS server's configuration on its type (for example, ISC or Microsoft), its role in deployment, and the portion of the network it is serving. The portion of the network relates to the association of a set of DNS domains, subnets, and address pools assigned to each server, and should align with the overall plan for address space and DNS domains.

Another function closely tied to IP management is configuration of routers with new, moved, or deleted subnets, as well as relay agent information regarding which DHCP servers to which to relay DHCP packets. Few IPAM systems on the market perform this level of router integration natively. Historically, the IP or server teams were distinct from router teams, so inter-team automation was discouraged; after all, if a router ended up being misconfigured, it would come back to the router team. Some IPAM systems enable automation of this process nonetheless, natively or via an API call, which can "hook" the output of an IPAM system subnet allocation to the input of a router configuration tool. The brute force method likely entails sending an email to the router team after a subnet has been allocated in the IP inventory repository.

Address Allocation Tasks

Address Block Allocation. Starting at the top of the IPAM food chain, allocation of address space is performed hierarchically from the top down. Planning address space allocations must consider business requirements with respect to address capacity for each application and user community at each site from the bottom up. Ultimately, each site will be served from the respective allocation, so capacity planning should incorporate addressing needs at each current and planned future site.

If you don't have time or resources to conduct a full capacity analysis, one rule of thumb for enterprise organizations is to consider the number of

employees at each location and multiply this number by four. This quantity provides a rough estimate and accounts for each employee's devices as well as infrastructure devices like routers and servers. On the other hand, if you have plentiful address space for the size of your organization, you may just want to allocate uniformly.

Once address capacity has been quantified per site, consider the routing topology and how to best model the addressing hierarchy. Using a core-regional-access router topology lends itself to a corresponding mapping of addressing hierarchy. Such a topology features a backbone or core network feeding regional networks, which in turn feed access or local networks. Routers serve as topological interfaces and provide aggregation of networks and routers for their downstream networks. Now integrate the capacity data with the topology to identify the roll-up of address space at each hierarchy level.

Let's illustrate this integration by example. Let's assume our network topology features a core or backbone network serving each global continental subnetwork. The regional level subdivides the continents of North America and Europe. If we assume our organization has 17,000 employees mapping to roughly 75,000 IP addresses, our 10.0.0.0/8 network should provide plenty of capacity with over 16 million IP addresses, let alone our IPv6 ULA space. Thus, our IP planners decide to utilize a *uniform* allocation strategy as much as possible.

When performing top-level allocations such as this, keep in mind not only the capacity required in terms of IP addresses, but the number of subdivisions or hierarchy layers that may be ultimately necessary. As we touched on in the previous chapter, allocation of address space by application may also be necessary to facilitate source address–based routing treatment. In our case, we will define our address hierarchy layers as follows:

- Application
- Continental or backbone router layer
- Regions
- Sites or buildings

Thus our top-level allocation will divide our address space by application. Each application-specific allocation will then be allocated at the core router or continental level, then by region, and finally by office. Since we have four layers of allocation hierarchy, we're going to have to allocate along non-octet boundaries. So let's look at this from both a CIDR network notation and the corresponding binary notation.

The binary representation of this network is shown below. The network portion of the address, whose length is identified by the /8 notation is highlighted as bold italics, while the local portion is in plain text.

Private Network 10.0.0.0/8 ***00001010*** 00000000 00000000 00000000

Before we allocate this space across the organization, let's assume that we're planning to roll out voice over IP in the near future and additional IP services later on. Let's take the next four bits of our network address and allocate equal-sized /12 networks. This would provide $2^{(12-8)} = 16$ potential high level allocations, while providing $2^{(32-12)} > 1$ million IP addresses per allocation. Thus, we'll allocate a /12 each for the infrastructure address space, a /12 for the voice over IP address "subspace," and a /12 for the data subspace. This allocation is illustrated below with the bold italic bits once again representing the network (network + subnet) portion and normally formatted bits representing host bits.

Private Network	10.0.0.0/8	***00001010*** 00000000 00000000 00000000
Infrastructure	10.0.0.0/12	***00001010 0000***0000 00000000 00000000
Voice	10.16.0.0/12	***00001010 0001***0000 00000000 00000000
Data	10.32.0.0/12	***00001010 0010***0000 00000000 00000000

Second-Level Allocation Logic. This initial application level allocation reformats our original monolithic /8 address space into three /12 spaces aligned per application. The decision to use /12 at this level is a trade-off between the number of first-level allocations and the number of addresses available per allocation. If we had decided to allocate /11s, we would have ended up with 8 /11s total, each with over two million IP addresses. In our example, having more top-level blocks available for future allocation was more of a concern than managing capacity per block, given one million IP addresses per /12 application block. If a particular allocation becomes exhausted, we can allocate an additional /12 block.

Block sizing decisions for second and subsequent level allocations depend largely on the relative scarcity of your address space. If address space is plentiful, for example, you run a small to modest-sized IP network, using uniform allocations keeps things simple. If not, for example, you run a large or service provider network, an optimal allocation strategy likely makes more sense. This optimal strategy entails successively halving the address space down to the size required. The key reason for this approach is that it enables you to keep larger blocks of unallocated address space available for subsequent requests and alternative allocations.

If you have ever endured a company merger, you may have encountered a situation like the following, which illustrates the optimal allocation motivation. Let's say your company acquires another company and the network integration strategy requires an allocation of 250,000 IP addresses to the newly acquired division. To minimize confusion (and to exude networking mastery over the rival IT organization), you desire the allocation of a single /14 to support 262,142 addresses.

If we've optimally allocated our address space, we may happen to have a /14 readily available if needed (and be hailed as an IPAM master!). If we had taken a uniform approach of allocating /16s everywhere, we may be lucky to identify four contiguous /16s, which comprises a /14 (lucky amateur!). If we cannot identify four contiguous /16s, we may have to assign four noncontiguous /16s; this adds four times the overhead to routing tables and routing protocol update

entries (four /16s vs. one /14—rookie!). With successive halving, a /14 is more likely to be readily available for assignment. Let's look at how this works.

If we start with our 10.0.0.0/12 infrastructure block and halve it, we end up with two /13 blocks as illustrated below. Note that thanks to binary arithmetic, associating the next "host" bit with the network enables halving of the original network. Note that the 10.0.0.0/12 network no longer exists, so we've grayed it out to illustrate this; it has been split into our two /13 networks.

Original Network	10.0.0.0/12	***00001010 0000***0000 00000000 00000000
First Half	10.0.0.0/13	***00001010 00000***000 00000000 00000000
Second Half	10.8.0.0/13	***00001010 00001***000 00000000 00000000

Next, let's halve the "first half" above, leaving the 10.8.0.0/13 block available for future allocation or subnetworking for infrastructure applications (or acquisitions!). We extend the network portion of the address now to the fourteenth bit to halve the 10.0.0.0/13 to yield two /14s as below. Note that as with the 10.0.0.0/12 network, the 10.0.0/13 network no longer exists as an entity and also has been grayed out. It has been split into the two /14s shown. However, the 10.8.0.0/13 network is available to the organization for further allocation as needed.

Original Network	10.0.0.0/12	***00001010 0000***0000 00000000 00000000
Original First Half	10.0.0.0/13	***00001010 00000***000 00000000 00000000
First /14	10.0.0.0/14	***00001010 000000***00 00000000 00000000
Second /14	10.4.0.0/14	***00001010 000001***00 00000000 00000000
Second Half	10.8.0.0/13	***00001010 00001***000 00000000 00000000

One way to visualize this halving process from an overall allocation perspective is to view the address space as a pie chart as shown in Figure 6-1. If our entire pie represents the base network, 10.0.0.0/12, then we cut it in half to render two /13s as shown on left of the figure. We can then leave one of the /13s as "available" (left half) and slice the other /13 (right half) into two /14s as shown on the right of the figure.

We have our /14 but let's continue with this scenario assuming we require three /16 blocks for an additional set of new regional offices. Continuing to apply this logic down to a /16, we end up with the following:

Original Network	10.0.0.0/12	***00001010 0000***0000 00000000 00000000
First Half (/13)	10.0.0.0/13	***00001010 00000***000 00000000 00000000
First /14	10.0.0.0/14	***00001010 000000***00 00000000 00000000
First /15	10.0.0.0/15	***00001010 0000000***0 00000000 00000000
First /16	10.0.0.0/16	***00001010 00000000*** 00000000 00000000
Second /16	10.1.0.0/16	***00001010 00000001*** 00000000 00000000
Second /15	10.2.0.0/15	***00001010 0000001***0 00000000 00000000
Second/14	10.4.0.0/14	***00001010 000001***00 00000000 00000000
Second Half (/13)	10.8.0.0/13	***00001010 00001***000 00000000 00000000

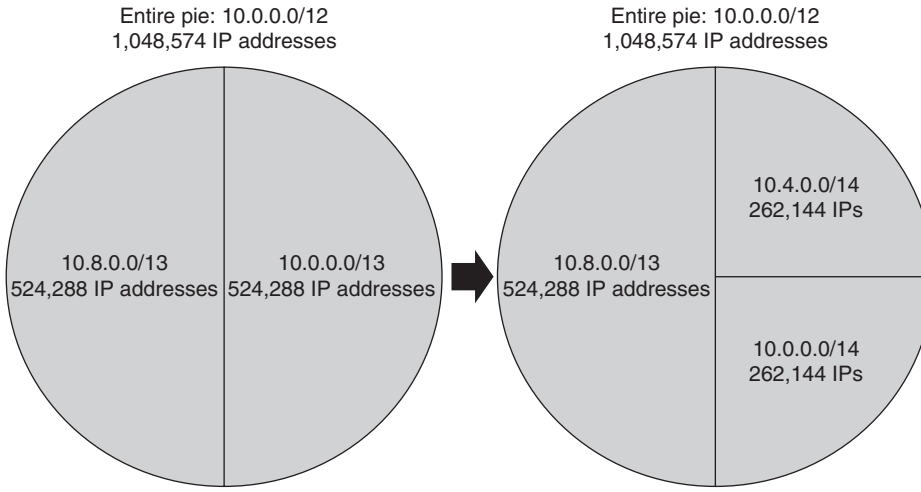

Figure 6-1: Pie chart view of address allocations

As each "first" block is split, it creates two networks of network mask length of one bit longer than the original block. Now that we've performed this split, we have two /16 networks: 10.0.0.0/16 and 10.1.0.0/16 as derived above. We also are left with one /15, one /14, and one /13 shown below our two /16 blocks. These exist because the "first" set of networks were successively sliced in half, yielding a "first" network that was further subdivided, and a "second" network that could be preserved for additional future allocations or assignments. The smallest "first" network, 10.0.0.0/16, is the one we can allocate as it is of the required size.

10.0.0.0/16	***00001010***	***00000000***	00000000	00000000
10.1.0.0/16	***00001010***	***00000001***	00000000	00000000
10.2.0.0/15	***00001010***	***0000001***0	00000000	00000000
10.4.0.0/14	***00001010***	***000001***00	00000000	00000000
10.8.0.0/13	***00001010***	***00001***000	00000000	00000000

But what if we needed a third /16? From which block shall we allocate this? In keeping with our recommendation to retain larger blocks, we'll take our next available network of the smallest size. In our case, from the listing above the 10.2.0.0/15 network is available for further allocation. If we split this /15 into two /16s we have 10.2.0.0/16 and 10.3.0.0/16. We can then assign the former of these two networks as we illustrated earlier, and retain the latter network as available for future assignment. The resulting pie chart is illustrated in Figure 6-2, with the allocated space highlighted.

Notice that we still have many large blocks available for further allocation or assignment. Only the darker-shaded wedge of the pie comprising our three /16 networks has been assigned. In relating the successive splits in the table above

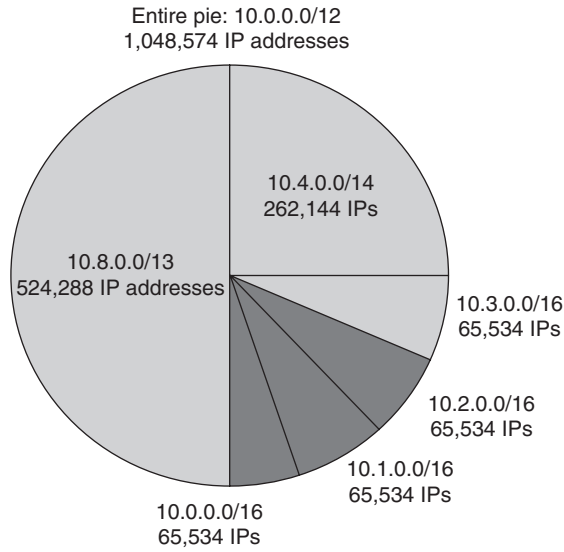

Figure 6-2: Allocation of 3 /16s from /12 space

to the pie chart, while each "first" half block was either assigned or divided into further allocations, it yielded a corresponding "second" half block that is still free or available. Thus the resulting address allocations for our three /16s based on this initial allocation is as follows:

Original Block	10.0.0.0/12	***00001010 0000***0000 00000000 00000000
Free Block	10.8.0.0/13	***00001010 00001***000 00000000 00000000
Free Block	10.4.0.0/14	***00001010 000001***00 00000000 00000000
N. America Block	**10.0.0.0/16**	***00001010 00000000*** 00000000 00000000
Europe Block	**10.1.0.0/16**	***00001010 00000001*** 00000000 00000000
Asia Block	**10.2.0.0/16**	***00001010 00000010*** 00000000 00000000
Free Block	10.3.0.0/16	***00001010 00000011*** 00000000 00000000

Further suballocations follow the same logic, further subdividing each block successively to countries, cities, buildings, etc. in accordance with the desired level of organization and routing topology.

IPv6 Block Allocation.* Though IPv6 addresses are represented differently than IPv4 addresses, the allocation process works essentially the same way. The main difference is in converting hexadecimal to binary and back instead of decimal to binary and back. The process of optimal assignment of the smallest available free block described above for IPv4 is an example of the best-fit allocation algorithm. Due to the vast difference in available address space, IPv6 sup-

*This discussion of IPv6 allocations is based on (118).

ports not only an analogous best-fit algorithm but also a sparse allocation method. We'll also discuss a random allocation method that can be used in lieu of simple subnet numbering starting from 1 and counting up.

Best Fit IPv6 Allocation. Using a best fit approach, we'll follow the same basic bit-wise allocation algorithm we used for IPv4 described earlier. After converting the hexadecimal to binary, the process is identical in terms of successive halving by seizing the next bit for the network portion of the address. For example, consider our example network 2001:0DB8::/32 below.

0010 0000 0000 0001 0000 1101 1011 1000 0000 0000 0000 0000 0000...

Let's say we'd like to allocate three /40 networks from this space. In following the analogous IPv4 allocation example from a binary perspective, by successively halving the address space down to a /40 size shown by the larger bold italic bits below, you should arrive at the following:

0010 0000 0000 0001 0000 1101 1011 1000 1000 0000 0000 0000 0000...
0010 0000 0000 0001 0000 1101 1011 1000 0100 0000 0000 0000 0000...
0010 0000 0000 0001 0000 1101 1011 1000 0010 0000 0000 0000 0000...
0010 0000 0000 0001 0000 1101 1011 1000 0001 0000 0000 0000 0000...
0010 0000 0000 0001 0000 1101 1011 1000 0000 1000 0000 0000 0000...
0010 0000 0000 0001 0000 1101 1011 1000 0000 0100 0000 0000 0000...
0010 0000 0000 0001 0000 1101 1011 1000 0000 0010 0000 0000 0000...
0010 0000 0000 0001 0000 1101 1011 1000 0000 0001 0000 0000 0000...
0010 0000 0000 0001 0000 1101 1011 1000 0000 0000 0000 0000 0000...

Here we readily have two /40 networks available (at the bottom of the list), and translating these back into hex we have: 2001:0DB8:0100::/40 and 2001:0DB8:0000::/40 (i.e., 2001:DB8::/40). After this allocation, to allocate a third /40 using the best fit approach, we can then take the next smallest available network, in this case a /39 (shown third from the bottom in the list above) and split it into two /40s:

0010 0000 0000 0001 0000 1101 1011 1000 0000 0010 0000 0000 0000...
0010 0000 0000 0001 0000 1101 1011 1000 0000 0011 0000 0000 0000...

We split this in half by taking the next bit, yielding two /40s. We can choose one to allocate and the other will be free for future assignment. So our three /40s for allocation are 2001:DB8::/40, 2001:DB8:0100::/40, and 2001:DB8:0200::/40. The other /40, that is, 2001:DB8:0300/40, is available for future assignment. Figure 6-3 illustrates this successive halving in a pie chart form.

After allocating these three/40 networks, highlighted in Figure 6-3, the remainder of the pie is available for allocation. These available networks appear as the top six in the successive halving list above, plus the unallocated half of the former 2001:DB8:200::/39 network.

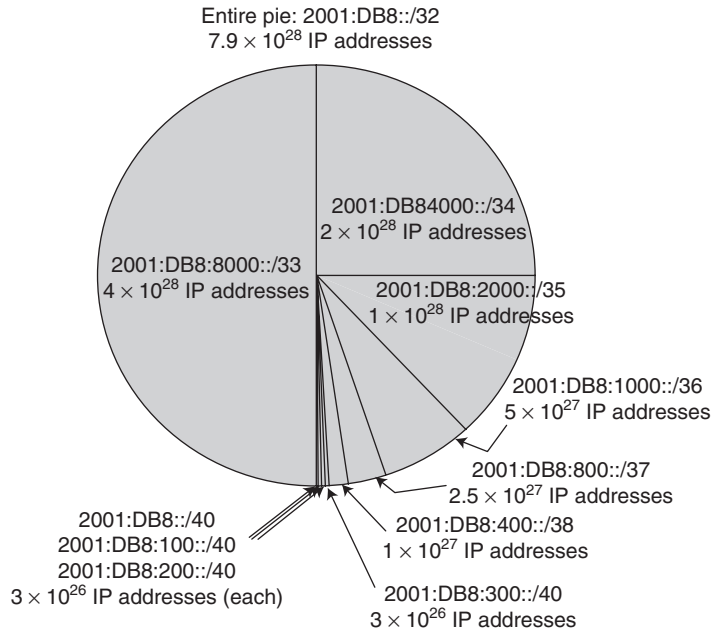

Figure 6-3: Allocation results from carving three /40 networks from a /32 network

Sparse Allocation Method. You'll notice from the prior algorithm that by allocating a /40 from a /32, we incrementally extend the network length to the 40th bit as we did with IPv4 allocation. We then assign the network by assigning a 0 or 1 to the 40th bit as with our first two /40 networks. In essence, we process each bit along the way, considering "1" the free block and "0" the allocated block. However, if we step back and consider the eight subnet ID bits that extend the /32 to a /40 as a whole, instead of incrementally halving the network, we observe that we've actually allocated our subnets by simply numbering or counting within the subnet ID field as denoted by the highlighted bold italic bits below:

0010 0000 0000 0001 0000 1101 1011 1000 0000 0000 0000 0000 0000... 2001:DB8::/40
0010 0000 0000 0001 0000 1101 1011 1000 0000 0001 0000 0000 0000... 2001:DB8:100::/40
0010 0000 0000 0001 0000 1101 1011 1000 0000 0010 0000 0000 0000... 2001:DB8:200::/40

Thus if you knew in advance that the original /32 network would be carved uniformly into only /40-sized blocks, a simpler allocation method would be to simply increment the subnet ID bits. The next allocation would use subnet ID values of `0000 0011, 0000 0100, 0000 0101`, and so on.

On the other hand, if you are a Local Internet Registry or ISP, a sparse allocation method may be more appropriate. The sparse allocation method seeks to spread out allocations to provide room for growth by allocating with the maximum space between allocations. The sparse algorithm also features halving

of the available address space, but instead of continuing this process down to the smallest size, it calls for allocating the next block on the edge of the new half. This results in allocations being spread out and not optimally allocated. Again, the philosophy is that this provides room for growth of allocated networks by leaving ample space between allocations in the plentiful IPv6 space. Considering an example, our allocation of three /40's from our 2001:DB8::/32 space would look like:

```
0010 0000 0000 0001 0000 1101 1011 1000 0000 0000 0000 0000 0000... 2001:DB8::/40
0010 0000 0000 0001 0000 1101 1011 1000 1000 0000 0000 0000 0000... 2001:DB8:8000::/40
0010 0000 0000 0001 0000 1101 1011 1000 0100 0000 0000 0000 0000... 2001:DB8:4000::/40
```

These translate as 2001:DB8::/40, 2001:DB8:8000::/40, and 2001:DB8:4000::/40, respectively. This allocation enables spreading out of address space as illustrated in Figure 6-4. Should the recipient of the 2001:DB8:8000::/40 network require an additional allocation, we could allocate a contiguous or adjacent block, 2001:DB8:8100::/40. This block will be among the last to be allocated under the sparse method, so there's a good chance it will be available. In such a case, the recipient of our two contiguous blocks could identify (and advertise) their address space as 2001:DB8:8000::/39. Note that our subnet ID bits are effectively counted from left to right, instead of the conventional right-to-left method used for "normal" counting.

RFC 3531 (29) describes the sparse allocation methodology. Because network allocations are expected to follow a multi-layered allocation hierarchy, different

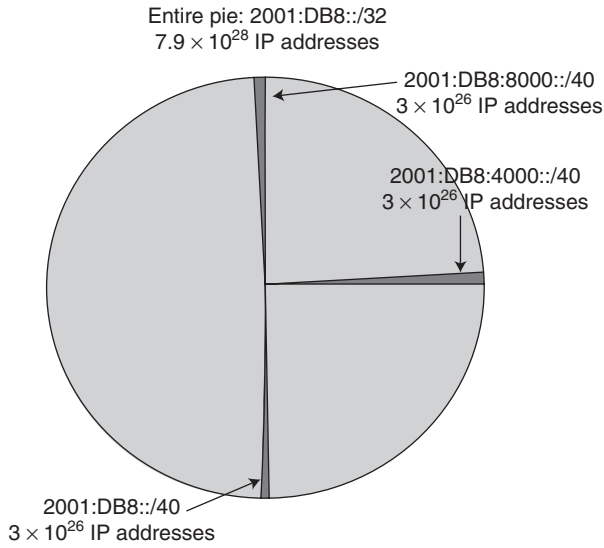

Figure 6-4: Sparse allocation example

sets of successive network bits can be allocated differently by different entities. For example, an Internet Registry may allocate the first macro block to a regional registry, which in turn will allocate from that space to a service provider, who may in turn allocate from their subspace to customers, who can further allocate across their enterprise networks. RFC 3531 recommends the higher-level allocations, for example, from the registries, utilize the leftmost counting or sparse allocation, the lowest level allocations use the rightmost or best-fit allocation, and others in the middle use either, or a center-most allocation scheme. In general, we can use the sparse method to allocate our inter-continental networks, leaving room for future growth at the top level. Note that while RFC 3531 addresses IPv6 allocation, we could also have allocated IPAM Worldwide's top-level IPv4 space in this manner to spread out initial allocations as 10.0.0.0/12 for infrastructure, 10.128.0.0/12 for VoIP, and 10.64.0.0/12 for data.

Random Allocation. The random allocation method selects a random number within the sizing of the subnetwork bits to allocate subnetworks. Using our /40 allocations from a /32, a random number would be generated between 0 and $2^8 - 1$ or 255 and allocated, assuming it's still available. This method provides a means for randomly spreading allocations across allocated entities and generally works best for "same size" allocations. Randomization provides a level of "privacy" in not ordering blocks and subnets consecutively starting with "1." Be aware that random allocation may render the identification of larger contiguous blocks per our earlier merger example more difficult as well as freeing up contiguous space for renumbering purposes. So while it makes sense to allocate sparsely at the top layer of allocation, the random or best fit methods are more appropriate at the subnet allocation level.

Block Allocation Summary. For base and subsequent address block allocations, updating the address plan is a necessary first step. But there's more to be done. To implement the plan, the allocated address space should be configured in the core routers to enable dynamic updating of routing tables. Updating of relay agent information in routers to relay to the DHCP server(s) is another required task. Additional housekeeping tasks may be necessary to add the newly allocated address space to server access control lists (ACLs) at the network interface level and also at the DNS service level regarding "allow" options such as allow-query, allow-recursion, etc. as well as view definitions if appropriate. There is perhaps more to block allocation than it seems! In summary, the task of address block allocation includes the following subtasks:

- Identify sites requiring IP space and quantities of users or IP devices required per site.
- Determine the routing topology in terms of address aggregation requirements.
- Determine whether allocation by application is warranted.

- Identify the minimum allocation at each level of the topology considering growth plans, and employ allocation policies such as a uniform or as-needed strategy.
- Identify free address space within the IP address inventory and allocate a block of the selected size, denoting the allocated block as such in the IP inventory repository.
- Design, procure, install, and configure DHCP and DNS servers if needed.
- Update router configurations with the allocated network and relay agent information.
- Update DHCP and DNS ACL configurations if appropriate.
- Manage the overall allocation process to track locations and servers coming online. Subnet allocations per location are covered next.

Subnet Allocation. Subnets are allocated by applying the same allocation logic in terms of hierarchically subdividing the "IP address pie" down to a subnetwork that is allocated for individual IP address assignment. Even after this base allocation has been deployed, there will be occasions for performing subsequent allocations. Business initiatives will likely drive such occasions. Planned deployment to support several new sites requiring IP addresses due to expansion of the business, plans for new service offerings such as voice over IP, and even a merger or acquisition each can heavily impact the IP address plan. A similar process with respect to sizing up expected capacity requirements, mapping capacity rollups to the supporting routing topology, and consideration of free address capacity and allocation policies can be used for subnet allocations.

This basic task of subnet allocation involves the identification of a subnet that is available, which rolls up with the address allocation plan for the given location and application, and assignment of the subnet in the IP address plan repository. In addition to identifying and recording the allocated block, the subnet allocation process requires provisioning of the allocated subnet on the appropriate router interface. Some addresses on the subnet need to be assigned to infrastructure devices like routers and servers. Defining and updating DHCP server configurations is also required to account for address pool(s) and corresponding DHCP options and/or client class parameters needed for devices that will require DHCP on the allocated subnet.

Devices to be assigned addresses on the subnet now and in the future will likely require name resolution information in DNS. At a minimum, this information applies to a forward domain for domain name-to-IP address lookup and a reverse domain for the IP address-to-name lookup. This requires defining and updating DNS server configurations with domain updates (e.g., in-addr.arpa and ip6.arpa domain(s)) and resource record updates for name servers and statically assigned addresses. Of course, these domains must exist or must be provisioned and configured on respective DNS servers.

Depending on your domain topology, adding a new subnet to an existing location may enable hosts on the subnet to fall within an existing domain, though

this is certainly not necessarily the case. If a new domain is desired, the new domain may be defined and configured as a subdomain or as a new zone on the appropriate DNS servers. In the same way, the reverse domain corresponding to the subnet address may need to be added as well, unless a higher layer in-addr.arpa or ip6.arpa zone will host the corresponding PTR resource records.

The subnet allocation process illustrates one example of the tight interrelationship among address allocation, assignment, and DHCP and DNS server configuration tasks. Depending on your business processes, subnets may be allocated or reserved prior to address assignment and DHCP/DNS configuration. Nonetheless, this set of steps will typically be required to bring a subnet into production.

In summary, the task of subnet allocation includes the following subtasks:

- Identify free address space within the scope of the topology where the subnet is needed.
- Allocate a subnet of the required size from the appropriate address space and record the allocation in the IP address inventory repository.
- Update router configurations regarding the allocated network.
- Assign and provision manually assigned addresses for routers, servers, or other subnet infrastructure devices.
- Design and configure DHCP address pools if necessary to server dynamic hosts on the subnet. This may require association of options, directives, and client classes based on requirements of devices planned for utilization of the address pool(s).
- Define new DNS domains required to serve hosts on the subnet and configure appropriate DNS servers.
- Complete the allocation process by confirming provisioning and reachability of the subnet, as well as by verifying corresponding DHCP and DNS configurations.

IP Address Assignment. Assigning, deassigning, and reassigning IP addresses to individual hosts is usually the most frequent IP management activity in most organizations. This is typically associated with deployment, redeployment, or decommissioning of IP devices, including routers, servers, printers, and the like. In terms of address assignment, the IP address inventory database must be consulted to identify an available IP address. If possible, it would be useful to ping the IP address to be assigned just to verify accuracy of the inventory, though we'll discuss the process of overall inventory assurance as a separate management task. The IP address to be assigned should then be noted as assigned to the given device within the IP inventory database.

The actual physical IP address assignment may be performed by manually (statically) configuring the device, by using DHCP (in this case, we'll assume Manual DHCP is used to assign the designated IP address to the corresponding host) or via IPv6 autoconfiguration. In the static assignment case, the assigned

address must be configured directly on the device, so unless the IP address assigner is also responsible for the physical assignment, this process would entail an email or phone call to the device owner conveying the assigned IP address information to be entered. When using M-DHCP, an entry in the appropriate DHCP server's configuration file would be necessary to map the device's hardware address to the assigned IP address. Autoconfigured devices will construct the autoconfiguration address and verify its uniqueness through the duplicate address detection process automatically.

Most devices with IP addresses will require corresponding DNS resource records to enable reachability by name. Using the DHCP method of address assignment, the DHCP server can be configured to update a DNS server upon assignment of the IP address. This update would affect the forward domain for domain name-to-IP address (A/AAAA record) lookup and the reverse domain for the reverse (PTR record) lookup. A similar DNS update task would be required if assigning the address manually or if a device autoconfigures. Updating DNS with this new host information may entail editing or updating the corresponding zone files on the server or by sending dynamic updates.

You may not want an autoconfigured device to update DNS on its own, at least on an enterprise network, though this may be suitable for a community or ad hoc network. Identifying the presence of a newly autoconfigured device to manually update DNS presents its own challenge! If such devices require resolution information in DNS, use of a router log or subnet snooping utility may be necessary to identify the IPv6 address.

In summary, the task of IP address assignment includes the following subtasks:

- Determine how the device will obtain its IP address: via manual configuration or via DHCP.
  - If D-DHCP or A-DHCP, determine whether current address pools, if any, on the subnet have capacity to support the device; if so, this task completes; if not, configure an address pool of the corresponding DHCP type on the DHCP server along with necessary option parameters.
  - If M-DHCP, identify a free IP address within the subnet where the device is located and assign the address to the device by configuring DHCP to reserve or use M-DHCP for the device's MAC address.
  - If manually configured on the device, identify a free IP address within the subnet where the device is located and assign the address to the device. Have the assigned static IP address configured on the device manually.
  - In all cases update the IP address plan with the assigned address, whether a spreadsheet or other IPAM tool.
- Determine whether DNS resource records need to be manually created and updated. This would generally be the case for statically assigned addresses. For DHCP-assigned devices, the DHCP server can be configured to perform dynamic updates, though in some cases where dynamic

updates are not feasible or allowed by policy, manual updating of corresponding resource records may be required.

- Verify completion of the address assignment process by pinging the address successfully and verifying its resource records in DNS. For devices assigned an address via an address pool, verification may not be needed; however, if it is, the address may not be known a priori. Locating the device's MAC address in the DHCP lease file, followed by a ping of the corresponding address, confirms its assignment in this case.

Address Deletion Tasks

As we've illustrated, address allocation is a top-down process, with allocation of hierarchical blocks, from which subnets can be allocated, from which IP addresses can be assigned. Deletion of address space requires the inverse operation and is necessarily bottom-up. Deleting an address block before the underlying blocks, subnets, and IP addresses have been deleted would strand these underlying elements, so unless you enjoy mass chaos, a more controlled process is warranted.

Deleting IP Addresses. Deleting an IP address is relatively straightforward: delete or free up the IP address in the IP inventory, removing the M-DHCP entry from the DHCP server if appropriate along with releasing the lease, and removing associated DNS resource records. However, care must be taken to assure the address has been relinquished by the device and that DHCP and DNS updates have been completed before assigning the address to another device. Simply deleting a lease on a DHCP server does not force the client holding that lease to relinquish it. Denoting the address as in a state of "pending deletion" or something similar would alert other administrators not to assign that address to another device until confirmation is received of its availability. This confirmation process entails pinging the address, perhaps successively over several days, and confirming the deletion of its associated data in DNS and DHCP servers.

Deleting Subnets. Deleting a subnet may be required when closing a site or consolidating address space. Devices with IP addresses on the subnet to be deleted should be moved or decommissioned such that the subnet is free of address assignments (other than perhaps subnet-serving routers). After all IP addresses have been verified as free, the subnet may be reclaimed into the free address space for future allocation.

Upon freeing up of a subnet, it may be possible to join the freed space with a contiguous free address block, creating a larger free block. For example, if I delete subnet 10.32.142.0/24, which is contiguous with the 10.32.143.0/24 block, which is also free, I could join these two blocks into a single free block, 10.32.142.0/23. Doing so now makes it clear that this "bigger block" /23 could be assigned for growth or a new location within the topology.

Deleting Blocks. Address block deletion may result from the withdrawal from a major business market or consolidation of sites, among other reasons. Generally all downstream IP addresses, subnets, address pools, resource records, and domains should first be decommissioned before the macro-level block can be freed up for future assignment consideration. Thus, after the individual delete IP address tasks and delete subnet tasks within the target block have been completed, the block itself may be freed up. As with a modest-to-large allocation task, project planning resources may be required to verify deletions up the hierarchy. The deleted or freed block space may be joined with contiguous free space of an equal size in the same manner as in deleting subnets. Additional housekeeping tasks related to DHCP and DNS ACL configurations should be considered too if impacted by the block deletion.

Address Renumbering or Movement Tasks

Moving or renumbering of address blocks, subnets, or individual addresses combines the allocation process with the deletion process. The allocation process, as described above, should be performed from a top-down perspective to allocate space to which underlying subnets and IP addresses will be moved. The deletion process frees up address space from the bottom up as addresses are moved to the target allocated space. In essence, the size of the scope of the addresses to be moved must be allocated to accommodate the addresses to be moved, temporarily doubling the address space associated with this set of devices. As addresses are moved, the former address space can be freed up, returning address allocations to previous levels.

IP Address Moves. Moving an IP address can be considered a combination of assigning an IP address on the destination subnet and deleting the IP address on the current subnet. Depending on the method of address assignment and the type of move, different tactics can be used. The type of move relates to physical movement of a device to a different subnet (physical move) vs. the reassignment of the IP address on the same or a different subnet (logical move). A physical move of a non-mobile IP device will typically involve a "reboot" of the device, which affords more control of the address assignment process.

Physical Moves. Physical moves imply powering down, moving, then powering up devices at the destination location. For D-DHCP and A-DHCP assigned devices, if an entire pool is being moved, the destination pool should be set up on a [same or different] DHCP server. Make sure the router(s) serving the destination subnet are configured to relay DHCP packets to the DHCP server configured with the new pool. When these devices power up, they will likely attempt to renew the most recent lease they possessed on the old subnet. Make sure that A-DHCP devices do issue Renews upon power up and don't just continue using their old IP lease; if they assume the old [infinite] lease is valid, manual intervention will be required to reset the device's address. Otherwise,

the DHCP server will NAK the Renew attempt by each client. Clients will then issue a Discover packet to request a new lease. The DHCP server obliges with a lease within the new destination pool.

Physical movement of a M-DHCP device entails creating the M-DHCP entry in the DHCP server serving the new subnet and deleting the entry on the former DHCP server. If the same DHCP server is being used, simply edit the IP address associated with the device's MAC address while the device is in transit. When the device powers up on the new subnet, it should follow a similar process to D-DHCP and A-DHCP, with a Renew attempt, which the DHCP server would NAK, followed by reversion to issuing a Discover and address reassignment using the standard DHCP process.

Moving a device that autoconfigures its IPv6 address will lead to the device detecting its new subnet via neighbor discovery, autoconfiguring its address, then verifying its uniqueness through duplicate address detection.

Updating of DNS resource records may be performed by the DHCP server or manually if dynamic updates are prohibited for these DHCP cases. DNS updates for autoconfigured devices present the same challenge as in adding such devices. Once all devices have physically moved, the pool serving the old subnet may be decommissioned.

Physical moves of manually configured devices requires assignment of an address from the IP inventory, and manually configuring the new IP address in the device as it powers up on the new subnet. At this point, the old address can be freed up. DNS resource records should be updated as well to reflect the device's new IP address.

In all of these cases, the IP inventory should be utilized to identify free address(es) on the destination subnet or pool, and to free up addresses on the old subnet as device moves are confirmed.

Logical Moves. Logical moves are a bit more challenging as they do not necessarily involve a device reinitializing. For DHCP devices, an address pool containing the destination IP addresses should be configured on the [same or different] DHCP server. The lease time for the pool or device should be stepped down in advance of the move date. For example, if a normal lease time is one week, it should be lowered to one day, for example, during the week leading up to the move and to 2–6 hours on the day of the move. A device may have renewed a weeklong lease just before you changed the lease time to days, so it will not attempt to renew until halfway through the week (or based on your lease time option setting). Thus, if your nominal lease time is two weeks, ratchet down the lease time two weeks before the planned move. On the day of the move, set the lease time to a minimum time if it's important that all devices move at nearly the same time. Minimum time can be on the order of minutes or hours depending on network traffic and server performance considerations. The shorter the lease time, the more DHCP packets will be sent but the more time-aligned the move of DHCP clients can be orchestrated. If move coincidence is not critical, leaving lease times on the order of hours should yield a complete move within a few hours.

In this scenario, it's recommended that the DHCP server perform DNS updates if possible to more closely map DNS information updates with actual address changes. Manual intervention of A-DHCP devices may be necessary unless they do adhere to lease renewal policies despite possessing infinite leases.

Movement of manually addressed devices follows the same process as in physical movement. A destination IP address is assigned from the IP inventory, and the new IP address is configured on the device. Once confirmed, the old address can be freed up. DNS resource records should be updated as well to reflect the device's new IP address.

Logical movement of an autoconfigured device can be performed by configuring the router serving the corresponding subnet to ratchet down the preferred and valid address lifetime values it advertisements during the neighbor (router) discovery process. Shortening these timer values for the address prefix from which the device is being moved while introducing the new prefix with a "normal" address lifetimes will enable autoconfigured devices to perform this logical move automatically. Once all devices have moved and the valid lifetime of the former prefix expires, the prefix can be removed.

Subnet Moves. Moving a subnet could involve one of two tasks: movement of the subnet and its assigned IP addresses to another router interface, preserving the current address assignment or movement to another router interface, requiring a new subnet address. We'll include the subnet renumbering task in this discussion with respect to the latter case as it also results in a new subnet address, though without necessarily moving the subnet to another router interface. The first subnet move case requires consideration of address space rollup within the hierarchy but generally consists of modifying and verifying router provisioning compliance with the plan, as well as updates to routing tables and DHCP relay addresses as necessary.

Movement of a subnet due to a physical move or higher level renumbering requires a bit more work. A physical movement where devices are physically moved, for example, when an office is moved, is inherently disruptive. However, reallocation of office wiring from one router to another is not. The destination subnet may be allocated and provisioned on the destination router interface, along with the other tasks described above related to reserving static addresses and updating DHCP and DNS configurations. When each moved device plugs in, it will need to be manually readdressed with the new address and/or obtain a DHCP lease on a pool relevant to the subnet as described above for IP address moves. Logical subnet moves or renumbering likewise follows the logical IP address move process for each device.

After all devices have been moved from the old subnet to the new, the old subnet may be freed up using the delete subnet process.

Block Moves. Moving macro-level blocks with underlying subnets and IP addresses requires careful project planning and execution. The allocation of the destination block should follow those tasks outlined for block allocation.

Assuming a move is for renumbering only, a like-sized destination block should be allocated. If the move is motivated by or otherwise spurs the opportunity for address consolidation or expansion, the destination allocation should be sized based on underlying capacity requirements and topology architecture as discussed in the Block Allocation section. Once the allocation has been made, suballocations and subnet allocations may begin. IP addresses and pools can then be moved following the process described for IP address moves. As IP addresses and subnets completely move from their old assignments, these can be decommissioned or freed up when moves have been confirmed.

Block/Subnet Splits

Splitting an address block entails the creation of two or more smaller-sized blocks from a given source block. Splits may be necessary to free up address space or even as a means of suballocation of address space. In the former case, the addresses within a subnet may be consolidated to the first half of the subnet, freeing up assignments in the second half. In this case, splitting the block yields an occupied subnet (first half) and a free subnet (second half). Some organizations have historically allocated regional blocks, then split them to assign sub-blocks and subnets lower in the address hierarchy. In some sense this is a form of block allocation.

Splitting a block need not be restricted to only splitting in half, say a /24 into two /25s. A split may be used to carve out a /23 from a /20, which yields a /23 and free space consisting of a /23, a /22, and a /21. In this example, we preserved large blocks following our optimal allocation strategy as we discussed in the block allocation section. Alternatively, we could have simply split our /20 into eight /23s, though this may be wasteful unless same-sized allocations are used by policy, which is usually the case with uniform allocations but not with the as-needed allocation strategy.

In summary, the process of splitting a block is similar to that of allocating a block. The block to be split is successively divided until the desired block size is attained. Remaining free blocks are either retained or also split to the same size as the desired block to render a uniform block split.

Block/Subnet Joins

A join combines two contiguous same-sized address blocks or subnets into a single block or subnet. We saw an example of joining blocks in the block deletion section. After freeing up the 10.32.142.0/24 block, we joined it to a contiguous free block, 10.32.143.0/24, to create a single 10.32.142.0/23 block. Successive joins may be performed to consolidate smaller chunks of contiguous address space. Joins are only valid for contiguous blocks of the same size. Joining a /25 and a /24 is not valid as the “other /25” not included in the join must remain uniquely identified. However two /25s and a /24 can be joined to form a /23 assuming all blocks are contiguous. The two /25s would be joined first to form a /24; then this and the other /24 can be joined to create a /23.

DHCP Server Configuration

DHCP server configuration is a key IPAM task. As we've discussed, the address management tasks covered so far have impacts on DHCP server configurations. This configuration goes beyond address pool creation, movement, and deletion, though the extent of additional functions is constrained by the capabilities of the DHCP server vendor. Key among DHCP server configuration parameters are:

- DHCP address pools—address ranges and associated DHCP options and server policies for dynamic, automatic and manual DHCP clients
- Client classes—parameter match values (e.g., vendor-class-identifier = "Avaya 4600") and associated allow/deny pools and DHCP options and server policies
- High availability parameter settings for primary/failover or split scopes
- Configuration of server policies such as dynamic DNS updates and other server directives and parameters

The actual server configuration syntax and interface will depend on the server type. For example, ISC DHCP servers can be configured by editing the dhcp.conf text file while Microsoft DHCP can be updated using a Windows Microsoft Management Console (MMC) interface. Both of these and other DHCP vendors also provide command line interfaces or APIs to perform configuration updates.

DNS Server Configuration

Like DHCP, DNS server configuration is tightly linked with address allocation, assignment, moves and deletions, as we've seen. These tasks discussed previously affect DNS domains, resource records, and possibly server configuration parameters. Key among DNS server configuration parameters are:

- Domains—adding, modifying, or deleting domains/zones on DNS servers
- Resource records—adding, modifying, or deleting resource records
- Server, view, and zone configurations—option parameters affecting ACLs, server performance

The actual DNS server configuration syntax will depend on the server type. ISC BIND servers can be configured by editing the named.conf and associated zone files on the server. DNS servers that support DDNS may also support resource record updates in this manner. The use of `nsupdate` or similar DDNS mechanism provides a means to perform incremental updates without having to manually edit zone text files and reload respective zones. DDNS updates apply to resource record adds/changes/deletes only, so any zone or server configuration parameter changes or zone additions or deletions would still require text file

editing and reloading of named.conf and/or affected zones. On a day-to-day basis, however, resource record changes usually occur much more frequently than server or zone changes.

Server Upgrades Management

New versions of DHCP and DNS server software are published periodically to address security vulnerabilities, provide bug fixes, or offer new features. The urgency to perform an upgrade is usually dictated by what's provided in the upgrade, with security vulnerabilities certainly being of highest urgency. The upgrade process is typically vendor-specific and may require alignment of which hardware platforms and operating systems the upgraded version has been certified to run on. Hopefully the underlying operating system requirements would change only for new feature introductions and not security or bug fixes, but this is governed by vendor policy.

Most vendor DHCP/DNS appliance upgrades roll in the operating system upgrades as necessary within an overall upgrade package. Because the appliance vendor typically provides the operating system with the hardware platform, they should publish compatibility upgrades as necessary for newer versions of their DHCP and DNS services. Most appliance solutions enable centralized staging of upgrade packages, with deployment to distributed appliances, vastly simplifying the upgrade process over a software-based upgrade process, where the planned software upgrade must be validated with current or planned operating system versions or patches.

If you're running ISC, Microsoft, or other vendor DHCP or DNS daemons on your own hardware, you'll need to keep apprised of not only DHCP/DNS security updates but also those affecting the corresponding operating system running on the hardware.

FAULT MANAGEMENT

Fault management encompasses not only fault detection, but alert notification, trouble isolation capabilities, trouble tracking, and resolution processes. Monitoring of DHCP and DNS servers for faults and events enables a proactive means of minimizing services outages. In a well-designed network services architecture, clients should be able to obtain leases and resolve domain names despite a given DHCP or DNS server outage. Nevertheless, detection of such an outage is important as the outage reduces the number of servers that clients may use to obtain these services thereby raising the vulnerabillity to service outage in the event of an additional server failure. For example, in a DHCP failover deployment, failure of one server will leave just one server available to service DHCP clients. In such a scenario, detection of the failed server facilitates timely, though not panicked, resolution of the server outage.

Monitoring and Fault Detection

Fault detection may be performed using a variety of methods depending on the capabilities supported by deployed DHCP and DNS servers. These range from proprietary polling or notification, to syslog scanning and/or forwarding, to SNMP polling and trap detection by SNMP-based network management systems. Some commercial IP management systems offer intra-system or proprietary monitoring, particularly for appliance-based products. Since appliances are fully self-contained solutions, incorporating not only DHCP and DNS services but a hardware platform and operating system, the appliance vendor should have the ability to fully access fault information related to the appliance at the hardware, operating system, and DHCP/DNS levels.

In addition to monitoring the state of DHCP and DNS servers, as reported by the servers, it's a good idea to monitor for hung services. This may occur if a service is running but is in a state where it is unable to perform its role in providing leases or resolving DNS queries. This can be detected by analyzing successive polls for lease or query trasactions received and processed, and verifying differential counts greater than zero, assuming normal transaction rates at that particular time of day are non-zero.

An alternative, on-demand form of service testing involves sending the server a DNS query or Discover (or Solicit) packet and verifying receipt of a proper response. This tactic provides some assurance that the services are not only running, but are responding to clients. The bottom line is that some form of service functional fault detection can provide a truer mapping to what an end user may consider a fault or outage.

In addition to monitoring DHCP and DNS servers, monitoring of the IP management system itself provides benefits of assuring access to IP address and DHCP and DNS server configuration information that may otherwise be prevented by an outage. At a minimum, backup or distribution of the data store provides a snapshot to reconstruct the information in the event of a site outage or disaster.

Monitoring of networking equipment and communications links is a common practice for general network monitoring and can provide insights to outages affecting the ability of clients to reach DHCP or DNS servers. This added information can be very helpful in troubeshooting a particular problem or fault.

Fault correlation is the analysis of individual faults received from multiple network elements or management systems to help isolate the root cause of a set of faults. For example, faults from a layer 2 switch, a router, and a WAN access device can be analyzed collectively to suggest that these three faults are related and the likely root cause is a link outage. Fault correlation is a common feature of large-scale network management systems, and if your IP management system provides alarm feeds, it may be able to feed into a higher level alert correlation function. Whether fault correlation is performed automatically by a network management system or manually by comparing information from multiple

systems, this process exposes a broader set of data for fault analysis with the goal of isolating a fault to a given server, link, or network element.

Troubleshooting and Fault Resolution

DHCP/DNS Server Troubleshooting. Fault management capability is an important consideration for those responsible for managing an IP network, and critical DHCP and DNS network services should be among those elements monitored. Mitigation of the impacts of faults may be achieved through deployment of highly available configurations to minimize end user impacts of an outage of any individual component. Once a fault has been detected, troubleshooting server faults requires network reachability to the server, secure login to view logs and to restart services or daemons as necessary. If the server is hung and needs to be rebooted this may require local manual intervention.

Use of "lights out" management capabilities such as Intel's Intelligent Platform Management Interface (IPMI) or HP's integrated Lights-Out (iLO) technologies enable remote rebooting or power cycling of a server or appliance. These interfaces also enable monitoring of the boot process as if one was viewing the console inteface locally. These interfaces are also helpful from a monitoring perspective in providing temperature, power, and other key indicator readings for the server.

IP Address Troubleshooting. A variety of tools are available to troubleshoot IP assignment, DNS, and DHCP faults, some of which may even be provided by your IPAM vendor. To verify or identify IP address assignments, intentional or otherwise, a variety of discovery techniques will prove beneficial. Ranging from a simple ICMP Echo request, ping, traceroute, nmap or SNMP, a variety of tools may be used to attempt to contact individual hosts or view router or switch ARP tables. Many IPAM systems incorporate at least one form of discovery to provide verification of IP address assignments or to assist in troubleshooting.

DNS Troubleshooting. Beyond server reachability and server/service status checks, troubleshooting of DNS resolution is a key function required to diagnose and resolve DNS issues. Among the most popular DNS diagnostic tools, nslookup (name server lookup) and dig (domain information groper), both of which ship with the BIND software distribution. Nslookup also ships with the Microsoft Windows DNS distribution.

NSLOOKUP. Nslookup (30) is a utility that enables querying of a DNS server. Today, many administrators prefer dig, which provides much more detail and control over the query formulation, resolver configuration override, output formatting, and more. To perform a single lookup using nslookup, simply type:

```
nslookup lookup-value [name server]
```

where *lookup-value* is the host domain name or IP address to lookup and the *name server* is the DNS server name or IP address to query. Additional options may be included preceding the *lookup-value* with each option name prefixed with a hyphen (e.g., `–timeout=5`). Interactive mode for nslookup may be invoked by either entering nslookup with no arguments or entering nslookup followed by a hyphen, space character, and name server hostname or IP address like:

```
nslookup - 172.18.71.105
```

Interactive mode enables entry of commands to formulate and perform queries.

DIG. Dig (31) enables the formulation of a DNS query using standard DNS messages, emulating a resolver or recursive server. Dig provides granular control of the format of a query that can be sent to a DNS server to analyze the results.

A common example usage of the dig command simply requests a resolution for a host name:

```
dig @ns1.ipamworldwide.com A ftp-sf.ipamworldwide.com
```

This example would result in the issuance of an A record query for ftp-sf.ipamworldwide.com to the DNS server ns1.ipamworldwide.com. Several additional arguments for the dig utility may be entered to focus the test query. For example, –p enables specification of the UDP (or TCP) port number to query as the destination port, -6 issues the query over IPv6, and –x IP-address queries for the PTR record(s) for the entered IP-address. All dig options and parameters are discussed in the dig man page or in *IP Address Management Pricinciples and Practice*.

DHCP Troubleshooting. DHCP transactions can be tested using DHCP client capabilities like ipconfig for Windows or ifconfig commands for Unix or Linux. These commands provide that ability to perform DHCP releases, renews, and set user class. For example, using `ipconfig` on a Microsoft Windows command line enables display of the IP configuration using the following arguments:

- `/all`—Displays IP configuration information for each interface including
  - IPv4 address and subnet mask
  - Additional IP addresses including IPv6 addresses
  - MAC address
  - Interface description
  - DNS domain suffix

  - Default gateway
  - DHCP server from which the lease was obtained along with dates/times the lease was obtained and that the lease expires.
  - DNS servers for resolver configuration
  - WINS servers to query for NetBIOS lookups if configured
- Omitting the /all argument displays the IP addresses, subnet mask, domain suffix and default gateway only.
- `/?`—Displays help in the form of a command summary.
- `/displaydns`—Displays contents of the resolver's cache.
- `/showclassid` *`adapter`*– displays the user class configured for the specified interface adapter.

`ipconfig` also provides the following commands:

- `/release` [*`adapter`*]– Issues a DHCPRELEASE to release the lease for all or the specified interface adapter.
- `/renew` [*`adapter`*]—Issues a DHCPRENEW to renew all leases or that for the specified interface adapter.
- `/registerdns`—Issues a DHCPRENEW to renew all leases and updates DNS A record(s) directly (client to DNS server, not DHCP server to DNS).
- `/flushdns`—Clears the resolver cache.
- `/setclassid` *`adapter class`* —Sets the user class name for the specified interface adapter.

ACCOUNTING MANAGEMENT

Accounting management basically intends to keep everyone honest. Are those assigned addresses still in use? Are any unassigned addresses actually being used? Did the new subnet get provisioned on the router yet? Thus, accounting management enables verification of successful configuration, as well as overall adherence to the IP addressing plan. Techniques for accounting management functions include discovery of IP addresses, router subnets, switch port mappings, DNS resource records, and DHCP lease files.

Analysis of discovered information is necessary in order to compare this information with the inventory "plan of record." Such discrepancy reporting and comparison is difficult work, but provides a level of assurance of inventory accuracy. Without such a function, rogue users could access free service or otherwise infiltrate the network. In addition, planned network changes yet unimplemented may cause downstream process delays and violation of internal or external service level agreements (SLA) on provisioning intervals.

Inventory Assurance

Each of the common IP management tasks we've covered so far relies on accurate IP address inventory to enable the allocation, deletion, and movement of blocks, subnets, IP addresses, and DHCP and DNS server configurations. But accurate inventory is also essential for general troubleshooting. Should a remote site be unreachable due to a network outage, it may be necessary to identify IP addresses, resource records, or other IPAM-related data for devices at the site. Only by maintaining an accurate IP inventory can such information be accessed when it may be needed most and when it cannot be obtained directly from the network.

In this section, we'll review steps you can take to assure the accuracy of your IP inventory. This includes controlling who can make certain changes to certain IPAM information, discovering actual network data, reconciling the actuals with the inventory, and finally reclaiming address space.

Change Control and Administrator Accountability. As we've seen in reviewing these IP management tasks, a change in IP inventory often affects other network elements, including routers, DHCP servers, and DNS servers. If different individuals or teams manage these different elements, it's a good idea to convene a planning or change control meeting periodically or as needed to review and schedule upcoming planned addressing changes. A little rigor can add some discipline to the process and keep those potentially affected by changes in the loop.

One way to help assure accuracy of IP inventory itself is to limit write access to the inventory to those who are authoritative for and keenly knowledgeable of the IP addressing plan. Using a single password-protected spreadsheet that the one and only IP planner can modify is one approach to protecting the IP inventory from inadvertent or erroneous changes. However, for even modestly sized organizations, this approach is unwieldy. With the organization reliant on a single individual for the entire IP address plan, the individual must work around the clock and, should he or she leave the organization, recovery of access to the inventory may be very difficult unless a successor is groomed in advance.

Support of multiple simultaneous administrators is a key feature of most IPAM systems on the market, and most allow some level of scope control so that certain administrators can only perform certain functions on certain devices or portions of the network. Make sure your chosen system supports administrator logging should you need to investigate "who did what" on the system.

As important as disciplined multi-administrator scoped access to the IP inventory is to delegating accountability, arbitrary changes to IP address assignments, DNS resource records, and subnet addresses can be made outside of the scope of the IP inventory. For example, manual configurations can be mistyped, subnets can be provisioned on the wrong router interface, and client or DHCP updates to DNS can all contribute to IP inventory drift from reality. The IP

inventory is a model of the IP address plan, and IPAM tasks rely on the accuracy of the plan. Therefore, additional "pulse readings" are required from the IP network itself. Periodically polling and comparing the actual assignments on the network with the inventory is key to assuring inventory accuracy.

Network Discovery. A variety of methods are available to gather network actuals data, from ping, to DNS lookups, port scans, and SNMP polls. Pinging enables detection of an occupant of an IP address and provides a basic method to determine which IP addresses are in use for comparison with the respective portion of the IP inventory. Ping is very useful but be aware that some routers or firewalls will drop ping packets and some devices can be configured to ignore pings. Setting up remote ping agents to perform local pinging on command can help avert the router/firewall traversal issue.

Nmap, freely available at insecure.org/nmap, is a particularly useful tool at the right price. It combines several discovery mechanisms to gather a variety of information from devices connected to the IP network, including ping sweeps, DNS lookups, and port scanning. When sweeping a subnet, nmap can perform these tasks in one command, issue a ping to each address, looking up a corresponding PTR record in DNS, and attempting connections to various TCP and UDP ports to identify the device's operating system. From an IPAM perspective, ping results help identify IP address occupancy, DNS lookups help corroborate hostname-to-IP address mapping between DNS servers and the IP inventory, and port scanning can provide additional information about the type of device occupying each IP address.

SNMP is another means of discovering IP inventory-related information. While most end devices like laptops or VoIP phones don't natively enable SNMP, most infrastructure elements like routers, switches, and servers do. Of particular interest within router MIBs are the Interfaces, IpAddresses, and Arp tables. If your infrastructure devices support MIB-II, the interpretation of these tables *should* be consistent across different products. Just be aware of minor variations, even among different products from the same vendor. The information in these tables enables collection of the interfaces and subnets per interface provisioned as reported by the router. This provides useful validation of inventory in general, but can also be polled when in the process of allocating, moving or deleting blocks and subnets.

Polling router ARP cache tables can provide a mapping of MAC addresses to IP addresses on recent subnet communications. Even if a device refuses to respond to a ping, it must use the address resolution protocol (ARP) to formulate a layer 2 (e.g., Ethernet) frame in which to envelop its intended IP packet. As implied by the fact that this is cached information, it is transient and may need to be polled frequently.

IP Inventory Reconciliation. Network discovery information provides a reality check on actual subnet allocations, IP address assignments, and associated resource records. By comparing discovered information with the IP inventory

database, discrepancies can be identified and investigated. While this comparison may require "eyeballing" the differences between the inventory spreadsheet and the discovery output, the effort can prove beneficial for several reasons. For example, database discrepancies can be identified that may be the result of:

- Incorrect router provisioning—Incorrect subnet, mask, router interface, etc.
- Incomplete router provisioning—Planned change not yet implemented.
- Device reachability issue—if a device should be at a given IP address and no response is received. This could result from a device outage, a transient outage (reboot), address reassignment, or network unreachability.
- Incorrect IP address assignment—Manually configured address is incorrect or device obtains a DHCP address from an unintended pool or address.
- Actual IP address assignment—In some decentralized scenarios, the installer of a device on the subnet may select an IP address; discovery can be used to update the IP inventory accordingly.
- Incomplete IP address assignment—All aspects of the assignment process, whether manual or DHCP, are incomplete. This issue is particularly applicable to manually assigned addresses where manual effort is needed to configure the assigned IP address and to update DNS.
- Rogue device presence—An unknown or unauthorized device has obtained an IP address. This provides an effective post access control mechanism to complement and audit a network access control solution.

In addition to detecting discrepancies, analyzing discovery information can confirm completion of allocation or assignment tasks, as well as delete tasks. Discovery data is indispensible when moving blocks, subnets, and IP addresses. Since moves require allocation of the new address(es), movement, then deletion of the old address(es), confirmation of move completion is essential prior to deleting the old address(es) from the IP inventory. These addresses should not be deleted before the move completes so they are not unknowingly reassigned to other devices or subnets prior to their actual relinquishment.

In summary, network discovery is essential to assuring the accuracy of the IP inventory. It is also beneficial to monitoring provisioning or assignment progress and time frames, managing the completion of tasks requiring multiple related subtasks, and detecting incorrect assignments as well as potentially rogue devices.

Address Reclamation

One of the benefits of network discovery and reconciliation just discussed is the detection of device reachability issues. If a server has been provisioned and has historically responded on a given IP address, but now no longer does so, such an event should stimulate further investigation. If there were no plans to move or decommission the device or there are no network problems reaching other

devices on the subnet, the device may be suffering an outage, may be rebooting, may have been moved or disconnected, or may have been re-addressed. If the server is providing critical services or applications, hopefully you're monitoring its status via a network management system (if the server is a DHCP or DNS server, it may be monitored via the IP management system) that can corroborate the outage prognosis and trigger corrective actions. If the IP address is discovered on the next attempt, perhaps it was simply rebooting. If it does not respond for the next *n* attempts, perhaps it is no longer physically (or at least electrically!) there. Unfortunately people don't always inform the IP planning team that a device has been removed or moved elsewhere, even in the tightest of organizations. A quick phone call to the site to check on the device's status may prove fruitful, but it's often difficult and time-consuming to identify the device's "owner" to verify status.

Nevertheless, the key point to assessing the possible fate of the device is that it may take multiple discovery attempts to determine whether a device was there and no longer is, suffered a transient outage or disconnect, or was borrowed and has now been returned. Tracking a succession of discovery attempts may be difficult. A running log or spreadsheet can be used to log discrepancies or "missing" IP addresses as they are [not] detected. Reviewing this log over time may help determine whether an IP address recorded as in use actually isn't.

In reviewing such a log, if a given IP address had been successfully discovered until a month ago, when it was last reachable after so many attempts, say 30, it may be confirmed as available for future assignment, or *reclaimable*. The concept of reclaim entails identifying IP addresses that are denoted as in use in the IP inventory, but are in reality not in use, nor have they been in use in recent history. Analyzing multiple discovery results provides a reasonably robust sample set on which to base a reclaim decision, essentially deleting the device from the inventory and freeing it up for assignment to another device.

Besides providing robust confirmation of a device deletion from the IP inventory, reclaim may likewise be applied to subnets. When deleting a subnet, it's generally advisable to verify that all IP address occupants have been deleted and are no longer using IP addresses on the subnet. Analyzing discovery results from all addresses on a given subnet can provide assurance that the subnet may be deleted. But like IP address reclaim, multiple sample sets provide more robust confirmation of the reclaimable disposition. Just keep in mind that you'll rarely see zero responses on a subnet, at least while it's still provisioned on a router interface, so you'll want to check successive discovery results ignoring routers, switches, and perhaps other device types.

PERFORMANCE MANAGEMENT

Performance management involves the monitoring functions of the IP management system and, more importantly, the DHCP and DNS servers operating in the network. It's useful to track basic server statistics such as CPU utilization,

memory, disk, and network interface input/output (I/O). Such monitoring enables tracking of the hardware's ability to support the DHCP and DNS (and any other services) running on the server. Trending analysis in this regard is beneficial as well to enable proactive planning of future hardware procurements to enable load distribution among more servers.

Services Monitoring

Monitoring of the DNS service helps assure adequate DNS horsepower to meet the demands for name resolution, and to help identify any exception conditions. BIND supports flexible logging of a variety of event types to a configurable set of output destinations or channels, including syslog, file, null, or stderr (the operating system's standard error output destination). Microsoft supports the DNS server event viewer with settable severity level reporting and counters for total queries/second received and responses/second sent. DHCP servers likewise provide logging to monitor overall service health and statistics, typically to a log file, syslog, or an event log.

These measures enable collection of performance data from the server's perspective. However, they don't convey the services' performance as experienced by DHCP clients and DNS resolvers. Measuring client performance requires the remote issuance of a DNS query or Discover (or Solicit) packet and measuring the response time for receipt of a proper response. As mentioned in the fault management section, the absence of a response may indicate a services outage and should be investigated if it persists. This remote issuance could originate from services probes deployed in various locations to generate these "synthetic transactions," and measure and store response time results. Analyzing historical data from different probes can provide keen insight into DNS/DHCP services and network performance.

Address Capacity Management

Overall IP address capacity monitoring is another key performance management function within the scope of IPAM. Along the lines of inventory assurance, tracking address utilization from devices manually addressed, as well as those obtaining addresses via DHCP, enables proactive management of address space. Address allocations are initially based on estimated forecasts, which hopefully are accurate. Even when the forecast is perfect, however, IP network dynamics due to employee movement, large events, subscriber growth, and unplanned address demands can consume the entire capacity of a subnet and its address pools. Periodic monitoring of utilization levels on pools and subnets, with historical tracking and trending, can provide forewarning of a capacity crunch to trigger a supplemental allocation to expand capacity before it runs out.

Many DHCP server products enable monitoring of lease levels by command line, scripts, or SNMP. Microsoft DHCP also provides a general 90 percent alerting threshold, providing notification should an address pool reach 90 percent

capacity. Other servers and IP management systems provide similar or additional alert threshold definition and application. It's usually better to be notified by an IP management or network monitoring system than by irritated end users attempting to access the network.

Auditing and Reporting

Most management systems in general provide some level of auditing of "who did what" and varying levels of reporting. These functions, which could just as easily be categorized under Accounting Management, enable administrators to track and troubleshoot activity and to convey status information in report format. Auditing of IP address usage, that is, who had a given IP address at a certain point in time, is valuable information when troubleshooting a network issue or investigating potential illicit activity. Likewise, if you are attempting to track the history of IP address occupancy for a given device, reporting by hardware address is also beneficial.

Performing such auditing without an IP management system may be difficult except for the smallest of networks. Iterative dumps of DHCP lease data to track dynamically addressed clients over time are necessary. The ability to search for a given IP address requires access to a single (or two if failover or split scopes is in effect) DHCP server's lease history, while the search by hardware address necessitates searching across all DHCP servers, assuming the device is capable of mobility. If you have several DHCP servers, this may be a tedious task.

Common reports of interest for IP address planning include the following, though your system may provide different or additional reports.

- Address utilization report—by pool, subnet, block, rollups through hierarchy
- Address assignment report—summary of assigned addresses by subnet or block as current snapshot and/or history
- Address discrepancy report—highlights of discrepancies between the IP inventory and discovered IP address information
- DHCP performance report—summary and details of DHCP protocol messages by type and/or client and server key metrics summary
- DNS performance report—summary and details of queries by type, by querier, by question and server key metrics summary
- Audit reports—administrator activity, by subnet, by IP address, by hardware address, by resource record, by server

SECURITY MANAGEMENT

Several IP management vendors have attempted to jump on the "NAC" bandwagon, enabling IP planners to restrict IP address assignments only to valid devices or users, as determined by DHCP MAC address filtering or user login.

Such a solution needs to provide exception reporting and, ideally, alerting, should a number of access attempts by a device or user fail. One failed access attempt can likely be attributed to mistyping, but several failures may indicate an attacker attempting to crack the system to access the network. Of course, if the attacker knows the subnet address, he/she could manually configure an IP address to access the network, bypassing DHCP. This process is discussed in detail along with alternative approaches in Chapter 9.

Chapters 9 also addresses securing the information on and transactions with DHCP and DNS servers respectively. Securing the IP inventory and DHCP and DNS configuration data is also important, as this information is critical and should be protected from sabotage. Protecting IP data requires administrator access controls, enabling at least password-protected access to the information. If your organization has more than two or three administrators, a more sophisticated administrator security approach may be warranted with the use of an IPAM system. Most systems go beyond minimal password entry to enable scoping of administrator access by system function, by portions or certain elements of the network, by access type, for example, super user vs. read-only access and more. The sophistication of access controls will be driven by the number of administrators, their respective roles and responsibilities, and organizational security policies.

DISASTER RECOVERY/BUSINESS CONTINUITY

Business continuity practices seek to maintain the operation of the enterprise in the face of a major outage. A major outage or "disaster," implies that the sheer magnitude of the outage goes beyond a handful of servers or network devices. Automated and manual procedures must be documented in advance to reconfigure or redeploy resources to maintain operation of the network and applications, or at least the critical services and applications.

DHCP and DNS server redundancy features should provide network services continuity in the event of a server outage. On top of these technologies, deployment of DHCP and DNS appliances may provide an added layer of availability. While typically providing intra-site hardware redundancy, this is not normally considered for disaster recovery solutions due to co-location requirements. Appliance redundancy nevertheless provides a viable redundancy option, especially for critical servers, such as master DNS servers.

Business continuity of IPAM operations will likely require deployment of multiple IPAM databases. Deployment of multiple active databases or primary/backup configurations will depend on your selected vendor. Vendors implement a wide variety of approaches to facilitate redundancy, from full database copies and transfer, multi-master databases that require some level of network partitioning, to deployment of database replication technologies using storage area networks or SQL or LDAP replication capabilities. Operations tasks required to perform a disaster recovery will likewise vary by vendor.

Evaluation of vendor disaster recovery capabilities will depend on your business objectives, budget, and polices, but three key questions should be considered:

1. Is the IPAM database involved in name resolution or address assignment? For example, some systems route dynamic updates from DHCP to the IPAM database for uniqueness checks prior to routing to DNS. If the IPAM database is in such a "critical path," redundancy and high availability are paramount.
2. How frequently do your administrators make changes to IPAM data? The more frequent the rate of change, the more data changes may be lost between data synchronizations between the primary and backup database(s). A daily database backup may be acceptable in cases where changes are made infrequently, whereas a sub-daily or transactional replication process may be required for high rate-of-change environments.
3. What is the process to perform invocation of the backup system? Some vendors provide a relatively simple recovery procedure, while others require more manual intervention. Hopefully, disasters occur infrequently if never, but when needed, the failover process should be executable by staff on hand within time constraints defined by policy.

These basic questions are inter-dependent. If the answers to questions 1 and 2 are "yes" and "frequently," respectively, then the answer to question 3 should probably be "single step" or at least "very streamlined."

ITIL® PROCESS MAPPINGS

The IT Infrastructure Library* is a documented set of best practices for use by an IT organization desiring to manage, monitor, and continually improve IT services provided to the enterprise organization. ITIL was developed by the U.K. Office of Government and Commerce, and its IT-service oriented approach has been deployed by a number of organizations. The most common drivers for ITIL implementation include:

- Costs reduction of IT services delivery to the organization
- IT service level consistency and improvements
- Risk management through disciplined planning and evaluation of potential service-affecting changes
- Efficiencies in utilizing documented processes and continual improvement

Many of the functions within these process areas are identical or similar to those discussed earlier in the chapter, so we'll simply summarize these within their respective mappings to ITIL process areas.

*Details are available at the ITIL website, http://www.itil-officialsite.com/home/home.asp (119).

ITIL Process Areas

ITIL processes are split into two areas: service delivery and service support. Service delivery involves the planning, development, and deployment of IT services, while service support entails service operations in managing service levels and support. The Service Desk is a third process area that integrates with both service delivery and service support to provide a unified interface for IT to the rest of the organization. ITIL version 3 builds upon these sets of processes, with a couple of additions and functional splits.

Service Delivery. **Service level management** is a service delivery process area that encompasses the specification of service levels for various services provided by the IT organization. This is akin to a service level agreement (SLA). Service level management also includes measurement of service delivery against these specifications to monitor SLA adherence and measure the level of service that IT provides.

From an IPAM perspective, service level management involves definition and measurement of the levels of service provided to those requesting IPAM-related services, whether it be end users requesting an IP address or the business needing to open a new office. Treating the end user or the business in these cases as customers, this process seeks to gauge whether service delivery is meeting defined service levels, such as timeliness of completion of these requests. Automating IPAM-related service delivery, whether solely IPAM impacting or involving IPAM as part of a larger IT service such as VoIP deployment, facilitates timely and accurate services delivery. An example metric is the time frame by which an IP address will be assigned or a DNS resolution available.

Financial management naturally includes accounting, similar to accounting management in the FCAPS model, though the financial management area addresses actual dollars and cents as well. This process area also deals with any chargebacks or cost allocations for certain departments under an IT funding or cost allocation model.

Some firms do implement cost chargebacks for IP address usage. In such a scenario, the financial management processes would need to account for tracking of IP address usage along with the corresponding user and chargeable entity (e.g., department). Depending on the billing or chargeback cycle, this IP address use information will need to be stored for the current cycle and more to enable archiving or dispute resolution. Audits and history data in your IPAM system can be a big help with cost allocation.

Capacity management simply involves assuring adequate IT resources of the proper type are available for the business to conduct its work. Considering the application of this concept to IPAM, certainly IP address capacity management springs to mind, but one should also consider DHCP and DNS server load capacity. In the former case, capacity management requires monitoring of addresses and address pools to provide enough IP addresses for employees to get an address and access the network. Monitoring for trends is helpful, and enabling

alerting for low pools is also recommended for tighlty allocated networks. Of course, given the magnitude of IPv6 address space, this will likely not be an issue for IPv6 space for the foreseeable future.

With respect to server capacity management, monitoring each server's network, memory, and CPU utilization over time can provide insights into its performance. Such performance tasks may in fact be required as a linkage to service level management in terms of percentage of transaction completion (lease or resolution) as well as response times. Regardless, excessive loads on servers can have detrimental impacts on the availability of DNS and DHCP services, so server monitoring and perhaps even probe-like transactional monitoring can provide effective measures of service levels and capacity.

Availability management is a service delivery process area focused on making sure IT services are available to end users. High availability, a common goal for applications including DHCP and DNS, requires deployment of redundant configurations and the ability to leverage these configurations to provide continuous service in the face of a component outage.

Deployment of redundant DHCP and DNS server clusters, for example, appliances, can provide localized clustering, while implementation of DHCP failover or split-scopes and multi-server DNS deployments provides an additional layer of redundancy. Redundant IPAM database deployments through LDAP or replicated relational databases can also assure availability of the IPAM application for managing IP space. Monitoring of the availability of each of these redundant components enables proactive detection of outages to facilitate rapid outage resolution (mean time to repair) while redundant components shoulder the load.

Continuity management is related to availability management in that it deals with providing continuous services. For example, in the event of a disaster, this process area would require a disaster recovery plan be in place. As discussed in the business continuity section earlier, a variety of strategies are available based on the criticality and scope requirements of the organization for particular DHCP/DNS servers and IPAM systems.

Service Desk. Serving as the interface to the user community, the Service Desk filters input to the IT organization for incident reporting, change requests, and even new service requests. It serves to filter and direct user requests or problems to any one of the other ITIL areas, providing end users with a helpdesk function.

The policies and culture of the organization will drive whether the Service Desk performs traditional "level 1" support only by logging troubles with subsequent follow-up, or higher support levels up front to perform a level of diagnosis. In the case of level 1 support, little more is needed than a ticketing system with the ability to assign tickets to those responsible for other process areas depending on the caller's issue. A service desk staffed to perform some trouble diagnosis will require access to status monitoring tools to try to "see what the caller sees" with respect to the issue.

For IP address or name resolution related calls, providing service desk personnel access to IP inventory information may prove beneficial. For instance, if a person located in the Headquarters office is not able to get an IP address, the service desk needs to know the address plan for Headquarters in order to focus the problem and trouble resolution process on that particular subnet, associated routers, or DHCP/DNS servers.

The service desk is the interface not only for trouble reports but for change requests, such as IP subnet or address assignments. Providing service desk personnel with basic access to the IPAM system to request such changes, or better yet, enable end users themselves to register such service requests to an automated IT portal, can increase end users' satisfaction with IT services through rapid fulfillment. As we'll discuss in Chapter 7, such a portal interface with linkages to your IPAM system can streamline the service request process.

Service Support. **Incident management** is a service support process area that involves tracking and resolving incidents. In ITIL version 2, it also deals with change requests, whereas in version 3, these are split into separate process areas. As described above, the service desk receives incident and change requests from the end user community directly. Through network management, IT can also detect and troubleshoot network issues proactively. Regardless of the means of detection for a given incident, access to IP inventory data is indispensible to troubleshooting and incident resolution. In addition, monitoring of server states with thresholds, alerts, logging information, and audits can provide a head start to incident detection and management.

Service requests can be handled by service request tickets initiated via the service desk or IT web portal as described in the Service Desk section.

Problem management calls for the tracking of known problems and resolutions in a known problem database. If someone calls into the service desk with an incident, for example, it could get bumped over to problem management to identify whether this incident has been reported and addressed in the past. If so, the defined resolution path can be followed to quickly troubleshoot and resolve the issue.

While IPAM systems traditionally don't store problem histories with resolution annotations, some can provide a database of problem information through logging history, as well as inventory change audits. Network management system integration through APIs can provide a holistic view of problem history by providing IPAM data through the API to a trouble ticketing system, for example. IPAM is a key part of the overall network or IT service management approach, but it's not comprehensive; no system is. Having that integration is key to having a holistic view of the problem management scope.

Configuration management within ITIL is similar to the FCAPS configuration management functionality in terms of identifying, recording, and controlling configuration items affecting IT services. And, as we discussed extensively in this chapter, configuration management functions are a large focus area of IPAM processes. This includes configuring new address pools from a DHCP perspective,

zones and resource records in DNS, IP addresses for subnets on routers, etc. The IPAM database can be considered a configuration management database (CMDB) component of an IT's confederation of CMDBs for network configuration inventory.

Administrator controls need to be considered for organizations with more than one IPAM administrator to ensure that changes to DHCP and DNS configurations are done with the appropriate scope and permissions. For instance, you may want administrators to be able to make changes, but not actually deploy them on the DHCP and DNS servers, restricting that function to a higher level of administrator. On the back end, audit information is key for accountability tracking and reporting.

Possessing accurate IP configuration information is needed to provide a solid foundation on which future configuration changes can be planned. A corollary requirement leads to the necessity of validating that inventory against network actual data. An IP inventory tracked on a spreadsheet is great but requires constant updating. So having the ability to collect information from the network and then compare it with the plan is very important. Audits go hand in hand with inventory information collection. Arming the service desk with this information can provide a solid first line for addressing calls immediately, or to at least moving them through the process more quickly.

Change management provides controls on the implementation of changes in the IT infrastructure. This involves assuring that all affected parties are in agreement with respect to the scope and implementation timing of the proposed change. In terms of IPAM, the scope of change management commonly affects IPAM components, such as the addition of an address pool, deployment of a new DHCP/DNS server in the network, or upgrading a server to a new software version. Basically, anything affecting any part of the infrastructure, whether it's physical or software or even underlying appliance operating system, falls under the change management process, which seeks to assure all appropriate approvals are in place and corresponding back-out plans are available.

Release management is a service support process area that provides controls on deployed releases for hardware and software versions, not only for operating systems, but also for applications and appliances. This process area is responsible for making those versions available and accessible on the IT network and making sure there's an authorized set of releases and versions available that can be deployed appropriately.

Release planning, release management, dealing with upgrades, and patch management for DHCP and DNS servers from a central location can be a big timesaver. The alternative requiring on-site upgrades of operating system, patches, and application software is costly and time-consuming. Release management of the IPAM system also falls within this category.

7

IP ADDRESS MANAGEMENT WORKFLOW

Automation is among the key benefits of implementing an IP address management system. With increasing adoption of Internet-based technologies and applications, the networking industry has promoted convergence of applications running on IP networks to simplify and reduce resource impacts for networked applications and related support. Convergence in this sense may provide financial, efficiency, and productivity benefits for organizations, but it also "raises the bar" in terms of reliance on and visibility of IP network performance, resiliency, and incorporation into key business processes.

Automation of all or a portion of the IPAM functions and processes that we covered in Chapter 6 can provide lucrative benefits to organizations, including streamlining of IPAM process workflows themselves by reducing duplicate data entry and driving out configuration errors. While every organization may perform their processes and workflows a bit differently than others, we'll examine a set of common "primitives" with respect to automation of IP address management processes, then discuss additional opportunities for automation by way of example. The objective is to illustrate how incorporation of IP address management functions within a broader workflow process can produce efficiency gains and streamline processes and service delivery. We'll consider the dollars and cents quantification in Chapter 10.

Introduction to IP Address Management, By Timothy Rooney

WHAT IS WORKFLOW?

A workflow is a repeatable set of processes required to perform a function or unit of work. These processes are especially important when several different organizations are required to perform individual functions in a structured sequence in order to complete the overall work. In general, the more tasks and organizations involved within the workflow, the more complex it is. Usually, this complexity results in a longer time to complete the overall workflow and a greater need for process coordination. This is due to process hand-offs between organizations or process owners within the overall workflow. These hand-offs are usually performed manually and are communicated via telephone, email, or process review meetings as necessary. These forms of communications are typically ad hoc in nature and add additional time and variability to the defined processes within the workflow.

Simply documenting current processes in terms of steps or tasks, groups responsible for each task, as well as task input and output requirements can be enlightening in defining processes and identifying possible improvements. Another benefit of documenting processes is that it can help identify approval points in the process. Such points enable review of the proposed task output or change for approval, modification, or disapproval. Change control teams, for example, are commonly inserted into key workflows to assure adequate communication and change approval. An expedited approval process can be defined to address cases of urgent change requests needed to resolve configuration or capacity issues.

Example Block Allocation Workflow

Let's examine a sample workflow, considering Figure 7-1, which illustrates a simple block allocation workflow. In this example, three different organizations

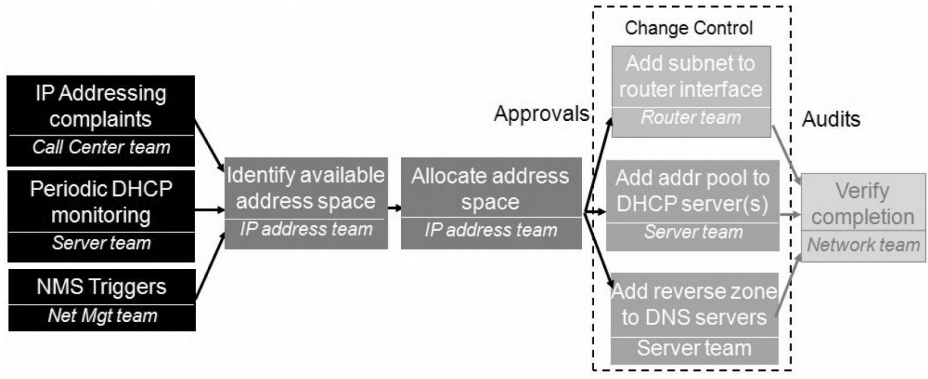

Figure 7-1: Address block allocation workflow example (32)

or entities may initiate the workflow as shown at the left of the figure. For instance, the help desk or call center team may receive a complaint about an inability to obtain an IP address, the server team within Operations or IT may notice that a DHCP server is at full capacity for one or more address pools, and/or the network management team may receive an alert from a network management system regarding a DHCP server capacity outage.

These initial notifications to the IP address team, which is responsible for IP address allocation, may be in the form of phone calls or emails. Upon receipt of the notification, the IP address team would need to locate the affected address pools and/or subnets, and identify available address space that could be assigned to the location to supplement the address capacity. Following the subnet allocation IPAM process we reviewed in Chapter 6, the IP address team would then allocate the identified address space, say 10.2.3.0/24, to the location. This requires updating of the IP address assignment database, spreadsheet, or IP management tool. There will likely be additional information to be defined and tracked with the assigned subnet, such any DHCP address pools, as well as host and domain names.

Once the IP address block has been allocated and recorded, its approval through a change control process may be necessary. Once approvals have been granted, the subnet needs to be deployed on the IP network. This deployment typically involves adding the block or subnet to the router or edge device serving the affected location. The router team is often an organization independent of the IP address team, so an inter-organizational communiqué would typically be issued to request the assignment of the subnet to the appropriate router interface. In parallel with this router configuration step, the DHCP server(s) serving devices at the location must be configured with the corresponding address pools within the allocated subnet. For example, allocation of a /24 network with 254 usable IP addresses may comprise 43 static devices and 211 dynamic or pool addresses. As we discussed in Chapter 6, the pool portion of the subnet must be configured in the DHCP server so it can service clients requesting space from the pool. The DHCP server team would perform this task based on an explicit communiqué from the IP address team with appropriate approvals.

Notice that a different set of information would be conveyed to the router team (add 10.2.3.0/24 to router X interface s1) than to the DHCP server team (add pool 10.2.3.44–10.2.3.254 to DHCP server Y). Additional information may be required for complete DHCP server configuration, including associated DHCP options, policies, and client classes required for the address pool. Meanwhile, a third communiqué containing a third set of information would be communicated to the DNS server team: create appropriate A records in the forward zone and create the 3.2.10.in-addr.arpa zone file and add pointer (PTR) records to it on master DNS server Z for static addresses 10.2.3.1–10.2.3.43 each with enumerated fully qualified domain names. Again, additional information may be conveyed to completely configure the DNS server for the network, including pre-population of resource records for the dynamic addresses and association of the subnet with a DNS view, for example.

Each of these three parallel processes must be completed to proceed to the next step, verification, or audit of completion by the network team and/or IP address team to close out the process. As the router team and DHCP and DNS server teams complete their respective work, notification must be provided to the network or IP team. Upon receiving confirmation from all three organizations, the network or IP team can verify proper configuration through reports, audit trails, or actual network testing and discovery. At this point, our sample workflow for allocating an address block is complete.

Streamlined Block Allocation Workflow

Our example block allocation workflow in Figure 7-1 is a relatively simple, high-level workflow. It's likely that each process illustrated in the figure itself involves several sub-processes. But even this high-level flow illustrates a key point: ad hoc communiqués among organizations are not usually efficient and generally introduce a delay in completion of the overall workflow. This delay stems from time lags in remembering to send and actually sending the communiqué from the source organization, and in receiving the communiqué (e.g., picking up emails and voicemails if not directly accessible) at the destination organization. Processes such as the network verification step in Figure 7-1 require multiple parallel processes to complete, and introduce an additional complexity in tracking acceptable input from each preceding process prior to beginning the next step. Required change control approval steps may also introduce delays in convening approval meetings, presenting the case, and garnering the approval. Delays may also result within each high-level process defined, which we will address later.

First, let's examine one method of simplifying this workflow by introducing automated inter-process communiqués via electronic messages such as intersystem API calls. Figure 7-2 illustrates the same process steps as in Figure 7-1, but by employing electronic interfaces between processes, the overall workflow can be streamlined from an overall time perspective as well as manual effort standpoint.

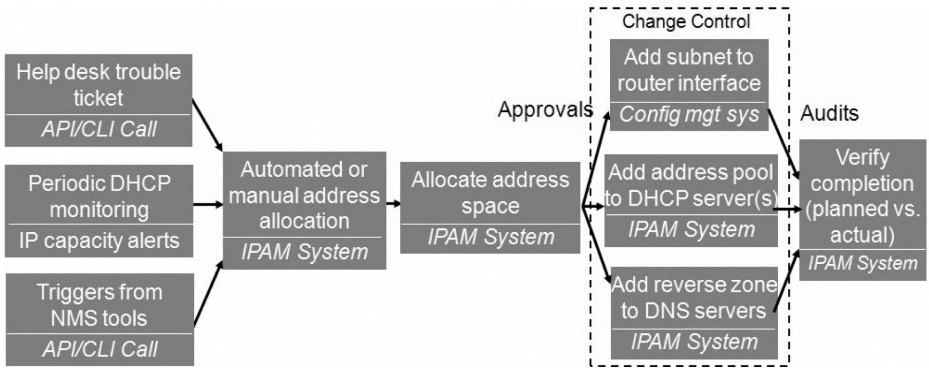

Figure 7-2: Fully automated workflow via electronic interfaces (32)

In Figure 7-2, the same three input processes may kick-off the address identification and allocation processes. In this streamlined process example, however, the help desk call or email to the IP address team is replaced by an API call from the call center's trouble ticketing system to the IP address management (IPAM) system. This call may request the IPAM system to automatically identify and pre-allocate or reserve an address block. Even if this call is a system-generated email to the IP address team, overall workflow time is reduced by eliminating the lag in the sender's remembering to send, constructing the message, and sending it. If the server team is utilizing a DHCP pool monitoring mechanism with capacity alerts, or if the IPAM system supports such monitoring and alerts, these alerts may trigger the IPAM system to take the next step in defining appropriate address space. Likewise, electronic triggers from a network management system (NMS) may invoke similar action.

Any of these triggers may stimulate the IPAM system to notify an administrator or even reserve or suggest an appropriate address block for allocation to the location in need. Hence the IP address team process may be configured for either notification only, for notification with reservation of address space pending team approval, or notification and automated identification and allocation of appropriate address space. The degree of such automation is dependent on organizational policies for such automation and the capabilities of the IPAM system itself to perform these steps, as we'll discuss in the next section.

Whether the address space is allocated manually by the IP address team via the IPAM system or the IPAM system automatically allocates a block, the IPAM system must communicate this information to requisite downstream processes. A "pending approval" state may be required to halt the process until the change control approval has been granted. Once approved, communication of the allocation to the router team may be in the form of an IPAM system-initiated email to the router team or by a "callout" from the IPAM system to the router configuration management system upon block allocation. In the former case, the router team must receive the email and perform the corresponding process; in the latter case, the callout from the IPAM system could call a script to create a work ticket or directly enter an API call to a configuration management system or router.

The updating of associated DHCP and DNS servers traditionally falls into the purview of the IPAM system itself. In this case, assuming configuration information at the level required for DNS and DHCP server updates is spawned during the block allocation process, and that the IPAM system can automatically or periodically update DHCP and DNS servers, then these processes can be fully automated. This level of automation saves time and effort not only in communicating the block information and associated DHCP and DNS information to each server team respectively; this can save time and effort in otherwise performing these tasks by manually configuring DHCP and DNS configuration files via text editors. This "intra-IPAM" system efficiency can reduce the duration of these processes and thereby contribute time savings to the overall workflow for all such IPAM functions. This automation, of course, is one of the key benefits of implementing an IPAM system and, as we'll discuss in Chapter 10, the system chosen

for deployment should be selected based on the features that it provides to minimize your costliest IPAM functions.

The final confirmation step in the workflow may also be automated to a large degree depending on the router configuration process and the capabilities of the IPAM system in use. If the router team communicates via email after completing its process, there is little efficiency gain. However, if the router team uses a configuration management system to configure the routers, and the configuration management system enables callouts upon provisioning completion, for example, the callout could be used to confirm completion to the IPAM system electronically. Likewise, confirmation of completion of DHCP and DNS server configuration updates can be streamlined when the IPAM system performs and records completion of these updates. Recording of the update information enables confirmation and audits of changes and change history. IP discovery provides an additional level of verification as well.

WORKFLOW REALIZATION

As you've probably noticed, there are a lot of "ifs" in the description above. This variability, on one hand, highlights the flexibility of defining numerous combinations of workflows to adapt to your policies and infrastructure capabilities, but, on the other hand, renders definition of cookie-cutter workflows rather difficult. The extent of your realization of these and other workflows and their corresponding benefits will depend upon the integration capabilities of your IP address management system, the mating capabilities of other systems with which integration is desired, and your policies, procedures, and desire to integrate. For example, in the streamlined block allocation process described above and outlined in Figure 7-2, the following capabilities would be required:

- Trouble ticketing system with the ability to automatically generate and email or issue an API call to the IPAM system.
- Network management system with the ability to issue an API call to the IPAM system based on definable events, such as address pool depletions; alternatively, if the IPAM system itself provides event triggers, the system must be able to initiate some action based on this trigger, namely the initiation of the block allocation process for the affected subnet or location.
- IPAM system with the ability to:
  - Receive API calls to initiate the block allocation process, which may entail reservation of address space or actual allocation based on organizational policies
  - Utilize addressing templates when allocating the block to assign addresses and address pools based on a predefined template
  - Trigger downstream events such as passing the allocated block information to a work order system or router configuration management system to enable router provisioning

    - Update relevant DNS and DHCP servers with information associated with the allocation automatically or based on administrator approval
- A work order or configuration management system with the ability to receive API calls to create work order tickets or to automate router provisioning, respectively.

With these components in place with the required capabilities, this streamlined workflow can be implemented, reducing the time required and minimizing data entry errors along the path of the workflow. While component system capabilities are required to fully automate this process, organizational policies or methods of procedure may dictate a semi-automated flow with one or more approval points.

For example, let's bring in the change control approval point prior to deployment of the subnet allocation. If the IPAM system can reserve the subnet intended for allocation and this pending allocation information can be summarized for presentation to the change control approval parties,* inserting this approval point is easily accomplished. Once approved, an authorized administrator can modify the state of the allocated subnet from a reserved or pending state to a deployed or allocated state. Assuming this state change could then trigger associated downstream events in the flow, automation may ensue to complete the workflow.

The bottom line is that you should consider the capabilities of your component systems and mask those with required policies and procedures. We'll provide examples of time-saving automated workflows later in this chapter, but keep in mind that these may need to be adapted to your systems' capabilities and organizational methods and politics.

WORKFLOW BENEFITS

Despite the variability of workflows just described, we'll discuss the automation of workflows for IPAM functions (intra-IPAM) as well as those peripherally impacting IPAM (extra-IPAM) functions, because they represent a substantial opportunity for reaping the following key benefits for your organization:

- Improve operational efficiencies through automation
    - Reduce manual effort
    - Shorten process intervals
    - Minimize costs
    - Improve end user satisfaction
- Provide process control points

*In this context, we mean presentation *to* the parties involved in change control, not *at* parties to discuss change control, which certainly might make such meetings infinitely more enjoyable!

  - Provide opportunities for approvals, as well as annotations and escalations
  - Measure intervals and other metrics for process improvement and service level agreements
- Enable quicker implementation of new IP-related offerings
  - Introduce new capabilities or services to associates or customers
  - Build upon existing workflows to improve services
  - Develop new flows associated with new offerings
- Facilitate continuous improvement
  - Feedback and metrics facilitate process improvement

Tips for Defining Workflows

First of all, you'll need to have a good handle on the capabilities of your IPAM system and those with which you'd like to integrate. We've also mentioned the impact of organizational policies, procedures, and politics in defining automation break points for approvals; the break points may also be necessary for inter-organizational (e.g., inter-business unit or department) hand-offs. Having such hand-offs occur behind the scenes electronically may not be acceptable, at least until the automation can be "trusted," meaning that it has been tested to the satisfaction of both ends of the hand-off. This highlights the fact that with any change in process, some resistance to change may naturally occur. Prove-in of benefits can go a long way to working through this resistance, though it may inevitably take some time.

Stepping back, the first thing to do is to understand your current processes, intra- and inter-organizational, for performing certain tasks. You can start with the list of discrete tasks we described in Chapter 6 as well as the examples later in this chapter. These define common processes, though your team may perform these tasks differently. Thus steps may be omitted, inserted, or re-ordered in accordance with the way you do business. Mapping out current processes provides a baseline from which process improvement or automation may be applied. Look for task transitions and dependencies. Can information between tasks be better conveyed electronically? Can other forms of automation, such as alerting, provide additional streamlining opportunities? When asking these questions, consider again the capabilities of your IP address management and associated systems. How can the automation of a particular task be implemented? If your vendor does not support a particular feature today, will they in the future? If so, when? Provide your input to the vendor, as this may help other customers of the same product attempting similar implementations.

It's a good idea to start small to prove in a workflow concept, and obtain measurable productivity or process time saving metrics. Such hard data provides the ammunition needed to further automate other tasks, especially when attempting to cross organizational lines. Potential workflow scenarios you may find useful are discussed next.

WORKFLOW SCENARIOS

Recognizing that most organizations seek to continuously improve operational efficiencies and desire to integrate IPAM processes into broader IT and Operations workflows in general or as part of a broader initiative such as ITIL®, the use of an IP address management system can facilitate automation of IPAM workflows. Many IPAM products provide one or more interfaces for electronic communiqués for workflow automation, typically including command line interfaces (CLIs) and an application programming interface (API). These interface points facilitate scripting of multiple tasks as well as integration with other management systems needing to initiate actions or obtain information from the IPAM system. Some systems also provide downstream indicators for actions taken within the IPAM system itself. For example, when a device or subnet is created, edited, or deleted, these systems enable the passing of relevant data fields to another management system. In this manner, the IPAM system provides integration points to externally initiate IPAM functions from an upstream system and to enable IPAM system-initiated functions to commence further actions downstream in the workflow.

Intra-IPAM Workflows

Before continuing with broad workflow examples involving multiple systems and functions, let's consider automation of some of the IPAM process we discussed in Chapter 6. For a variety of reasons, inter-system workflows may not be feasible or desired. Even if desired, evolution to full workflow automation may take time to validate proper configuration and "trust" that the automated workflow will perform automatically. Nonetheless, many of the benefits described above are realizable even when automating intra-IPAM related workflows, so we will consider these first, then expand our set of examples to broader possibilities.

For these scenarios, the input mechanism could simply be the IPAM system native user interface, its CLI or API interfaces, or even a more abstract form of input: a web-based IPAM portal. The IT or Operations organization can create a web portal or expand an existing one for distributed administrators or even end users to log requests for IP subnets, addresses, hostnames, IPAM-related information, and more. The benefits of creating such a portal include the following:

- Offload the processing of common, more mundane tasks from the IP management team, enabling them to focus on issues requiring their higher level of expertise. This promotes resource efficiency and can improve job satisfaction for members of the IP management team by providing focus on more challenging work.
- Empower distributed administrators or end users with the ability to log IP address-related requests and obtain immediate or deferred (e.g., upon approval) feedback and resolution. Empowerment and rapid feedback elevates end user satisfaction and productivity.

- Maintain IP address management data integrity with the ability to log requests and make changes using a variety of user interfaces, while enforcing end user access controls to viewing or modifying such data within the centralized repository.
- Enables the IP management team to retain overall control of the information in the IP address management system, while exposing only those parameters or functions that are required for distributed administrators. In other words, the IP management team can minimize the ability of end users to enter "dangerous" data that could have ripple effects in the system by virtue of what features are exposed on the portal input screens. This abstract user interface layer on top of the IPAM system is also useful if the granularity of access controls is insufficient on the native IP address management system.

You can use your favorite web technology (e.g., JavaScript, Perl/CGI, PHP, etc.) to create the portal input screens or link off an existing IT portal to process user requests and perform API calls into the IPAM system. The web request page can be created to define input parameters that are germane to your conduct of business. A script can be processed upon submitting the web request page to parse the input parameters and interface to the IPAM system. For example, a web page that requests a user ID and password along with location, subnet size needed, description, and purpose would be appropriate for a subnet request screen. The script called from clicking the "submit" button can either pass entered information directly to the IPAM system, or it can be used to authenticate the requestor, then replaced by an employee ID that can be mapped to a "group" user ID in the IPAM system with permissions to make such requests. The other input fields can be mapped directly to the IPAM system API call to log the request. Error processing on the API call also needs to be coded to provide direct user feedback regarding the status of the request such as: entry error, rejection, success, pending approval, etc.

Keep in mind that there are many other ways of handling the mapping of the information you need to collect on the web request page to what your IPAM system needs to process the request. This enables maximum flexibility to adapting the request information to your organization's policies.

DHCP Server Configuration. We consider this flow as intra-IPAM as it entails automated creation of a DHCP server configuration file or command set based on the IP address plan, and implementing this on a distributed DHCP server. Many IPAM systems include this inherently, but if you're using spreadsheets, an in-house system, or other less integrated IPAM systems, this workflow could save time and reduce configuration errors. A DHCP server's configuration consists of IP address pools, individual assignments for M-DHCP clients, associated option settings, potentially failover configuration, and other functional parameter settings. Certainly the address pools and individual assignments must

be linked tightly with the overall IP address plan. The other configuration aspects listed may also be definable within the IPAM system or by using a script that creates these configuration directives and transfers them to or executes them on the corresponding DHCP server.

A deployment approval point in the process may be appropriate, whereby delegated administrators make proposed changes in DHCP configuration using the IPAM system, but such changes are not deployed to respective DHCP servers without review and approval by a senior administrator or the change control team. A means of viewing proposed changes would be helpful in the review and approval process, whether provided natively by the IPAM system or by an external method. This workflow example could be considered a building block to our subnet allocation example illustrated in Figure 7-2, if such a subnet contained a DHCP pool or M-DHCP objects.

DNS Server Configuration. As with DHCP server configuration, the automated generation of configuration files or commands for DNS server updates can help save time. Leveraging linkages with the IP address plan can reduce entry errors and facilitate data consistency. As you assign IP addresses, you may in most cases also need to update corresponding forward and reverse DNS zones associated with the IP address and domain name. And as with the DHCP example, a deployment approval process may be desirable to control changes impacting DNS configuration. This workflow scenario can also be considered a building block to the prior subnet allocation workflow example of Figure 7-2.

Subnet Assignment. Let's now adapt our earlier example relating to allocation of a subnet in reaction to another event, and apply it to a proactive intra-IPAM flow. In this case, an associate in the field can request a subnet via the IPAM portal. Considering the flow depicted in Figure 7-3, the end user accesses the IPAM portal to submit a request for an address block or subnet. The request is logged by the portal and a subnet is allocated directly via the IPAM system using an API call from the web script to the IPAM system. The portal should indicate success or failure status of the request, ideally with the assigned subnet within successful results to provide the immediate feedback to the requestor.

Alternatively, if the request requires common IP address assignments within the subnet given its purpose, then the IPAM system can also perform individual address assignments within the subnet. This might be the case when opening a

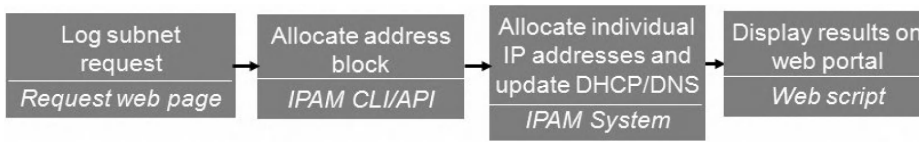

Figure 7-3: Portal-based IP subnet allocation flow

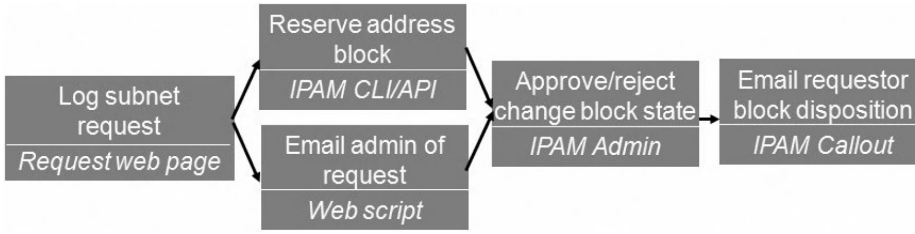

Figure 7-4: Portal-based IP block reservation with approval flow (32)

new retail store, branch office, distribution center, etc. For each of these "types" of locations, many have common IP address subnet sizes and associated address layouts (e.g., IP addresses 1–3 are for routers, 4–8 for printers, etc.). This template-driven allocation enables one-click subnet allocation with sets of individual address assignments. The DHCP and DNS server configuration flows may be used within this subnet allocation flow to complete the network services configuration associated with the added subnet.

Another variation on this flow provides an approval-based flow, if you'd prefer the ability to review and approve such requests manually. In this case, shown in Figure 7-4, the request is logged and the IPAM system's API is utilized to reserve a subnet and send an email to one or more IPAM system administrators. The email can be directed to a standard email server or to a cell phone or pager depending on how quickly you'd like to respond to these requests, and can pass the requestor's identity as well as location, size requested, and other parameters posted on your portal page. Using this information, the IPAM system administrator can login to the IPAM system, locate the reserved address space and either approve the request by changing its status from reserved to approved or reject it by deleting or freeing up the block. Once the administrator commits this state change, ideally the system can transmit an automated email to the requestor notifying him or her of the disposition of the request, and the resulting allocation if approved.

IP Address Assignment Request. Another popular time-saving workflow features the logging and creation of individual IP address assignments. If your IP management team spends a lot of time responding to phone calls or emails requesting individual IP addresses, this flow can be very useful. Create a web portal page to enable end users to identify who and where they are, then simply request an IP address. Additional input criteria may be desired to declare the purpose of the request, to identify the particular device type, for example, a printer, to enable entry of hostname information for creation of A/AAAA and PTR resource records, and even to define whether the device address will be manually (statically) configured or obtained via Manual DHCP (in this case, the MAC address will also be a required input field).

Extra-IPAM Workflow Examples

The intra-IPAM workflows defined above can be leveraged to enable their insertion into broader workflows involving additional systems beyond the IPAM system. We've already illustrated the block allocation flow with the IPAM system receiving input from an upstream trouble ticketing or network management system and providing output downstream to a work order or configuration management system. In this case we essentially interchange the web portal input upstream with the ticketing or management system. Creation of such task routines, comprising discrete sets of tasks to perform a particular function, enables re-use among many workflow variations. This in itself saves work in redefining flows, creating new ones, and executing flow updates and improvements. Therefore, it is conceivable that sets of example flows discussed in this chapter can be joined together to create larger macro flows. Again, we recommend you start small, but take heart in knowing that simple flows will likely be reusable when building larger ones.

Regional Internet Registry Reporting. Service providers and some enterprises obtain additional address space directly from Regional Internet Registries (RIRs) such as the American Registry for Internet Numbers (ARIN) and Réseaux IP Européens (RIPE). These and other RIRs have defined policies and automated email interfaces to enable IP address consumers to notify them upon allocation of address space internally or to their customers. Hence, as an additional step added to the block allocation flow previously discussed, the allocation information could be automatically conveyed to the corresponding RIR in the format required. This would typically require inclusion of the address block allocated, a network name, and additional contact information associated with the block. Therefore, this form of workflow would generally be a function of the IPAM system, post-processing each block allocation, edit, or delete. In this manner, the RIR is informed of address space utilization over time. When additional address space is needed, the request process with the RIR can be expedited as the registry already has much of the supporting information. This ongoing reporting also saves the otherwise manual effort required to identify each block allocation and its respective utilization in order to justify the request for additional address space.

Router Configuration Provisioning. We hinted at this workflow as part of our initial block allocation example. If you're inventorying router devices and associated subnet assignments, this workflow may come in handy. In some organizations, politics may disqualify such an automated provisioning workflow. Those responsible for router configurations may need to retain control and/or perform updates at specified times. In this situation, the work order creation flow may be a better choice. In either case, information pertaining to the block allocated and its association with a router and interface information would be passed

to a router configuration management tool or to a work order creation system, respectively. This automation can help reduce manual transcription of IP subnet information for entry into a router command line or configuration tool, especially when considering IPv6 subnets!

Customer Provisioning. Service providers offering IP services from ISPs to broadband providers to managed services providers typically allocate an IP address, several IP addresses, or an address block to their customers. The block or IP address assignment process applies, but most likely billing and support systems must also be provisioned. Entry of new customer information into a customer relationship management (CRM) or billing system would be ideally performed using one interface, with the associated external system interfaces enabling automation among systems, including the IPAM system. As billing systems would retain the plan of record for customer billing information, the IPAM system would need to be updated to maintain the plan of record for the IP address plan.

Asset Inventory Integration. Many organizations employ an asset management system to track valuable assets such as servers, laptops, printers, VoIP phones, and so on. Many of these systems are keyed by an asset identifier and not IP address. However, some IPAM systems can facilitate integration between asset and IP inventory systems with the use of its integration points and user-defined fields. If a given asset ID is a field that can be entered for an IP address, your IPAM system may be able to pass this information downstream to the asset management system using its CLI/API. This workflow can eliminate the duplicate entry of information in two different systems. If desired, the converse flow can also be implemented.

Another form of asset-IPAM system linkage can be achieved with the use of URL-based data entry fields or even SQL, or LDAP lookups. If the asset system provides URL, SQL or LDAP reachable access, such a linkage could be defined in your IPAM and/or asset system to eliminate duplication of information entry across systems. Thus, not only can the creation process be streamlined, but the inter-system navigation can be as well.

Trouble Ticket Creation. If your IPAM system tracks DHCP address utilization data from the DHCP servers running across your network, it may also provide fixed or user definable thresholds to trigger alerts. If these alerts enable triggering of downstream events, you may be able to leverage this automated trigger point. Should a threshold be triggered, you could utilize this mechanism to automatically create a trouble ticket or otherwise notify relevant personnel. This notification is very helpful in identifying pending address depletions before they actually occur. Once such a depletion occurs, end users will be affected and the help desk phones will ring.

Opening a New Office or Retail Store. In an automated scenario, a one-button process could be developed for defining and assigning IP addresses and

other elements required for a new office location. Many large retailers employ this type of workflow to maintain consistency and simplify administration. This workflow could include the full block allocation flow, including individual address assignments along with relevant DHCP/DNS server configuration. It may also include automated multiple block allocations for different applications, one for management, one for VoIP, one for cash registers, etc. Different store "templates" can be useful to streamline the provisioning of address space and network services for different types or sizes of stores or offices.

SUMMARY

Automating workflows may provide substantial savings of time, money, and effort within your organization. Depending on your current business processes and the tools you use or plan to use, including IPAM tools, workflows can be defined and implemented to automate common functions to streamline IPAM-related and extra-IPAM processes.

8

IPv6 DEPLOYMENT AND IPv4 CO-EXISTENCE

IPv6 was originally specified in the mid-1990s to address a then urgent need to supplement the rapidly diminishing IPv4 address space. At the time work was begun in earnest on defining IPv6 or IPng (IP next generation), as it was initially called, the Internet was just starting to catch on with the general public. More and more enterprises were expanding their internal networks to enable connection to the global Internet. Since every reachable host required a unique public IPv4 address, the demand for addresses skyrocketed.

While these events spurred the development of IPv6, they also stimulated the implementation of other technologies that prolonged the life expectancy of IPv4 address space. As we discussed in Chapter 2, Classless Inter-Domain Routing (CIDR) enabled Regional Internet Registries and ISPs to allocate address space more efficiently than with former classful-only allocation methods. For example, instead of allocating an entire /16 network to a client requiring 2000 IP addresses, the RIR or ISP could allocate a /21 network. Given that there are $2^{(21-16)} = 32$ /21s in a /16, 32 similarly sized customer requests could be handled from the address space that may have formerly been distributed to one.

Another IPv4 allocation strategy discussed in Chapter 2 that vastly reduced the amount of address space required by organizations from RIRs or ISPs was the allocation of private address space. Defined in RFC 1918, the allocation of

Introduction to IP Address Management, By Timothy Rooney

private networks enabled every organization to use the same address space for their internal networks. Communicating to Internet hosts or among organizations still required public addresses, but the use of network address translation (NAT) firewalls provided private-to-public address translation for internal hosts accessing the Internet.

One could also argue that DHCP itself enabled better utilization of address space, with its ability to share addresses among a number of users on an as-needed basis. While predominantly configured with private address space within organizations having little impact on public space, DHCP is also used by broadband and wireless service providers to enable Internet access for their respective subscriber bases.

These growing subscriber bases are still driving increasing IPv4 address consumption, diminishing available capacity. And in many parts of the world, current allocations of IPv4 address space are inadequate. For example, Asia has been allocated only 20 percent of IPv4 space but contains half of the world's population! Asia has been among the leaders in deployments of IPv6, followed by Europe. While North America has enjoyed relatively plentiful IPv4 address space, many organizations are evaluating a move to IPv6, especially among government and service provider organizations.*

WHY IMPLEMENT IPv6?

On May 22, 2007, the American Registry of Internet Numbers (ARIN—one of the RIRs) board issued a public statement "advis[ing] the Internet community that migration to IPv6 numbering resources is necessary for any applications which require ongoing availability from ARIN of contiguous IP numbering resources." This in essence stated that IPv4 numbering resources (including IPv4 addresses) are becoming more scarce and that entities (LIRs, ISPs) requiring additional addresses over time need to plan for IPv6. Some estimates predict depletion of IPv4 space at the RIR level some time around 2012. In fact, Dr. Geoff Huston of APNIC publishes an enlightening analysis, updated daily, at ipv4.potaroo.net (33) lending evidence to these dire predictions. While one can surmise the exhaustion date of IPv4 address space from this analysis, the intent of Dr. Huston's analysis focuses more on when RIR policies must be in place to deal with the new RIR role for IPv4 address space, including no longer allocating IPv4 space.

Nevertheless, upon this final RIR allocation of IPv4 space in 2012 or whenever this occurs, ISPs will still have space to allocate; the last block allocated by an RIR will exhaust RIR space, but this allocated ISP block is still available for ISP allocation. ISP space will deplete when the ISP customer base fully consumes their allocatable space, which could take a couple years after that. Enterprises that haven't submitted nor have plans to submit requests for IP addresses in the

*Material in this chapter is based on (117).

foreseeable future may conclude that IPv6 won't affect them. But at the point in time where only IPv6 space is available, new ISPs or organizations seeking to expand address space will generate end users with IPv6-only connectivity. This will require IPv4-only organizations to implement IPv6 on external (Internet-facing) web, email, and other public servers as this IPv6-only population grows.

In addition, like many IP network changes such as those driven by wireless and PDA adoption within organizations, IPv6 may end up being driven by employees connecting to the network via next generation cell phones, PDAs, or even with Windows Vista (or 7) via network or home connections! In this case, the requirement to manage IPv6 address space may be thrust upon IT managers. Whether driven by external IPv6-only users or internal users, it's more prudent to proactively plan now for IPv6, whether you plan to fully migrate your network or plan to deal with individual IPv6 devices attempting to connect.

IPv4-IPv6 Co-Existence Technologies

Numerous technologies are available to facilitate the migration of devices to IPv6. The approaches will be discussed according to the following basic categories:

- Dual stack—support of both IPv4 and IPv6 on network devices
- Tunneling—encapsulation of an IPv6 packet within an IPv4 packet for transmission over an IPv4 network or vice-versa
- Translation—IP header, address, and/or port translation such as that performed by gateway or NAT devices

The selected strategy requires effective coordination of the following:

- IPv4 and IPv6 network and subnet allocations, existing and planned
- DNS resource record configuration corresponding to appropriate name resolution to address(es) for desired tunneling or translation
- Compatible client/host and router support of selected tunneling mode as appropriate
- Deployment of translation gateway(s) as appropriate

DUAL STACK APPROACH

The dual stack approach consists of implementing both IPv4 and IPv6 protocol stacks on devices requiring access to both network layer technologies, including routers, other infrastructure devices, application servers, and end user devices. Such devices would be configured with both IPv4 and IPv6 addresses, and they may obtain these addresses via methods defined for the respective protocols as enabled by administrators. For example, an IPv4 address may be obtained via DHCPv4, while the IPv6 address may be autoconfigured.

Implementations may vary with dual stack approaches with respect to the scope of the stack that is shared vs. what is unique to each IP version. Ideally, only the network layer would be dualized, using a common application, transport, and data link layer. This is the approach implemented in Microsoft Vista, as opposed to the XP implementation, which utilized dual transport and network layers, requiring in some cases redundant configuration of each stack. Other approaches may span the entire stack, down to the physical layer requiring a separate network interface for IPv6 vs. IPv4. This approach, while contrary to the benefits of a layered protocol model, may be intentional and even desirable, especially in the case of network servers with multiple applications or services, some of which intentionally support only one version or the other.

Dual Stack Deployment

Deployment of dual stacked devices sharing a common physical network interface implies the operation of both IPv4 and IPv6 over the same physical link. After all, Ethernet and other layer 2 technologies support either IPv4 or IPv6 payload thanks to protocol layering. Dual stacked devices require routers supporting such links to be dual stacked as well. This overlay approach is expected to be very common in IPv6 deployments and is depicted in Figure 8-1. This diagram can be extended beyond a physical LAN to a multi-hop network where routers support IPv4 and IPv6 and route IPv4 packets among native IPv4 hosts and IPv6 packets among IPv6-capable hosts.

DNS Considerations

As we shall see, DNS plays a crucial role in proper operation of each transition technology; after all, it provides the vital linkage between end user naming, for example, website address at the application layer and the destination IP address, whether IPv4 or IPv6 at the network layer. End users attempting to access a dual

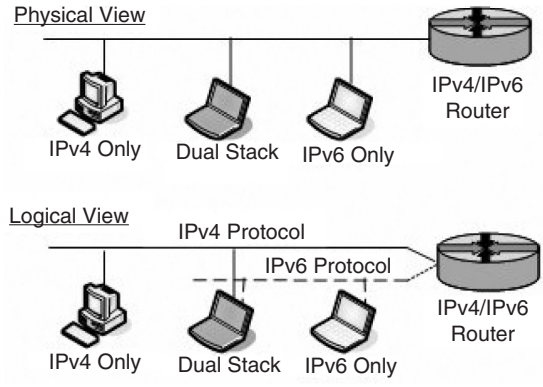

Figure 8-1: Dual stacked network perspectives

stack device will query DNS, which can be configured by administrators with an A resource record corresponding to the node's IPv4 address and a AAAA resource record corresponding to its IPv6 address. The owner field of the resource record may have a common host name corresponding to the device per the following example:

```
dual-stack-host.ipamworldwide.com. 86400 IN  A     10.200.0.16
dual-stack-host.ipamworldwide.com. 86400 IN  AAAA  2001:DB8:2200::A
```

Resolution of IP-address-to-host name may also be configured in DNS within the appropriate .arpa domain:

```
16.0.200.10.in-addr.arpa. 86400 IN PTR dual-stack-host.ipamworldwide.com.
A.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.2.8.B.D.0.1.0.0.2.ip6.arpa.
86400   IN   PTR   dual-stack-host.ipamworldwide.com.
```

A dual stack node itself must be able to support query and receipt of A and AAAA records during its own DNS resolution processing, and communicate with the intended destination using the address and protocol corresponding to the returned record. Some resolver configurations may enable definition of the preferred network protocol when both an A and AAAA record are returned from the query, not to mention the protocol to use when issuing DNS queries themselves. In addition, as we shall see, some automatic tunneling technologies utilize specific IPv6 address formats, so addresses corresponding to one or more tunneled address formats may also be returned and may be used to the extent that the resolving host supports the corresponding tunneling technology.

In terms of the IP version used in the transport of DNS queries and answers, RFC 3901 (34) (Internet Best Current Practice 91) recommends that each recursive DNS server should support IPv4-only or dual-stack IPv4/IPv6. The RFC also recommends that every DNS zone should be served by at least one IPv4-reachable authoritative DNS server. These recommendations were set forth to provide backward compatibilty for IPv4-only resolvers which will be around for quite some time.

DHCP Considerations

The mechanism for using DHCP under a dual stack implementation is simply that each stack use its version of DHCP. That is, to obtain an IPv4 address, use DHCP; to obtain an IPv6 address or prefix, use DHCPv6. However, additional configuration information is provided by both forms of DHCP, such as which DNS or NTP server to use. These server addresses are formatted within the same protocol version of the respective version of DHCP. Hence DHCP for IPv4 options using IPv4 address-valued options, which DHCPv6 uses IPv6. The information obtained could lead to incorrect behavior on the client, depending on how the information from both servers is merged together. This remains an

ongoing area of concern, as documented in RFC 4477 (35), but the current standard is to use a DHCP server for IPv4 and a DHCPv6 server for IPv6, possibly implemented on a common physical server.

TUNNELING APPROACHES

A variety of tunneling technologies have been developed to support IPv4 over IPv6 and IPv6 over IPv4 tunneling. These technologies are generally categorized as *configured* or *automatic*. Configured tunnels are predefined, whereas automatic tunnels are created and torn down on the fly. We'll discuss these two tunnel types after reviewing some tunneling basics.

In general, tunneling of IPv6 packets over an IPv4 network entails prefixing an IPv6 packet with an IPv4 header as illustrated in Figure 8-2. This enables the tunneled packet to be routed over an IPv4 routing infrastructure; the IPv6 packet is simply considered payload within the IPv4 packet. The entry node of the tunnel, whether a router or host, performs the encapsulation. The source IPv4 address in the IPv4 header is populated with that node's IPv4 address and the destination address is that of the tunnel endpoint. The Protocol field of the IPv4 header is set to 41 (decimal), indicating an encapsulated IPv6 packet. The exit node or tunnel endpoint performs decapsulation to strip off the IPv4 header and routes the packet as appropriate to the ultimate destination via IPv6.

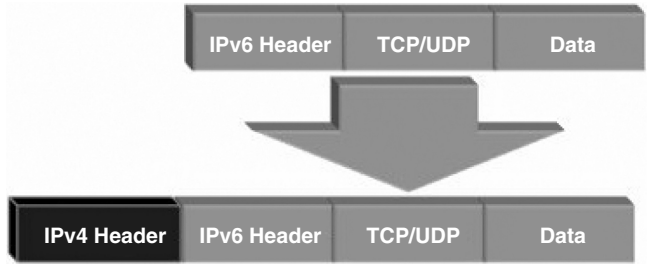

Figure 8-2: IPv6 over IPv4 tunneling

Tunneling Scenarios for IPv6 Packets over IPv4 Networks

Using this basic tunneling approach, a variety of scenarios based on tunnel endpoints have been defined. Probably the most common configuration is a router-to-router tunnel, depicted in Figure 8-3.

In this figure, the originating IPv6 host on the left has IPv6 address of W (for simplicity and brevity). A packet* destined for the host on the far end of the

*This packet is crudely identified in the figure as the solid-line rectangle beneath the originating host displaying the packet's IPv6 source address of W and destination address of Z. The tunnel header is shown as the dotted-line rectangle in this and subsequent tunneling figures.

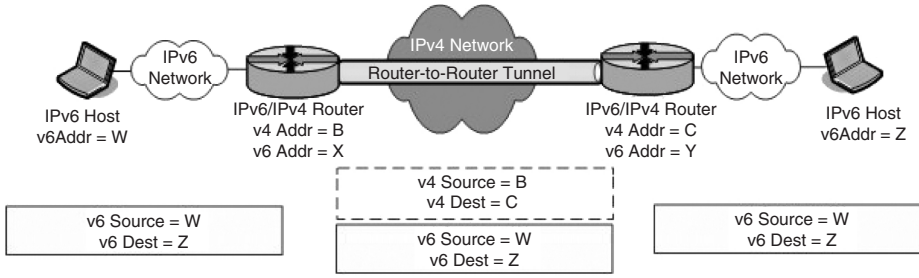

Figure 8-3: Router-to-router tunnel

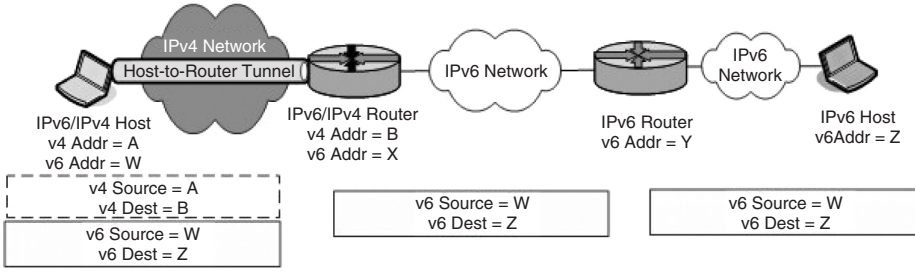

Figure 8-4: Host-to-router tunneling configuration

diagram with IPv6 address of Z is sent to a router serving the subnet. This router, with IPv4 address of B and IPv6 address of X, receives the IPv6 packet. Configured to tunnel packets destined for the network on which host Z resides, the router encapsulates the IPv6 packet with an IPv4 header. The router uses its IPv4 address (B) as the source IPv4 address and the tunnel endpoint router, with IPv4 address of C, as the destination address as depicted by the dashed rectangle beneath the IPv4 network in the center of Figure 8-3. The tunneled packets are routed like "regular" IPv4 packets to the destination tunnel endpoint router. This endpoint router decapsulates the packet, stripping off the IPv4 header and routes the original IPv6 packet to its intended destination, Z.

Another tunneling scenario features an IPv6/IPv4 host, capable of supporting both IPv4 and IPv6 protocols, tunneling a packet to a router, which in turn decapsulates the packet and routes it natively via IPv6. This flow and packet header addresses are shown in Figure 8-4. The tunneling mechanism is the same as in the router-to-router case, but the tunnel endpoints are different.

The Router-to-Host configuration is the converse of Host-to-Router and is shown in Figure 8-5. The originating IPv6 host on the left of the diagram sends the IPv6 packet to its local router, which routes it to a router closest to the destination. The serving router is configured to tunnel IPv6 packets over IPv4 to the host as shown in the figure.

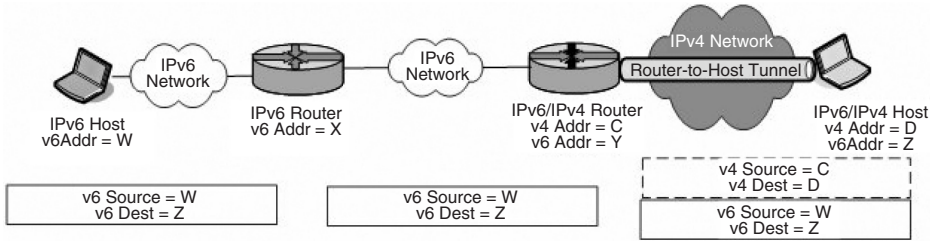

Figure 8-5: Router-to-host tunnel configuration

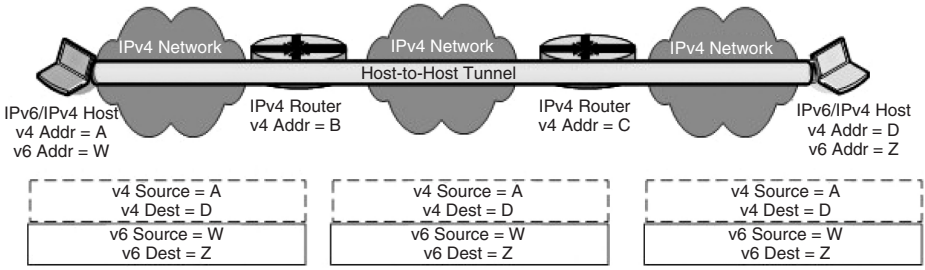

Figure 8-6: Router-to-host tunnel configuration

The final tunneling configuration is one that spans end-to-end, from host-to-host. If the routing infrastructure has not yet been upgraded to support IPv6, this tunneling configuration enables two IPv6/IPv4 hosts to communicate via a tunnel as shown in Figure 8-6. In this example, the communication is IPv4 from end-to-end.

Tunnel Types

As mentioned, tunnels are either configured or automatic. Configured tunnels are predefined by administrators in advance of communications. In the scenarios described above, configuration of the respective tunnel endpoints is required to configure each device regarding when to tunnel IPv6 packets, that is, based on destination, along with other tunnel configuration parameters that may be required by the tunnel implementation.

An automatic tunnel does not require tunnel preconfiguration, though enablement of tunneling configuration may be required. Tunnels are created based on information contained within the IPv6 packet, such as the source or destination IP address. The following automatic tunneling techniques have been defined:

- 6to4—6to4 is an IPv6 over IPv4 tunneling technique that relies on a particular IPv6 address format to identify 6to4 packets and to tunnel them accordingly. The address format consists of a 6to4 prefix, 2002::/16,

followed by a globally unique IPv4 address for the intended destination site. For example, a router with unique IPv4 address of 192.0.2.131 as a tunnel endpoint would advertise its reachability to the corresponding 2002:C000:283::/48 prefix. We converted our 192.0.2.131 address into hexadecimal, C0.00.02.83, and appended this to the 6to4 2002::/16 prefix.

- ISATAP—Intra-Site Automatic Tunneling Addressing Protocol provides automatic host-to-router, router-to-host, or host-to-host tunneling; ISATAP IPv6 addresses are formed using an IPv4 address to define its interface ID. The Interface ID is comprised of ::5EFE:a.b.c.d, where a.b.c.d is the dotted decimal IPv4 notation. So an ISATAP interface ID corresponding to 192.0.2.131 is denoted as ::5EFE:192.0.2.131. The IPv4 notation provides a clear indication that the ISATAP address contains an IPv4 address without having to translate the IPv4 address into hexadecimal. This ISATAP interface ID can be used as a normal interface ID in appending it to advertised network prefixes to autoconfigure IPv6 addresses.
- 6over4—6over4 is an automatic tunneling technique that leverages IPv4 multicast. IPv4 multicast is required and is considered a *virtual link layer* or *virtual Ethernet* by 6over4. Because of the virtual link layer perspective, IPv6 addresses are formed using a link local scope (FE80::/10 prefix). A host's IPv4 address comprises its 6over4 interface ID portion of its IPv6 address. For example, a 6over4 host with IPv4 address of 192.0.2.85 would formulate an IPv6 interface ID of ::C000:255, translating decimal to hex, and thus a 6over4 address of FE80::C000:255. 6over4 tunnels can be of the form host-to-host, host-to-router, and router-to-host, where respective hosts and routers must be configured to support 6over4. IPv6 packets are tunneled in IPv4 headers using corresponding IPv4 multicast addresses. All members of the multicast group receive the tunneled packets, thus the analogy of virtual Ethernet, and the intended recipient strips off the IPv4 header and processes the IPv6 packet. As long as at least one IPv6 router also running 6over4 is reachable via the IPv4 multicast mechanism, the router can serve as a tunnel endpoint and route the packet via IPv6.
- Tunnel Brokers—Tunnel brokers automate tunnel setup by assigning tunnel gateway resources on behalf of hosts requiring tunneling of IPv4 packets over IPv6. The tunnel broker manages (brokers) tunnel requests from dual stack clients and tunnel broker servers, which connect to the intended IPv6 network. These requests may result from an end user logging into the tunnel broker to connect to an IPv6 host. Upon successful authentication, the tunnel broker configures a tunnel server regarding the new tunnel, assigns an IPv6 address or prefix to the client, registers the client in DNS, and informs the client of this information.
- Teredo—Teredo provides automatic tunneling through NAT firewalls whereby IPv6 packets are tunneled over UDP over IPv4 for host-to-host automatic tunnels. Teredo incorporates the additional UDP header in order to facilitate NAT/firewall traversal. Many NAT/firewall devices will not

allow traversal of IPv4 packets with the protocol field set to 41, which is the setting for tunneling of IPv6 packets as described previously. The additional UDP header further "buries" the tunnel to enable its traversal through NAT/firewall devices, most of which support UDP port translation.

- Dual Stack Transition Mechanism—DSTM provides a means to tunnel IPv4 packets over IPv6 networks, ultimately to the destination IPv4 network and host. The host on the IPv6 network intending to communicate to the IPv4 host would require a dual stack, as well as a DSTM client. Upon resolving the hostname of the intended destination host to only an IPv4 address, the client would initiate the DSTM process, which is very similar to the tunnel broker approach.

TRANSLATION APPROACHES

Translation techniques perform IPv4-to-IPv6 translation (and vice-versa) at a particular layer of the protocol stack, typically network, transport, or application. Unlike tunneling, which does not alter the tunneled data packet but merely appends a header, translation mechanisms do modify or translate IP packets commutatively between IPv4 and IPv6. Translation approaches are generally recommended in an environment with IPv6-only nodes communicating with IPv4-only nodes. In dual stack environments, native or tunneling mechanisms are preferable per RFC 2766 (36).

Stateless IP/ICMP Translation (SIIT) Algorithm

The common algorithm for translating IPv4 and IPv6 packets is the Stateless IP/ICMP Translation (SIIT) algorithm. SIIT provides translation of IP packet headers between IPv4 and IPv6. SIIT resides on an IPv6 host and converts outgoing IPv6 packet headers into IPv4 headers, and incoming IPv4 headers into IPv6. To perform this task, the IPv6 host must be provided an IPv4 address, either configured on the host or obtained via a network service unspecified in RFC 2765 (37). When the IPv6 host desires to communicate with an IPv4 host, the SIIT algorithm would convert the IPv6 packet header into IPv4 format. The SIIT algorithm recognizes such a case when the IPv6 address is an IPv4-mapped address, formatted as illustrated in Figure 8-7. The mechanism to convert the resolved IPv4 address into an IPv4-mapped address is provided by bump-in-the-stack (BIS) or bump-in-the-API (BIA) techniques described later.

Figure 8-7: IPv4 mapped address format (11)

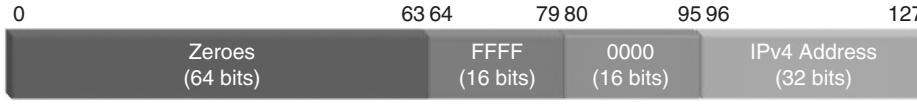

Figure 8-8: IPv4 Translated Address Format used within SIIT (37)

Based on the presence of the IPv4-mapped address format as the destination IP address, SIIT performs header translation to yield an IPv4 packet for transmission via the data link and physical layers. The source IP address for an IPv6 node uses a different format, that of the IPv4-translated format, shown in Figure 8-8, though RFC 2765 does not specify how this address is initially configured.

Now let's look at some techniques that employ the SIIT algorithm to translate IPv4 and IPv6 packets.

- Bump in the Stack (BIS)—BIS enables hosts using IPv4 applications to communicate over IPv6 networks. BIS snoops data flowing between the TCP/IPv4 software and link layer device drivers (e.g., network interface cards) and translates each IPv4 packet into an IPv6 packet. This technique translates the IPv4 header into an IPv6 header according to the SIIT algorithm. The destination IPv6 address is determined by snooping DNS queries for the A record type; upon detecting such a query, an additional query is sent for the AAAA record type for the same host domain name.
- Bump in the API (BIA)—The BIA strategy enables the use of IPv4 applications, while communicating over an IPv6 network. Unlike IP header modification provided by BIS, the BIA approach translates between IPv4 and IPv6 APIs. BIA is implemented between the application and TCP/UDP layer of the stack on the host. Like the BIS approach, BIA snoops outgoing API calls (e.g., gethostbyname()) and adds an AAAA query for each A record DNS query.
- Network Address Translation with Protocol Translation (NAT-PT)—DEPRECATED—As the name implies, the NAT-PT process entails translating IPv4 addresses into IPv6 addresses like an IPv4 NAT, but also performs protocol header translation. A NAT-PT device serves as a gateway between an IPv6 network and an IPv4 network and enables native IPv6 devices to communicate with hosts on the IPv4 Internet for example. The NAT-PT device maintains an IPv4 address pool, and associates a given IPv4 address with an IPv6 address while the communications ensues.
- Network Address Port Translation with Protocol Translation (NAPT-PT)—NAPT-PT enables IPv6 nodes to communicate with IPv4 nodes using a single IPv4 address. Thus, instead of maintaining a one-to-one association of an IPv6 address and a unique IPv4 address as in NAT-PT, NAP-PT maps each IPv6 address to a common IPv4 address with a unique TCP or UDP port value set in the corresponding IPv4 packet. The use of a single shared IPv4 address eliminates the possibility of IPv4 address pool depletion under the NAT-PT scenario.

- SOCKS IPv6/IPv4 Gateway—SOCKS, defined in RFC 1928 (38), provides transport relay for applications traversing firewalls, effectively providing application proxy services. RFC 3089 (39) applies the SOCKS protocol for translating IPv4 and IPv6 communications. And like the other translation technologies already discussed, this approach includes special DNS treatment, termed *DNS name resolving delegation*, which delegates name resolution from the resolver client to the SOCKS IPv6/IPv4 gateway. An IPv4 or IPv6 application can be "socksified" to communicate with the SOCKS gateway proxy for ultimate connection to a host supporting the opposite protocol.
- Transport Relay Translator (TRT)—Much like the SOCKS configuration, TRT features a stateful gateway device that interlinks two "independent" connections over different networks. The TCP/UDP connection from a host terminates on the TRT, and the TRT creates a separate connection to the destination host and relays between the two connections. TRT requires a DNS-Application Layer Gateway, DNS-ALG, which acts as a DNS proxy. TRT is specified to enable IPv6 hosts to communicate with IPv4 destinations. As such, the primary function of the DNS-ALG is to perform a AAAA resource record query as requested by IPv6 resolvers; if a AAAA record is returned, the reply is passed on to the resolver and the data connection may ensue as an IPv6 connection. If no AAAA records are returned, the DNS-ALG performs an A record query, and if an answer is received, the DNS-ALG formulates an IPv6 address using the IPv4 address contained in the returned A record.
- Application Layer Gateway (ALG)—ALGs perform protocol translation at the application layer and perform application proxy functions, similar to HTTP proxies. A client's application would typically need to be configured with the IP address of the proxy server, to which a connection would be made upon opening the application, for example, a web browser for the HTTP proxy case. An ALG may be useful for web or other application-specific access to the IPv4 Internet by hosts on an IPv6-only network.

For more details on these translation and funneling techniques, as well as a discussion of emerging IPv4–IPv6 service provider strategies including 6rd (IPv6 Rapid Deployment) and Dual Stack Lite, please consult *IP Address Management Principles and Practice*.

Application Migration

The de facto application programming interface (API) for TCP/IP applications is the *sockets* interface originally implemented on BSD UNIX (on which BIND was also originally implemented). The sockets interface defines program calls to enable applications to interface with TCP/IP layers to communicate over IP networks. Microsoft's Winsock API is also based on the sockets interface. Both sockets and Winsock interfaces have been modified to support IPv6's longer

address size and additional features. In fact, most major operating system have implemented support for sockets or Winsock including Microsoft (XP SP1, Vista, 7, Server 2003 & 2008), Solaris (8+), Linux (kernel 2.4+), Mac OS (X.10.2), AIX (4.3+), and HP-UX (11i with upgrade). The updated sockets interface supports both IPv4 and IPv6 and provides the ability for IPv6 applications to interoperate with IPv4 applications by use of IPv4-mapped IPv6 addresses. Check with your application vendors for IPv6 compatibility and requirements.

Planning the IPv6 Deployment Process

There's certainly no shortage of technology options when considering an IPv6 implementation approach. Having many options is good, but it can be intimidating. Selecting the right path will depend on your current environment in terms of end user devices and operating systems, router models and versions, key applications, budget and resources, as well as time frames. Given the proliferation of dual-stack support in leading operating systems and networking products, dual stack will likely be the most prevalent approach.

IPv6 implementation requires development of an overall plan for IPv6 network allocation and deployment. The scope of your plan will depend on whether you plan to deploy IPv6 only on externally (Internet) reachable resources or whether you'd like to also extend IPv6 deployment within your enterprise. While there are numerous detailed steps involved, the following four high-level steps are recommended for planning and performing such a transition.

1. Baseline your current environment.
   - Inventory and baseline the current IPv4 network environment in determining what IPv4 address spaces are in use, how they have been allocated, what individual IPv4 addresses have been assigned and are in use, and other IP related information per device.
   - As a corollary to the prior step, inventory and baseline the associated DHCP server configurations with respect to address pools and associated policies and options.
   - Identify and record associated DNS configurations and resource records associated with IPv4 devices.
   - Inventory network devices (infrastructure and end user) to assess current and expected future IPv4/IPv6 level of support.
   - Analyze applications for potential migration impacts and if so, plan for addressing these impacts.
2. Plan your IPv6 deployment.
   - Map out a strategy for IPv6 implementation including consideration of the following:
     - Application migration and required upgrades
     - Networking/router migrations or upgrades

    - Selection of co-existence technologies from tunneling, translation, and/or dual-stack approaches and planning corresponding DNS impacts
    - Selection of IPv6 device configuration strategy, for example, stateless autoconfiguration, stateful configuration, or hybrid
  - Consider time frame requirements if any, analyze dependencies across affected elements, and determine budgeting requirements to derive a migration plan.
3. Execute the Deployment Process.
  - Begin execution of the deployment plan based on the prior step and project manage the process with respect to schedules, budget, dependencies, and contingency plans.
  - Track the IP address inventory of baselined IPv4 space and associated IPv6 overlay space. Of course, growth and changes in the IPv4 space will most likely continue throughout the transition, so dovetail these updates into the plan as appropriate.
  - Update DNS and DHCP services to accommodate the transitioning environment for example, to support both IPv4 and IPv6 hosts, as well as IPv4 and IPv6 transport.
4. Manage the IPv4-IPv6 Network.
  - Manage IPv4 and IPv6 address space and corresponding DHCP and DNS services.
  - Based on your plan, some IPv4 space may be decommissioned. This can be done as portions of your network fully migrate to IPv6 or as a final decommissioning step across the network.
  - You may desire to leave IPv4 protocol operational throughout or on portions of your network depending on requirements to service external hosts which may be IPv4-only.

9

SECURITY CONSIDERATIONS

Security is at or near the top of every IP planner's list of networking concerns. IP address management–related security topics are no exception. This chapter will delve first into the area of network access control with discussion of some of common strategies for deploying prudent address assignment policies. Then we'll discuss DHCP information and communications security tactics before moving on to cover DNS vulnerabilities and mitigation strategies including DNSSEC.

DHCP AND NETWORK ACCESS SECURITY

There are a number of security threats to DHCP information and in its communications with those requesting information. Given the role of DHCP in disseminating IP addresses for access to the network, the DHCP service itself plays a key role in providing a basic level of network access control by virtue of its inherent function. Will you configure DHCP to provide an IP address to any device that requests one or will you configure a more discriminating policy?

Introduction to IP Address Management, By Timothy Rooney

NETWORK ACCESS CONTROL (NAC)

The term "NAC" has been hyped in recent years, but the underlying concept is fundamental: identify who is attempting to access your network prior to providing such access. Various techniques are available offering various levels of access control. We'll start by analyzing DHCP-based access control, which admittedly is among the weaker approaches to NAC. We'll then touch on more wide-reaching techniques.

Discriminatory Address Assignment with DHCP

Let's focus first on DHCP services and some approaches to implement discriminatory address assignment. There are several levels of policies or controls most DHCP solutions provide for discrimination of "who's asking" for an IP address via DHCP. The first is to simply filter requests by an available form of client identifier such as the MAC address of the client requesting an address. Recall that the MAC address is found in the client hardware address (chaddr) field of a DHCPv4 packet. DHCPv6 device identifiers consist of the Device Unique ID (DUID) and Identity Associations (IAs) that identify each client and interface, respectively.

If the DHCP server has a list of acceptable (and/or unacceptable) device identifiers, it can be configured to provide a certain IP address and associated parameters to those clients with an acceptable identifier, and either no IP address or a limited function IP address to those without an acceptable device identifier. By *limited function IP address*, we mean that the network routing infrastructure is preconfigured to route IP packets with such source IP addresses to only certain networks, such as to the Internet only, or even nowhere. An IP packet with source address A may be routable across the enterprise while one with source address B may be routable only to the Internet for example.

This type of IP address and configuration assignment is also possible by filtering on the client class of the device requesting an IP address as we discussed in Chapter 3. Certain clients, such as VoIP phones, provide additional information about themselves when requesting an IP address in the vendor class identifier field of the DHCP packet. The user class identifier field may also be used. The DHCP server can be configured to recognize user classes and/or vendor classes of devices on your network to provide additional information to the DHCP server when requesting IP address and configuration parameters. Addresses can be assigned from a certain pool and/or additional configuration parameters can be assigned to the client via standard or vendor-specific DHCP options.

A third level of discriminating IP address assignment is possible by authenticating the user of the machine requesting an IP address. This function can be used in conjunction with device identifier and client class discrimination described above. For example, if a client with an unknown or unacceptable device identifier attempts to obtain an IP address, one option is to completely deny an address;

another option is to require the user of the client to login via a secure access web portal page.

This enables easier capture of new device identifiers for legitimate users of your network. (Those users sometimes pop in new interface cards!) Solutions ranging from simple perl scripts such as NetReg, to sophisticated integrated software solutions are available to direct such users to a login/password requesting webpage. A simple lookup against a database of legitimate users then allows access or denial of the client to a production IP address. These systems typically work in accordance with the following packet flow shown in Figure 9-1 (the DHCPv4 process is shown but a comparable flow could be employed with DHCPv6).

Walking through this flow, the process begins with a device connecting to the network, attempting to obtain an IP address via DHCP. The DHCP server, employing device identifier or client class type filtering discussed above, determines whether the device is a known user device (in some cases, even known user devices may require periodic re-authentication as a security precaution). If the device is known or has otherwise already authenticated, the DHCP process may continue by providing a production IP address with an Offer, followed by a Request and Ack. However, if the device is not known or is required to authenticate, the DHCP server can still provide an IP address by completing the DORA process; but the IP address assigned in this case would be a *captive portal*, *walled garden*, or *quarantined* IP address.

These terms refer to the fact that the IP address assigned to the client will only be routed to the subnet or VLAN that has the authentication web server and associated servers running. This quarantined VLAN enables IP communications but only to this restricted set of devices. This cordons off the device from infiltrating the rest of the network until the corresponding user can be authenticated. The routing infrastructure must be configured to route packets with a source address from the quarantined address pool to the quarantined VLAN and/or the client must be configured with the classless static route option. Thus, address X as shown in Figure 9-1 is a member of the quarantined VLAN, on which only limited network resources are available. Figure 9-2 illustrates the network topology of this captive portal configuration.

Now when the user opens up a web browser, he/she can type in any web address. A *limited configuration* DNS server is required on the quarantined VLAN; limited in the sense that it will resolve any and every query to the IP address of the authentication web server. Thus, no matter what web address is entered in the web browser, the web address is resolved to the captive portal web server. The authentication web server presents the login page. You may have seen something similar to this if you travel and use a hotel's broadband or wireless service. Once the requested credentials are entered, which for an enterprise environment, would typically comprise a user ID and password, the web page can call a CGI script to pass the entered credentials to a back-end database. This authentication database could be an LDAP server, a Windows Domain Controller, a Radius server, or other form of authentication data store.

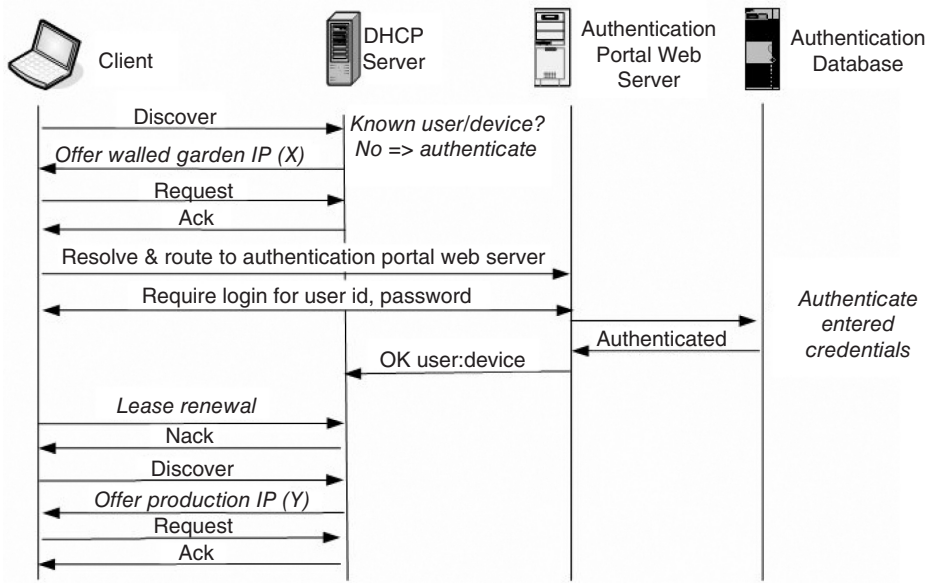

Figure 9-1: Basic DHCP captive portal flow

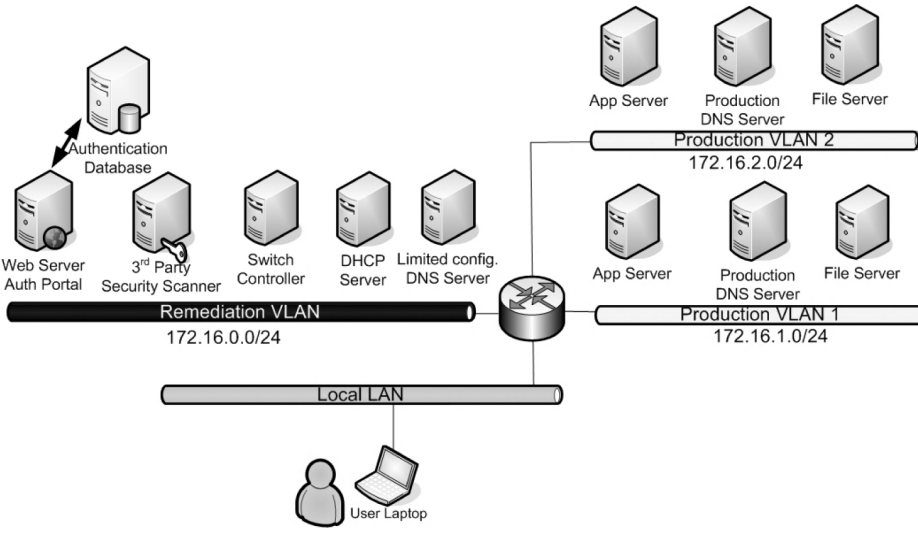

Figure 9-2: Captive portal network diagram

Based upon the results of the authentication, the requesting device would then be deemed authorized or not, and if authorized, optionally what class of authorization is granted. The class of authorization provides more granularity than a simple boolean "authorized or not," where different authorized users can be assigned a different production IP address, which in turn can provide access to different network resources. For example, basic level users may be granted access to a basic set of resources, while advanced level users may be granted access to additional resources, for example, IT resources. Once again this requires the routing topology be configured with multiple source-routed or VLAN segments, with these networks and corresponding routing plan mapped to DHCP server configurations in terms of associating address pools with service levels.

The manner in which the production IP address is assigned follows from expiration or denial of renewal of the quarantined IP address. The quarantined IP address lease time is generally configured as a short lease time (~1–5 minutes). This enables the device to attempt to renew quickly. Should the device still be in the process of authentication, its renewal attempt would be ACK'd, extending the lease. Once authentication is completed successfully, the authentication system updates the DHCP server to add the client MAC address to the "known" or "allow" pool. The renewal attempt for the quarantined address would then be NAK'd, enabling a fresh DORA process to provide a "production" IP address (address Y in Figure 9-1). Should the device fail authentication, the renewal can be NAK'd and subsequent address attempts denied; alternatively, the quarantined address renewal attempt can be granted in order to provide access only to resources on the quarantined network if desired.

Beyond these device and user identification measures based on device identifiers, client classes, and user authentication, this general flow can also provide additional validation on the machine requesting the IP address. The DHCP process can be used to invoke an external security scanning system like Nessus or another third-party application to scan the requesting client for viruses or to validate use of acceptable virus protection software. This device scanning step can be used alone or in conjunction with the device identification measures to provide a robust access security solution via DHCP.

One example network configuration for DHCP based secure access is depicted in Figure 9-2. The DHCP server shown in the diagram would be configured with a number of client class sets. We refer to the client class as the matching criteria in the DHCP packet, and link this to client class set mapping to the associated network accessibility. For example, we would need a client class set for at least each of the following in our example:

- Captive portal network (Remediation VLAN)
- Production network 1 network
- Production network 2 network

Think of these client class sets as bins into which individual clients are placed based on the linking of their authentication state to the device's client class.

These client classes would generally map to pool definitions on the DHCP server as shown in the following example configuration for an ISC DHCP server (40). Note that additional options can be defined for each of the pools to provide additional configuration granularity to clients falling into each set or pool.

```
subnet 172.16.0.0 netmask 255.255.252.0 {
# subnet level options here…
  pool{                                          #captive portal pool
    range 172.16.0.10 172.16.0.254;
    option domain-name-servers 172.16.0.5;       #limited config DNS server
    default-lease-time 150;                      #short lease time
    allow unknown clients;                       #clients not predefined.
  }
  pool {                                         #Prod Net 1
    range 172.16.1.10 172.16.1.254;
    option domain-name-servers 172.16.1.5;       #production DNS server
    default-lease-time 14400;                    #normal lease time
    deny unknown clients;                        #clients must be predefined.
    allow members of "net1";                     #client class net1 allowed
  }
  pool {                                         #Prod Net 2
    range 172.16.2.10 172.16.2.254;
    option domain-name-servers 172.16.2.5;       #production DNS server
    default-lease-time 14400;                    #normal lease time
    deny unknown clients;                        #clients must be predefined.
    allow members of "net2";                     #client class net2 allowed
   }
}
```

Based on the results of the authentication process, the authentication server must be able to update the DHCP configuration to place the client into the appropriate bin or class. Thus if the device is successfully authenticated for access to production network 2, the authentication portal needs to add the specific device's client class value (e.g., MAC address) to the client class group for production network 2 ("net2" in the example above). This update may be performed, for example, using the ISC DHCP server OMAPI interface (requires version 3.1 or above). This client class declaration can define class-specific options on the DHCP server to provide to the client, for example, default gateway, DNS server, along with any other option.

The captive portal VLAN may consist only of "unknown clients," a designation configurable with the ISC DHCP server. The captive portal network (the remediation VLAN in the figure) is deployed including the limited configuration

DNS server, web server as the authentication portal, with access to an authentication database, and optionally a security scanning server and any other required pre-access services.

More than one DHCP server may be deployed for high availability and/or for scaling for larger networks. This approach does complicate things, as the DHCP server configurations need to be consistent to route unknown clients or clients requiring authentication to the captive portal net.

ALTERNATIVE ACCESS CONTROL APPROACHES

You may be thinking that the DHCP-based approach is fine for clients utilizing DHCP; but what about those clever users who figure out the subnet address, then manually encode a static IP address on their machine to access the network? These clever users may after all be those of most concern from a secure access perspective. And for devices using IPv6 stateless autoconfiguration, no DHCP interaction is required for address assignment.

There are three basic alternative approaches for enabling detection and associated remediation action of devices without relying on the DHCP-based approach:

- DHCP Lease Query
- Layer 2 switch alerting
- 802.1X

DHCP Lease Query

If most or all addresses on a subnet are configured using DHCP by policy, that is, each IP address *should* have a corresponding DHCP lease, then the Lease Query approach may be used. The DHCP Lease Query is a DHCP protocol message that enables an edge router to query the DHCP server regarding the lease status of a particular device's hardware address. This provides some assurance that a device attempting to communicate via the router has not spoofed an address that should have been assigned by the DHCP server.

When the router receives IP traffic within a layer 2 frame from a particular MAC address, it can issue a DHCPLeaseQuery message to its configured DHCP servers (i.e., in its role as relay agent). If a DHCP server had previously provided a lease for the client, it will respond to the router, and the router will give the green light and route the device's packets. If not, the device does not have a lease and the router can drop the device's packets. The router can cache this information as well so the Lease Query rate is not exceedingly high. Of course, this form of access control applies only when all clients on a subnet use DHCP, such as in broadband access networks, not when other statically addressed devices communicate on the subnet.

Layer 2 Switch Alerting

Another approach takes advantage of SNMP-enabled switches to issue an SNMP trap upon a link-up state on one of its ports and to accept port level VLAN configurations. This alerting capability along with SNMP writeable configuration information can enable gatekeeper-like functionality by dynamically identifying devices attempting to access the network, and configuring the switch to provision the port to a particular VLAN. A third-party system or product would be needed to process traps, make decisions on appropriate VLAN assignments, and configure the switch accordingly.

Let's look at how this would work. If we consider the process of a device connecting to a network from the beginning, the device "boots up" on the network from layer 1 on up. Thus the physical layer/electrical connectivity is first attained; then the data link layer is initialized whereby layer 2 frame synchronization occurs. Then layer 3 follows, with the issuing of a DHCP packet, for example, or directly issuing IP packets if a static address is encoded at layer 3. As the data link layer initializes (prior to layer 3), the switch to which the device is connected will deem the "link up" and issue a trap. Because the trap is sent prior to layer 3 initialization, this scheme can identify both statically addressed and DHCP-addressed devices.

Traps would be directed to a system that can identify the link up state, ascertain the link layer (MAC) address of the newly connected device, then determine whether the device requires authentication or validation. This determination can be made via a MAC address database within the system that identifies known or acceptable MAC addresses and differentiates these from unknown or known unacceptable MAC addresses. The system would associate these two or perhaps more MAC address categorizations with corresponding VLAN assignments, which would then be programmed on the corresponding switch for the given port. The connected device would then be connected to the assigned VLAN. You can probably see the analogy to the DHCP scenario we discusssed using client classes. In this case, the third-party system uses its database and configures the layer 2 switch using SNMP or other means instead of assigning an IP address using DHCP.

For quarantined or captive portal access, the VLAN assignment would lead only to the authentication network. For those passing authentication and/or device validation, the system could reassign the MAC address to the acceptable list, then configure the switch accordingly to change the port's VLAN association. Depending on the authentication method, client software may or may not be required. For web-based login/password, it may not be necessary to configure each of your client computers with authentication clients. However, if Radius or other challenge/response authentication strategies are employed, client software will be necessary.

802.1X

IEEE 802.1X is a protocol specification enabling edge device capture of new access attempts, with the use of Radius authentication and dynamic switch port configuration. You may have used Radius in the days of pre-broadband Internet

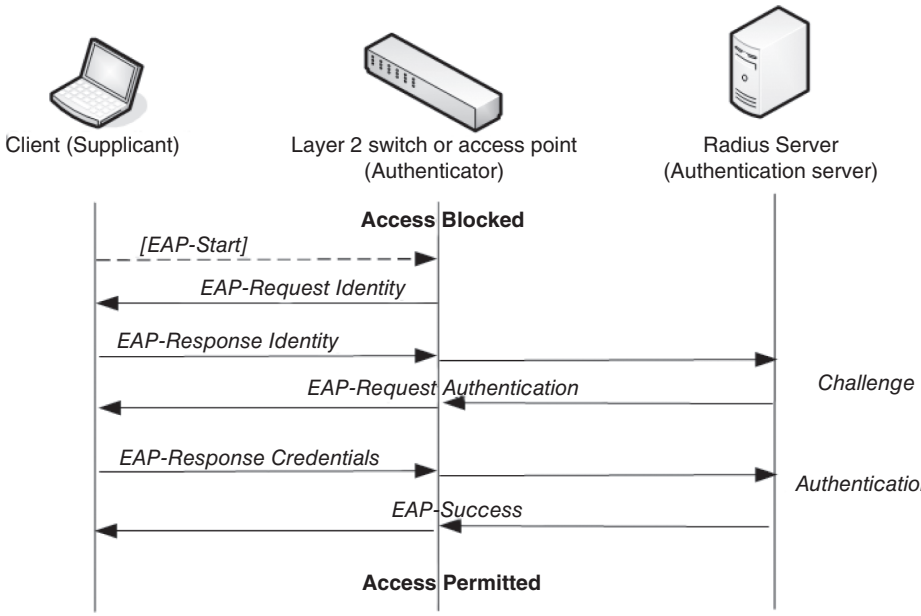

Figure 9-3: 802.1X authentication

dial-up, but this form uses the Point-to-Point protocol at layer 3. 802.1X was developed by the IEEE 802.1 working group focused on layer 2 protocols, so it's no surprise that 802.1X is a layer 2 protocol. Like the switch-based authentication strategy discussed in the previous section, this approach operates at layer 2, prior to the device getting a layer 3 (IP) address via DHCP. And 802.1X is based on standards, which theoretically enables use of different vendors' products as components within the overall solution.

As depicted in Figure 9-3, 802.1X requires a client or agent called a *supplicant*, which interacts with an *authentication server* by way of an *authenticator* (e.g., switch). Upon initial connection to a network, the supplicant utilizes the Extensible Authentication Protocol (EAP) over 802.1x to initiate a connection request to the network access device. The switch can be configured to block all traffic by default except EAP packets from unauthenticated ports.

The access switch to which the device is connected at the data link layer transmits the EAP traffic to the authentication (Radius) server. The Radius server, in turn, challenges the client for an ID and password. Upon successful authentication, the Radius server communicates to the edge device to enable access to the associated device's port.

Cisco Network Admission Control (NAC)

Cisco's NAC offering (41) is based on 802.1X. It requires a Cisco Trust Agent (CTA) together with a Cisco Security Agent installed on each end user device.

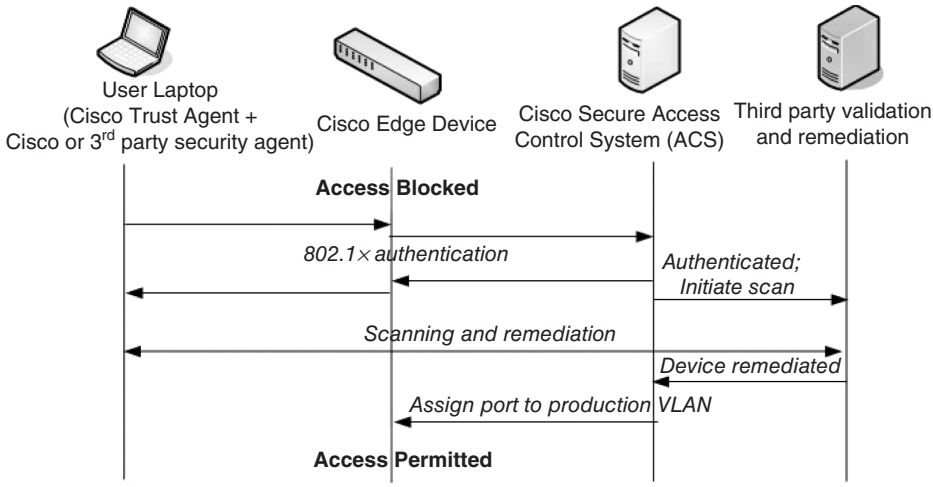

Figure 9-4: Cisco NAC basic flow

The Trust Agent contains a Radius client. Upon initial connection to a network, the CTA utilizes the Extensible Authentication Protocol (EAP) over 802.1x or UDP to initiate a connection request to the network as illustrated in Figure 9-4.

The network access device, typically a switch to which the device is connected at the data link layer, transmits the EAP traffic to the Cisco NAC appliance or Access Control Server (ACS), either of which provides Radius services. This Radius component, in turn, challenges the client for an ID and password. A third-party validation solution may be invoked to scan the device attempting to gain access. Upon successful authentication and validation, the ACS or NAC appliance communicates to the edge device to enable access to the associated device's port.

Microsoft Network Access Protection (NAP)

Microsoft introduced Network Access Protection (NAP) (42) to enable administrators to assure computers accessing a network have appropriate software installed at or above a specified version and is supported in Microsoft Vista™ and 7 client and Windows Server 2008 implementations. Microsoft also supports an API to enable other vendors to support NAP technologies. NAP primarily emphasizes device compliance and health, and as a byproduct, access control. That is, NAP is intended to enable network administrators to validate device compliance with current software releases upon network access, and not inherently to prevent access by malicious attackers. Nonetheless, NAP does entail denial of access pending health status verification with its three key functions:

- Health Policy Validation—Upon a network access attempt, a device's "health state" is obtained and compared with the administrator's defined "health policies." If the device complies with the specified health

policies, the device is permitted unrestricted access; if the device is not compliant, the device can be provided restricted access or full access if in monitoring-only mode.

- Health Policy Compliance—Non-compliant devices can optionally be automatically upgraded upon access attempt. If in monitoring-only mode, the device will enjoy full network access. In restricted access mode, the device will have only restricted network access until compliance is achieved.
- Limited Access—Administrators can constrain the scope of network accessibility to noncompliant devices.

Microsoft Vista (and above) clients contain a NAP client that communicates with a NAP Policy Server (NPS), part of Microsoft Windows Server 2008, during an attempt to access a network. The NPS enforces the compliance policies and is consulted during access attempts over a variety of technologies, including IPSec, 802.1X, VPN, Radius, and DHCP. IPSec is the strongest form of policy enforcement and consists of a Health Registration Authority (HRA), which issues X.509 certificates to compliant NAP Enforcement Clients (ECs) based on compliance verification by the NPS.

The 802.1X access flow follows that described above for 802.1X, with the addition of the NAP Policy Server validating the device's compliance. VPN, Radius, and DHCP access components include a NAP Enforcement Server (ES) and a NAP EC which communicate regarding policy compliance during access attempts to the network via the respective technologies.

The client device, or NAP client, contains an NAP Agent, as well as Microsoft provided and API accessible System Health Agents (SHAs) and NAP ECs in support of additional applications. Likewise, the NPS features an API for corresponding System Health Validators (SHVs). When attempting to access the network, the NAP client will provide a Statement of Health (SoH) to the NPS via the corresponding access server; for example, the HRA, VPN server, DHCP server, etc. The access server passes this on to the NPS, which validates the policy compliance and either permits or restricts access based on the access technology and NAP policies. Full or restricted access is conveyed to the NAP EC on the client for enforcement, though other network configuration such as router access lists or static routes may also be required. The basic relationship among these NAP components is shown in Figure 9-5.

SECURING DHCP SERVERS

DHCP Threats

Within enterprise environments, most threats to DHCP are posed by internal (i.e., intra-organizational) clients. DHCP servers should not be reachable by external clients by simply not deploying DHCP servers on external subnets nor relaying DHCP packets from external sources. For service providers that initialize

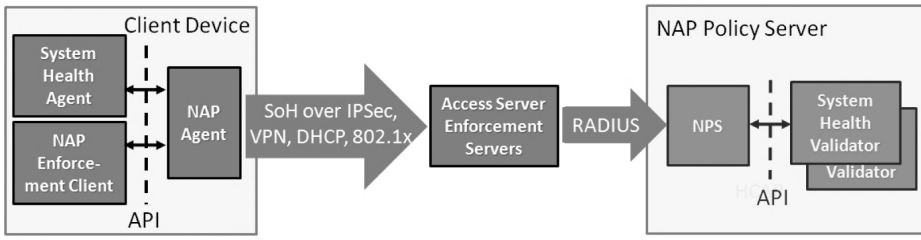

Figure 9-5: Microsoft NAP components (42)

subscriber devices using DHCP, whether cell phones, cable or fiber routers, machine-to-machine (M2M), etc., threats to DHCP service can by definition originate externally to the network. In short, all organizations using DHCP are vulnerable. The degree of vulnerability and the impacts of compromise should drive the response in the form of securing DHCP to minimize such impacts. We'll look at the major forms of attack next.

Like all network services, DHCP is vulnerable to denial of service (DOS) attacks. When an attacker floods a given server with requests too numerous for the server to handle, the server may spend all of its cycles attempting to deal with the flood and not on legitimate client requests; thus these legitimate clients go unserved, and service is denied them.

Another type of attack involves a rogue client attempting to obtain a valid IP address and configuration to access the network. This could be malicious, for example, theft of broadband service, or merely accidental, for example, a visitor plugging into the wall jack in the conference room.

A third form of attack features a rogue DHCP server that responds to lease requests from clients with incorrect IP address and/or option parameter information. This "man in the middle" type of attack may attempt to set improper configuration parameters on the client, such as the default gateway or DNS server address(es) to use. Note that with IPv4, a rogue DHCP server attack is generally only applicable when the server is on the same subnet as the client; relay agents presumably would be configured to relay DHCP packets to authorized DHCP servers. A remote rogue DHCPv6 server may be reachable via the DHCP multicast address.

The client may receive DHCPOFFERs from both the legitimate DHCP server(s) and the rogue server. Many clients will select the first offer that includes its requested parameters. If the rogue server is on the same subnet as the client, and legitimate servers are not, then it's likely the rogue server may be able to specify the IP configuration of the client.

DHCP Threat Mitigation

Protection against DOS attacks should be implemented in a broader context beyond just DHCP. Other potential targets within an organization including DNS servers or web servers imply that a gateway-based or packet-filtering

approach be considered to protect all servers with a common solution. Such a solution typically involves packet filtering and thresholds limiting the number of outstanding packets in process, though care must be taken with DHCP since most clients' transactions are funneled through DHCP relay agents, concentrating packets from a given set of source addresses.

Mitigation steps for the threat of unknown clients accessing the IP network by illicitly obtaining an IP address from DHCP requires identification of clients based on various access control techniques we discussed at the beginning of this chapter.

Rogue DHCP servers may be difficult to detect, especially for clients on the same subnet as the rogue server. But both ISC and Microsoft implementations provide means to mitigate rogue servers. For ISC, use the `authoritative` directive, which configures the server to issue a DHCPNAK if a client requests a lease for an address for which the server is authoritative yet the server has no record of it. Microsoft requires DHCP servers to be authorized within Active Directory; thus when a Windows DHCP server boots it verifies its authorization in Active Directory before processing DHCP packets.

DHCP Authentication

The IETF has described DHCP authentication, defined in RFC 3118 (43), as a mechanism providing validation of the sender and receiver of DHCP packets via the use of shared tokens or keys. A token is simply a fixed value that is inserted into the DHCP Authentication option. The receiver of the packet examines the token and if the token matches its configured token, the packet is accepted; otherwise it is dropped. This method provides weak endpoint authentication and no message verification. The use of shared keys can provide stronger endpoint authentication with message verification. However, shared keys must be configured on each client, with each client's key configured on each DHCP server to which the client obtains leases. The DHCP Authentication specification does not define the mechanism for key distribution. Mobile clients, for example, would need to be configured with tokens for each DHCP server with which they may interact and vice versa.

Here's how DHCP authentication works. The client would create an HMAC-MD5 hash of its DHCPDISCOVER packet and sign it using the shared key. The resulting digest would be placed in the DHCP Authentication option and transmitted within the DHCPDISCOVER packet to the server. The DHCP server would then compute a hash of the received message utilizing the shared key associated with the client (identified by the secret ID field of the DHCP Authentication option). If the calculated hash matches that transmitted in the original DHCP Authentication option, the client and the contents of the packet are considered authenticated. The DHCP server utilizes the same shared key to compute the hash value of its DHCP Authentication option when it prepares its DHCPOFFER and future packets to the client.

There have been very few implementations of DHCP Authentication. The challenges of key management and processing delays due to hash computation

have been deemed too heavy a price to pay for the perceived benefits. Security of the DHCP service then typically falls on DHCP server administrators to monitor servers and react to threats as they occur.

DNS VULNERABILITIES

It's instructive in discussing DNS security vulnerabilities to consider the data sources and data flow within the DNS, depicted in Figure 9-6. Starting in the upper right hand corner of the figure, DNS servers are initially configured with configuration and zone file information. This configuration step may be performed using a text editor or an IPAM system. For Microsoft implementations, the "IPAM system" may be the Microsoft Management Console (MMC).

For BIND implementations, this configuration consists of a named.conf file and associated zone files on the master server. While a server may be master for some zones and slave for others, we'll use the master server terminology in assessing the vulnerability of a particular zone's information. Configuration of slave servers requires creation of the configuration file only, which defines the server's configuration parameters and its authority for particular zones. Slaves transfer zone information from corresponding masters.

Zone information may be updated by external sources as well, particularly DHCP servers. Dynamic updates can be accepted for clients obtaining dynamic IP addresses requiring DNS updating of address-to-name mappings. These updates will typically originate from the DHCP server assigning the address and will be directed to the server acting as master for the given zone. The master will

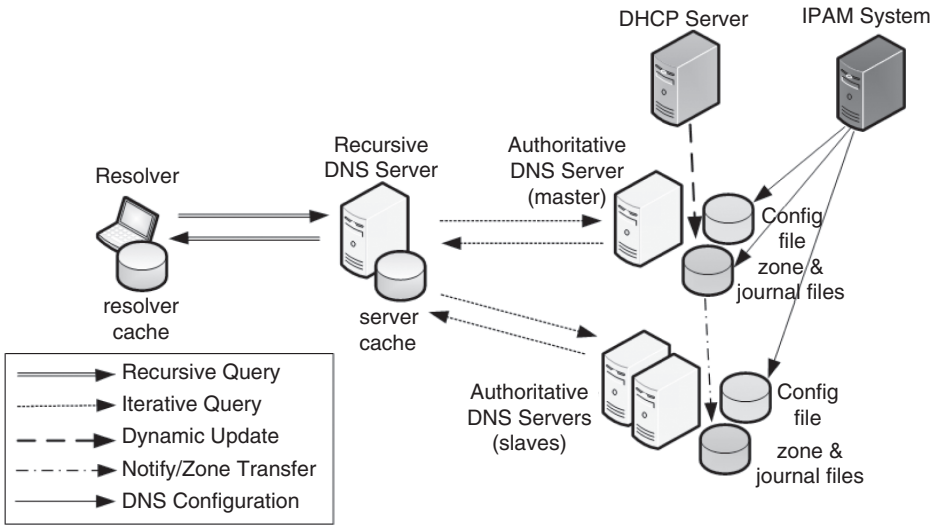

Figure 9-6: DNS data stores and update sources

add the update to its journal file and may then notify its slaves of the update, who may request an incremental zone transfer to capture the updated zone information. Thus, authoritative zone information can be configured on a name server by zone file editing directly or via an IPAM system, and via zone transfers and dynamic updates, yielding several potential data sources and data update communications paths.

Beyond the configuration information and zone files, the third information respository within a DNS server is its cache. Cache information is accumulated through the query resolution process. As query answers are sought and received, corresponding answers are cached by the server. The cached information may comprise not only the answer within the DNS message, but also additional information purportedly supplemental to the answer, as provided by the answering server. This information may include the authoritative servers for the relevant zone and other information related to the query (e.g., the A/AAAA "glue" record for an NS query).

The query resolution flow, beginning on the left of the figure, begins with the client resolver initiating a recursive query to its recursive server. Recall that the target server to query is defined in the client's resolver configuration, managed manually or via DHCP. The recursive server will issue iterative queries as necessary through the domain tree to resolve the query, generally ending with a name server that is authoritative for the zone corresponding to the query. The master or any of the slaves are authoritative with the zone information. The authoritative server responds with the answer and potentially related information. The recursive server will generally cache this information, and this cache will be relied upon for similar future queries to improve resolution performance. So it's important to assure data integrity of information returned to both the resolver and recursive server (i.e., both resolvers, the stub resolver in the client and the resolver within the recursive server).

Now let's look at the vulnerability of this information and communications model. RFC 3833 (44) thoroughly discusses various vulnerabilities to the DNS protocol and information integrity. We'll summarize those and some other vulnerabilities here, and then address mitigation strategies.

Resolution Attacks

Packet Interception or Spoofing. Like other client/server applications, DNS is susceptible to "man-in-the-middle" attacks where an attacker responds to a DNS query with false or misleading information. The attacker spoofs the DNS server response, leading the client to resolve and cache this information. This can result in hijacking resolvers and hence applications to incorrect destinations, for example, web sites.

ID Guessing or Query Prediction. Another form of malicious resolution is ID guessing. The ID field of the DNS packet header is 16 bits in length, as is the UDP packet header ID. If an attacker can provide a response with the correct

ID field and UDP port number, the resolver will accept the response, whatever it contains. This enables the attacker to provide falsified results, assuming the query type and queried name are known or guessed by the attacker. This attack can potentially redirect the host to an illicit site. Guessing a 2^{32} number is relatively easy even with brute force methods, especially when IDs used to be incremented linearly.

Name Chaining or Cache Poisoning. This packet interception style attack features an attacker providing supplemental resolution information, thereby poisoning the cache with malicious query information. This may, for example, attempt to falsify information for a popular website such as cnn.com, google.com, or the like, so when such a query is requested, the resolver will rely on this falsified cached information. When the resolver is asked to resolve such a query, it will access its cache and utilize the malicious information to essentially redirect the client to the attacker's intended destination. An alternative approach to forcing the resolver to access the poisoned cache data is to lure the user to click a link in a phishing email, which when followed, will resolve to the intended poisoned hostname. The so-called Kaminsky DNS vulnerability is a cache poisoning type of attack.

Resolver Configuration Attack. The resolver on the client must be configured with at least one DNS server IP address to which DNS queries can be issued. This configuration may be performed manually by hard-coding the DNS server IP address in the TCP/IP stack, or automatically via DHCP or PPP. This type of attack may alternatively originate from an attacker launching it via a web plug-in, for example. This type of attack seeks to redirect the resolver to an attacker's DNS server to resolve to malicious data.

Configuration and Server Attacks

Dynamic Updates. An attacker may attempt to inject or modify data in a DNS zone by attempting a dynamic update to the server. This type of attack attempts to redirect resolutions from clients for the intended destination to an attacker-specified destination. Imagine an attacker updating your www record on your external DNS servers!

Zone Transfers. Impersonating a slave and attempting to perform a zone transfer from a master is a form of attack that attempts to map or footprint the zone. That is, by identifying host to IP address mappings, as well as other resource records, the attacker attempts to identify targets for direct attacks. Attacking a particular host using its hostname as a clue ("payroll," for example) provides an easy target to attempt access or denial of service.

Server Configuration. An attacker may attempt to gain access to the physical server running the DNS service. Requiring a login and password to access the

server locally or remotely is highly recommended to defend against direct server access. Use of secure shell (SSH) is also recommended for remote access. When using an IPAM system, verify that IPAM-to-DNS server communications are secure. Beyond being able to manipulate named and zone information, an attack of this type certainly enables the use of the server as a stepping stone to other targets, especially if this server happens to be trusted internally.

Control Channel Attack. Access to DNS server control channels (rndc, remote name daemon control) provides powerful remote control capabilities, such as stopping/halting named, reloading a zone, and more. Accessing the control channel and stopping the service thereby denies the service to querying servers and resolvers.

Buffer Overflows and Operating System Attacks. An attacker may attempt to gain access to the server by overflowing the code execution stack or buffer. Without going into details, such an attack calls a subroutine that returns to the main program at a point defined by the attacker. This is one example of several similar types of OS level attacks, exploiting OS vulnerabilities on which the DNS service is running.

Configuration Errors. While typically not malicious (though many attacks are initiated from internal sources), misconfiguring the DNS service and/or zone information may lead to improper resolution or server behavior.

Denial of Service Attacks

Denial of Service. DNS, like other network services, is vulnerable to denial of service (DOS) attacks, which features an attacker sending thousands of packets to a server in hopes of overloading the server, causing it to crash or become otherwise unavailable to other queriers. The service is rendered unavailable and thus denied to others.

Distributed Denial of Service. A variant of this type of attack is the use of multiple distributed attack points and is referred to as a distributed denial of service (DDOS) attack. The intent is the same, though the scale is larger, potentially impacting several servers.

Reflector Attack. this form of attack attempts to use DNS servers to launch massive amounts of data at a particular target, thereby denying service for the target machine. The attacker issues numerous queries to one or more DNS servers using the target machine's IP address as the source IP address in each DNS query. Querying for records with large quantities of data such as NAPTR, EDNS0, and DNSSEC queries magnifies this attack. Each responding server responds with the data to the "requestor" at the spoofed IP address to inundate this target with a large data flow.

MITIGATION APPROACHES

Strategies for addressing these vulnerabilities are summarized in Table 9-1. In general, you should keep tabs on vulnerability reports from vendors and perform fixes and upgrades when they become available. Deployment strategies for hidden masters and generally deploying role-based DNS servers are also effective in mitigating attacks.

Several vulnerabilities, though not all, can be addressed with the deployment of DNSSEC, so let's take a closer look at this technology.

TABLE 9-1: DNS Vulnerabilities and Mitigation Approaches

Vulnerability	Mitigation
Packet interception/spoofing	DNSSEC provides effective mitigation of this vulnerability by providing: • Origin authentication—verification of the data source • Data integrity verification—data received is the same as the data sent • Authenticated denial of existence—a sought resource record does not exist
ID guessing/query prediction	DNSSEC effectively mitigates this vulnerability; in addition, BIND 9 randomizes DNS header message IDs to reduce the chance of guessing its value in a fake response. Since mid-July 2008, BIND also randomizes UDP port numbers on outbound queries to reduce the risk of this vulnerability.
Name chaining/cache poisoning	DNSSEC provides origin authentication and data integrity verification to resist these vulnerabilities; additional BIND directives for cache and additional section cache enabling and cleaning intervals can also help; transaction ID and UDP port randomization also help reduce the risk of this vulnerability.
Resolver configuration attack	Configure DNS servers via DHCP; monitor or periodically audit each client for misconfigurations or anomalies
Illicit dynamic update	Use ACLs on allow-update, allow-notify, notify-source. ACLs may also be defined as requiring transaction signatures for added origin authentication.
Illicit zone transfer	Use ACLs with TSIG on allow-transfer; and use transfer-source IP address and port to use a nonstandard port for zone transfers

TABLE 9-1: *Continued*

Vulnerability	Mitigation
Server attack/hijack	• Use hidden masters to inhibit detection of the zone master • Disallow recursive queries on masters and on ALL external DNS servers • Keep server operating system up to date • Limit port or console access • Implement chroot
Control channel attack	Use ACLs within the controls statement to restrict who can perform rndc command; require rndc key
Buffer overflows and OS level attacks	Keep OS updated, limit cache, acache (additional section cache) sizes, and define cached cleaning intervals
Named service misconfiguration	Use checkzone and checkconf utilities, as well as an IPAM system with error checking; keep fresh backups for reload if needed
Denial of service	• Limit communications using such parameters as recursive-clients, max-clients-per-query, transfers-in, transfers-per-ns, cache and acache sizes • Consider anycast deployment
Reflector attacks	• Use allow-query/allow-recursion ACLs • Use views if appropriate • Require TSIG on queries if possible

DNS SECURITY EXTENSIONS (DNSSEC)

When I sign a letter or check, I inherently demonstrate my approval and authorization by virtue of my signature. When I sign more important documents, like a mortgage note, I need to have my signature validated, typically through a notary public. The notary verifies my identity and also validates my signature, generally by comparing it with a driver's license or passport signature. By stamping my mortgage note, the notary confirms that it is I who signed the document and therefore my signature is trusted. DNSSEC works in a loosely analogous fashion. A resolver, or recursive server resolving on behalf of a stub resolver receives resolution data along with a signature on the data. As long as I trust the signer, I can validate the data using the signature. The element of trust requires some initial configuration of trust information from the signer in the form of trusted keys, which are used to verify the trustworthiness of the signer of the data received. If I don't trust the signer directly, I need to seek signature validation by seeking an entity that I trust that will "vouch for" the signer. Not that

my mortgage company doesn't trust me, but they required validation on my signature!

DNS security extensions, DNSSEC, originally defined in RFC 2535 (45), was modified and recast as DNSSEC*bis*, defined in RFCs 4033-4035 (46)–(48). This recasting was due to scalability issues with the original specification. DNSSEC*bis*, hereafter referred to simply as DNSSEC, provides a means to authenticate the origin of resolution data within DNS and to verify the integrity of that data. Thus, DNSSEC enables detection of packet interception, ID guessing, and cache poisoning attacks. DNSSEC utilizes digital signatures to perform data origin authentication and end-to-end data integrity verification.

Digital Signatures

Digital signatures enable the originator of a given set of data to sign the data such that those receiving the data and the signature, along with a corresponding public key for deciphering the signature, can perform data origin and integrity verification. DNSSEC uses an asymmetric key pair (private key/public key) model. In such a model, data encrypted with a private key can be validated by decrypting the data with the corresponding public key and vice versa. The private key and public key form a key pair. Conceptually, the private/public key pairs provide a means for holders of the public key to verify that data was signed using the corresponding private key. This provides authentication that the data verified was indeed signed by the holder of the private key. Digital signatures also enable verification that the data received matches the data published and was not tampered with in transit.

Refer to Figure 9-7. The data originator, shown on the left of the figure, generates a private key/public key pair and utilizes the private key to sign the data. The first step in signing the data is to produce a hash of the data, sometimes also referred to as a digest. Hashes are one-way functions to scramble data into a fixed length string for simpler manipulation, and represent a "fingerprint" of the data. A one-way function means that the original data is not uniquely derivable from the hash. That is, one can apply an algorithm to create the hash, but there is no inverse algorithm to perform on the hash to arrive at the original data.

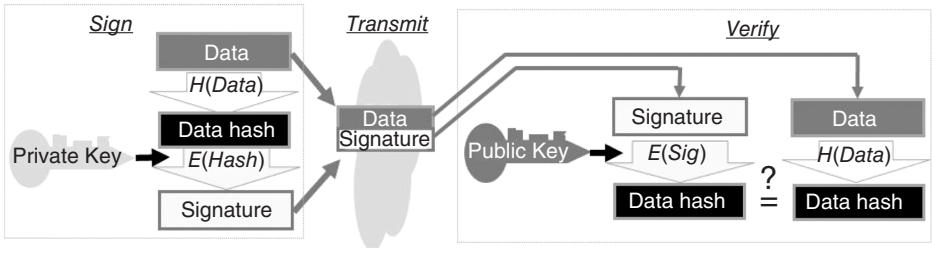

Figure 9-7: Digital signature creation and verification process

This means that it is very unlikely that another data input could produce the same hash value. Thus hashes are often used as checksums but don't provide any origin authentication (anyone knowing the hash algorithm can simply hash arbitrary data). Common hash algorithms include HMAC-MD5, RSA SHA-1, and RSA SHA-256. The hash is encrypted using the private key to produce the signature.

Both the data and its associated signature are transmitted to the recipient. Note that the data is not encrypted, merely signed. The recipient must have access to the public key that corresponds with the private key used to sign the data. In some cases a secure (trusted) public key distribution system such as a public key infrastructure (PKI) is used to make public keys available. In the case of DNSSEC, public keys are published within DNS, along with the resolution information and corresponding signature.

The recipient computes a hash of the received data, as did the data originator. The recipient applies the encryption algorithm to the received signature using the originator's public key. This operation is the inverse of the signature production process and produces the original data hash as its output. The output of this decryption, the original data hash, is compared with the recipient's computed hash of the data. If they match, the data has not been modified and the private key holder signed the data. If the private key holder can be trusted, the data can be considered validated.

DNSSEC Overview

DNSSEC utilizes this asymmetric key pair cryptographic approach to provide data origin authentication and end-to-end data integrity assurance. Any attempt to spoof or otherwise modify data en route to the destination will be detected by the recipient, that is, the resolver or more typically, its recursive/caching DNS server on its behalf. This feature makes DNSSEC an effective mitigation strategy against man-in-the-middle and cache poisoning attacks.

DNSSEC does not account for a secure key distribution system, so one or more trusted keys must be manually configured on the resolver or recursive name server. Each trusted key identifies the public key corresponding to a given trusted zone as authorized by the zone administrator. This is analogous to the bank notary being trusted by the bank to validate my identity. After all, any imposter may sign invalid zone data with a private key and publish the corresponding data, signatures, and public key. Thus the recursive server must be configured *a priori* with a key or set of keys that are trusted corresponding to trusted signed zones. The public key used by the trusted zone administrator must be conveyed to the resolver administrator. DNSSEC specifications do not specify the means for securely distributing these keys, so trusted keys must be communicated out of band.

A given trusted zone can authenticate a child zone's public key, extending the trust model from just the trusted zone to the trusted zone and its authenticated child zones. Likewise, these child zones can authenticate their children and so on, forming a *chain of trust* from the trusted zone to all signed delegated

zones. Ideally, the root zones would be signed and considered trusted zones, enabling resolvers to simply configure the root's public keys as trusted keys. This is scheduled to occur on July 1, 2010 as recent DNS cache poisoning publicity has given rise to substantial industry momentum to deploy DNSSEC.

Without root zone keys, configuration of trusted keys requires creation of a trust relationship with a given zone administrator to obtain his/her public key, whether by authenticated web access to the corresponding public key or other "out of band" mechanism. ISC has also created a trusted key registry as a repository of trusted keys for registered domains to enable "lookaside" validation in acting as a "parent zone proxy," reducing the requirement impact of forming individual relationships with each domain administrator.

Just think of all the domains your users access via email, browsers or other IP applications. To securely resolve them, trusted keys for each of these zones must be configured within recursive servers in the absence of root or TLD zone signing. With lookaside validation, the organization trusts the trusted key registry, which authenticates the zones for which it serves as signing "parent."

While not explicitly required by DNSSEC specifications, operational experience has led to the recommendation that two keys be used per zone: a zone signing key (ZSK) and a key signing key (KSK).* The ZSK is used to sign the data within the zone and the KSK is a longer-term key that signs the ZSK. Both the ZSK and KSK are each comprised of a public and private key pair. The private keys are used to sign zone information and must be secured, ideally on a secure server or host. The corresponding public keys are published within the zone file in the form of DNSKEY resource records.

The public KSK of the trusted zone is the trusted key configured in each recursive server. A trusted key is synonomous with a trust anchor, which is also known as a Secure Entry Point (SEP) into the DNS domain tree. This key is matched against the public KSK obtained from the zone during the resolution process. The resolved data's signature is validated using the zone's ZSK, and the ZSK is signed and its signature thereby validated by the trusted KSK.

If this KSK is not trusted, an attempt is made to check whether the parent zone is signed or if lookaside validation is configured. If signed, this parent zone (or lookaside regisry) signs its delegation to the child by signing the child's public KSK in the form of a Delegation Signer (DS) record (or DNSSEC Lookaside Validation, DLV record) in the parent zone. This delegation in turn is signed with the parent's ZSK, which is signed with the parent's KSK. Once again, if these signatures are valid and the KSK matches a trusted key, the resolution is complete and secure. Otherwise, the process continues with the parent's parent zone and so on.

The validation process works up the chain of trust to a matching trusted key, which if found, deems the resolution data validated. Otherwise, it will be considered insecure.

*The motivation for this recommendation and discussion of other DNSSEC operational practices are discussed in RFC 4641 (120).

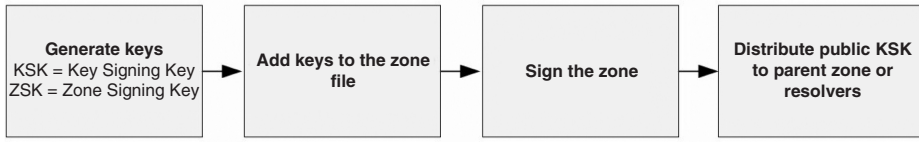

Figure 9-8: Basic DNSSEC implementation steps

Configuring DNSSEC

The process of implementing DNSSEC involves creating private/public key pairs, adding the public key information to the zone file to be signed, signing the zone, and distributing the public KSK information to either parent zone administrators or to resolver administrators who trust you and your zone information. Figure 9-8 illustrates the basic process.

Let's now illustrate this basic process looking at the mechanics for implementing DNSSEC. We'll illustrate the process using BIND methods and utilities (49) which support DNSSEC today. Microsoft supports DNSSEC*bis* in its Windows Server 2008 R2 release.

Generate Keys. Our first step is to generate keys that will be used to sign our zone information. BIND ships with the `dnssec-keygen` utility, which provides a simple command line to generate a private/public key pair. It even creates the DNSKEY record!

Add Keys to the Zone File. Before we sign the zone, we need to include our two DNSKEY resource records within the zone file. Since the .key files contain our DNSKEY resource records, you can either cut from the file and paste into the zone or simply use a $INCLUDE statement for each file. Don't forget to increment your serial number, too. As always, it's a good idea to run `named-checkzone` first, before running the `dnssec-signzone` utility.

Sign the Zone. The zone signature process utilizes another BIND utility, `dnssec-signzone`. The `dnssec-signzone` utility performs a number of functions to sign the zone. First, it canonically orders the resource records within the zone. This essentially alphabetizes the resource records within the zone. This facilitates grouping of resource records with common owner name, class, and type into resource record sets (RRSets) for signature application. The other reason for canonically ordering resource records is to identify gaps between RRSets within the zone file and population with Next SECure resource records that provide authenticated denial of existence of a given resource record within a zone. The `dnssec-signzone` utility also enables creation of NSEC3 records.

After canonical ordering and insertion of NSEC[3] records, `dnseec-signzone` signs the RRSets within the zone file, including DNSKEY RRSets (previously $INCLUDE'd in our example) and NSEC RRSets. The signed

zone file contains the original RRSets, canonically ordered and each signed with corresponding signatures stored as RRSet signature (RRSIG) records.

Fortunately the `dnseec-signzone` utility performs all of these steps automatically: canonical ordering, NSEC[3] insertion and RRSIG creation and insertion to render a signed zone. The output of the dnssec-signzone utility is the signed zone which uses the same name as the original unsigned zone, concatenated with a ".signed" suffix. Be aware that signing a zone increases its size tremendously. It also increases resolution packet sizes, given the extra RRSIG and NSEC information with each RRSet, not to mention the potential for additional message traffic to build the chain of trust back to a trust anchor.

Link the Chain of Trust. Now that the zone has been signed, you should determine its place in the chain of trust. That is, determine whether the parent zone is signed or not. If the parent zone is not signed and the newly signed zone is the top-level domain that is signed (i.e., zone apex; for example, `.com` is not signed yet!), recursive resolvers querying on behalf of stub resolvers must be configured with the zone's public KSK as a trusted key.

If the parent zone is signed, you'll need to provide the zone's public KSK to your parent zone administrator in the form of a DS resource record. The parent zone administrator will include the DS record in its zone file and re-sign the zone to link the chain of trust. Otherwise, if you use DNSSEC lookaside validation, provide the lookaside registry with the corresponding DLV resource record. Once the root and TLD zones are signed, this process will boil down to updating your parent zone with the updated DS record.

Key Rollover

The most administratively intensive issue with the ongoing management of DNSSEC deals with the process of key rollover, particularly KSK rollover. Like passwords, keys must be periodically changed. It's best to provide a moving target to would-be attackers! The use of separate key-signing vs. zone-signing keys helps the administration of this process. This is due to the fact that any zone administrator can simply re-sign his/her zone using a ZSK without affecting anyone else. Whatever ZSK is used, it is ultimately signed by the KSK, which may be configured as a trust anchor or referenced by a DS or DLV resource record in the "parent" zone. Thus, ZSKs can be changed at will. However, because KSKs are configured as trust anchors and are potentially referenced by other zones' DS or DLV records, they do impact other administrators and require a fairly tight integration process.

The two basic methods used for key rollover are pre-seeding the key, which is effective for rolling over ZSKs and the dual-key signature approach, which can be used to rollover KSKs. The key issue (no pun intended) with rollover relates to the updating of cached resolution and signature information in recursive servers and resolvers. When a resolver obtains authenticated resolution information, it will cache this information, including the DNSKEY records containing

ZSK and KSKs, for the duration of the original record TTL. After the TTL expires, the resolver must issue a new query for the corresponding information. If a zone administrator performs a flash cut of a new key for an old key, resolvers and recursive servers having performed queries with the old key, still valid per its TTL, will be unable to authenticate other data resolved within the zone. Therefore, maintaining a window during which key updates can be propagated and two or more keys are valid comprise the basic rollover techniques.

Let's discuss and compare the two common rollover strategies. ZSK and KSK rollover may occur independently. We'll assume the pre-seed strategy is applied to ZSK rollover, while dual signature is applied to KSK rollover. Examining first ZSK rollover, a second "passive" ZSK is created using the dnssec-keygen utility and its corresponding DNSKEY resource record is included in the zone file along with the active ZSK currently in use. The passive ZSK is made available for resolver and recursive server caching, but is not yet used to sign zone data. After both ZSKs have been added to the zone file, the zone can be signed with the still currently active ZSK and KSK.

Once published, the validity of the active ZSK exists until all slave servers obtain the zone file via zone transfers plus the key expiration time. The key expiration time should be longer than the zone or resource record TTLs. When this time has passed, the rollover time, the zone can be re-signed, this time using the now formerly passive ZSK. The formerly active ZSK can be removed from the zone file, and the zone re-signed with the new ZSK and the current KSK. Depending on the desired frequency of ZSK rollover, a new passive ZSK could be added to the zone at this point such that the zone would always have two ZSKs, one active and one passive. Otherwise, at this point only the active ZSK is included in the zone for signature.

Now let's examine the double signature rollover method, generally applied to KSKs. A new KSK is created using the dnssec-keygen utility with the –k option. Now sign the zone using the dnssec-signzone utility using *both* the current and new KSKs. The dnssec-signzone utility permits specification of multiple KSKs for zone signing. The public key of the new KSK must then be communicated to all resolvers/recursive servers that utilize this zone as a trust anchor. In addition (or more likely alternatively), the parent zone that authenticates this zone must be updated. An output of the dnssec-signzone utility when using the –g option includes a dsset-<zonename> file containing the corresponding DS resource record(s) that can be included in the parent's zone file. The –l option creates a dlvset-<zonename> file containing the corresponding DLV resource records.

The parent zone or DLV administrator must copy or include these DS or DLV records, respectively, and re-sign the parent zone. Given the manual configuration required to perform these tasks on the parent zone and resolvers/recursive servers, this time frame is less deterministic than the pre-seed method. Once this time has elapsed and the parent zone and trust anchor configurations have been updated, the old KSK may be removed from the zone file and the zone re-signed using only the newly current KSK.

Emergency rollover procedures should be devised in the event of compromise of a private key corresponding to an active KSK or ZSK. Should an attacker obtain the private key, he/she could forge zone data and sign it with the private key. Resolvers and recursive servers would authenticate the falsified data based on the corresponding published public key, which would appear valid. As we've seen, the ZSK can be changed autocratically and should be changed immediately. Changing the KSK however, does require broader involvement and coordination. We recommend documenting a process for emergency rollovers that includes the parent zone administrator and DLV registry contacts, as well as a means to communicate to users who have configured the KSK as a trust anchor. This could be via a registered email list or secure web site posting.

DNSSEC and Dynamic Updates

You may be wondering given that the zone signing process requires the canonical ordering of a zone, then signature, how does one insert a new resource record into the zone securely? First off, the update itself must be secure. The server should require signatures on update messages and should define which servers or networks may perform updates. Several BIND options provide for configuration of secure updates. When an update has been received and authenticated, it remains within the journal file (a dynamic update cache of sorts). Prior to BIND 9.6, to fully sign the zone with the update, the server must temporarily freeze dynamic updates using the `rndc freeze` command. This shuts off acceptance of dynamic updates. Once frozen, the zone must be resigned using the `dnssec-signzone` functionality. Then, dynamic updates may be re-enabled using the `rndc thaw` command.

BIND 9.6 has added an automated signing mechanims for dynamic updates, vastly simplifying this process. Along with its normal integration of journal updates into the zone file, BIND signs each update using the ZSK along with the corresponding "before" and "after" NSEC (3) records to canoncially insert the record into the zone. BIND 9.6 also periodically examines the zone for signatures nearing expiration. It will then automatically generate new signatures in such cases. To perform this automated signature process, BIND must have access to the ZSK private key to sign or re-sign records.

BIND 9.7 added more automation with key meta data for automated publication, activation, and revocation. Automated KSK trust anchor update is also supported in BIND 9.7.

Several DNSSEC tools are appearing on the market from DNS appliance vendors, IPAM vendors, and even freeware tools, such as those at dnssec-tools.org.

DNSSEC Deployment Considerations

BIND provides two critical utilities for the creation of keys and for signing zones to simplify the DNSSEC implementation process. However, a few key

issues remain that must be considered carefully when deciding to deploy DNSSEC:

- The size of a signed zone file is much larger than that of a corresponding unsigned zone file. This may affect required disk space and memory for large zones as well as zone loading time.
- The resolution response for a given query would also grow larger, given the attachment of RRSIG records and potentially DNSKEY records corresponding to the query. This could adversely affect query response time and performance.
- The resolution process performance may also be further adversely impacted by the trust anchor confirmation process, where keys and delegation signer records are validated up to the trust anchor zone or DLV registry.
- DNSSEC introduces the requirement for time synchronization given the absolute time references denoting valid and expiration times in RRSIG records.
- Zone footprinting by hopping NSEC records is a potential information over-exposure though the NSEC3 record makes this process more difficult. Consider whether zone footprinting is really an issue for you (generally information published in DNS is public information!) due to the computational complexity and hence time for generating NSEC3 records within a large signed zone.
- Key update procedures for initialization and rollover must be devised to provide authenticated access to updated KSKs via an out of band mechanism or via automated trust anchor update. The KSK public key update must be communicated to all who trust your zone as well as your parent zone or DLV, if any, in the form of a DS or DLV record, respectively.
- DNSSEC performs data origin authentication, data integrity verification, and authenticated denial of existence. It does not protect against other vulnerabilities. Don't forget to implement the mitigation tactics discussed earlier in this chapter to protect against other vulnerabilities.

10

IP ADDRESS MANAGEMENT BUSINESS CASE

No two enterprise IP networks are the same. And the corresponding management approaches likewise differ. Throughout this book, we've mentioned the use of spreadsheets and text editors as vehicles for IP management. These primitive "brute force" methods are certainly valid for any size organization, but they offer none of the benefits one can attain by deploying an IP address management system, such as automation, integration, reporting, and scalability. In this chapter, we'll discuss the business aspects of IP management, highlighting key areas where its implementation can provide a return on an investment in an IPAM tool. The exercise of creating a business case can not only help you identify the key aspects of IP management that may be lacking in your organization, and therefore inflicting high costs, but it can help you target your search among the multitude of IPAM system vendors in the marketplace. Knowing whether you need a complete overhaul or simply an improvement in a subset of key functions is important when considering alternative solutions. But it's up to you to investigate which solutions can best help you based on your particular environment.

Introduction to IP Address Management, By Timothy Rooney

BUSINESS CASE OVERVIEW

We'll start our business case analysis by considering key cost saving opportunities when implementing an IPAM system. As with any system you purchase, the intent is to reap savings of time as well as reduced errors and rework. These savings represent the "income" portion of the business case. The costs of a tool and its implementation, support, and ongoing management represent the expense side of the equation. In most cases, investment in an IP management tool requires an up-front purchase payment. This initial outlay will be "paid back" over time as savings mount. Note that managed IPAM services, either in the form of consultants operating your IPAM system of choice or actual managed services from an IPAM service provider offer an additional vehicle for managing IPAM within an organization. The expense side of our business case example is applicable in these cases, but the savings and "investment" side will depend on capital requirements and ongoing services charges.

As we examine the income and expense sides of the ROI equation, we'll walk through an example to illustrate this payback. The base assumptions for our example are as follows. Again, every network is different and only your input parameters will map the relevance to your network.

Macro IP address blocks:	40	DHCP servers:	16
IP subnets:	380	DNS servers:	24
Assigned IP addresses:	75,000	Routers:	48

The general parameters of IPAM operation are as follows:

Ten percent of blocks, subnets, IP addresses are allocated and assigned on average per year

Five percent of blocks, subnets, IP addresses are deleted on average per year

Five percent of blocks, subnets, IP addresses are changed on average per year

Of changes, about 75 percent are parameter changes, 25 percent moves

Of IP address assignments, 75 percent manual, 25 percent MDHCP

Router and IP discoveries are run once a week

BUSINESS CASE COST BASIS

To derive an accurate cost estimate, you will need to analyze the manual labor costs of performing IP address management tasks over a month or so to sample how much IP address management is costing your organization on a daily or weekly basis. If this sounds like a time and motion analysis, it is! And this practice from the field of industrial engineering has been helping maximize the efficiency of manufacturing and other process-oriented workflows for decades. Certainly, if you are comfortable making rough estimates you may do so to save time. People often remark that labor savings are "soft costs," and thus it's difficult to justify

the investment in IP management tools. If this argument had deterred Henry Ford, perhaps we might still be riding horses. Service providers will typically have additional income components to the business case with the very necessity of IPAM-related functions in providing services to paying customers. This revenue may be for broadband or WAN service or for a managed IPAM service and render investment in an IP management tool easier to justify.

If you desire to quantify these labor costs for your environment, record the time spent on every IP management task (in a spreadsheet if desired!), from block allocation, to address assignment to configuring DHCP and DNS servers. We covered the details of these tasks in Chapter 6, though we'll summarize them here and group common sets of tasks. When considering each task, don't just record the time it takes to type in the information. Consider the "think time" required to select an appropriate subnet or IP address, to define the DHCP address pool parameters, to create the zone file or file changes, and so on. After all, the think time is the hard part, where mistakes can be made creating future rework and hence more time. If a resource is unavailable, such as a server requiring an update or a spreadsheet in use by another, record this wait time as well; it adds into the lost productivity of the administrator doing the waiting, and perhaps the end user(s) also awaiting the change. Help desk calls referred to the IP team should also be recorded in terms of duration, impact on the end user community if outage-related, and time taken for help desk staff logging of the trouble, referral to the server or networking team, problem isolation and resolution, and closure of the trouble with feedback to the help desk. This type of information is usually readily available from your help desk phone system or trouble ticketing system.

We refer to an "outage" in this chapter as a situation where a client is unable to obtain an IP address (fully booked address pool, duplicate address, server outage, incorrect DHCP configuration, etc.) or resolve an IP address (server or network down, incorrect DNS configuration, etc.) Such outages themselves may provide sufficient justification to implement a more rigorous IP management process and tool.

Outages represent one form of an exception condition, where work is generated that requires an interrupt level prioritization to complete, at the expense of whatever else was being worked on. Other forms of exception conditions might include periodic auditing requirements for internal or regulatory controls, major project planning and execution, such as expanding the network or opening a new store, and performing upgrades of DHCP and DNS servers. These tasks are beyond the day-to-day moves, adds, and changes, and usually require concentrated resources for bursts of time. The length of time and resources required for the burst depends on the scope of the work.

The following discussion breaks down common IP management tasks including exception conditions. For each event, determine the amount of time spent per resource type, assuming different pay scales. Normalize the time, including exceptions to a common time of a month or year. Multiply the time by the cost per unit time and sum to derive your costs for your current IP management

methodology. Review the areas that are high cost runners for your organization. This will be helpful when searching for an IP management system, to focus your search on reducing those particular cost areas.

Address Block Management

If your network address plan accurately reflects your production addressing hierarchy, you may think there's no need to focus on address block allocations. In this context, we're referring to "blocks" as those allocated above the subnet level in the address space hierarchy, for example, recall the application, continental and regional allocations we discussed in Chapter 6. As long as you have your plan well documented, the underlying subnets and address assignments align with the plan, and the rate of change of network allocation is low, you're probably right. However, should one of these three conditions not hold, you should consider the cost of documenting the plan, renumbering your network to attain alignment, and updating your allocation process to prevent "misaligned" allocations in the future and to accommodate even occasional moves, adds, and changes. Stabilizing this information and the processes that influence it provide a solid foundation for the more frequent downstream subnet and IP allocation tasks, which in turn will be performed with fewer costly errors.

Another problem with minimizing the address block allocation function is that the allocation plan feeds not only downstream allocations, including subnet allocations, router provisioning, and corresponding router protocol updates and tables, but also subnet-resident static address assignments, DHCP pool configuration, and DNS domains and resource records updates. Another point to consider is that business initiatives may impact the address block allocation plan. Planned deployment of many new sites requiring IP addresses due to expansion of the business, plans for new service offerings such as voice over IP, and even a merger or acquisition each can heavily impact the IP address plan.

Planning address space allocations for any of these business initiatives begins with understanding the business requirements with respect to address capacity for each application at each site. This in itself can be a tedious process. Once this capacity has been quantified, this should be folded into the overall IP allocation plan. For example, if we embarked on an expansion into Latin America, the IP planners may decide to deploy a core level router to participate at the top level of the corporate backbone. Allocating address space for each suballocation "beneath" this router in the topology requires updating of the spreadsheet or database, enumerating the locations and capacity requirements. Aggregating these capacity requirements and adding some additional space for growth, the allocation may be sized. From this allocation, suballocations to each set of locations by application may commence.

The key steps involved in address block allocation include identification of address capacity needs based on the defined block hierarchy, rollup of capacity requirements within the hierarchy, sizing up the proper allocation based on the capacity requirements, recording allocations and suballocations, and updating

corresponding core routers. If a new allocation is driven by a major business initiative, such as our expansion into Latin America, it's likely that additional DHCP and DNS servers will be required to service the end users within the region. Thus, we've included tasks for DHCP and DNS server sizing (how many servers of a given size of each type are needed where), procurement of the servers, then base-level server configuration. This base configuration would include basic policy definition, as well as zones corresponding to the domain plan for the addition. A project manager resource or team would also likely be required to coordinate and manage the deployment process to completion.

Table 10-1 illustrates an example block allocation cost analysis for this scenario. While not explicitly enumerated in the table, updating of relay agent information in routers to relay to the DHCP server(s) is another required task. Note that additional costs for DHCP and DNS server hardware, shipping, travel, and expenses to have them installed are not shown in the example below but need to be considered in the overall ROI analysis. Note also that these costs, as

TABLE 10-1: Block Allocation Cost Analysis

Block Allocation	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Identify capacity requirements	Business team	160.00	$30	$4,800	4	$19,200
Design Rollup to address capacity (new)	Engineering	40.00	$50	$2,000	1	$2,000
Design Rollup to address capacity (build on existing)	Engineering	2.00	$50	$100	3	$300
Address plan update	Engineering	0.50	$50	$25	4	$100
Router provisioning of new network(s)	Router Ops	0.25	$40	$10	4	$40
DHCP and DNS server sizing	Engineering	4.00	$50	$200	4	$800
DHCP and DNS server procurement	Engineering & Purchasing	24.00	$50	$1,200	1	$1,200
Basic DHCP and DNS server configuration	Server Ops	1.00	$40	$40	4	$160
TOTAL						**$23,800**

well as sizing, procurement, and installation, will likely be incurred whether an IPAM system is used or not.

The largest cost component of block allocation relates to analyzing capacity requirements to determine the need for a new allocation or carving up of an existing one. This task likely involves multiple team members, for example, from facilities planning, human resources, and related business functions, or from product management and marketing for a service provider organization. As such, four staff members for one week amounts to 160 hours for each of our four allocations per year. Assuming only one of these four involves a new allocation, about a week of rollup planning and block design is assumed. Recall that multiple parallel allocations may be needed to maintain allocation consistency. For example, we may allocate five blocks per location for various applications, so adequate allocations must be designed.

For all allocations, the block inventory database must be updated and corresponding routers provisioned. DHCP and DNS services requirements must be assessed, and if needed, additional servers procured and configured. On average, let's assume we currently have 40 DHCP and DNS servers deployed across 40 blocks. Applying this 1:1 ratio to our four annual allocations, linear growth would predict the requirement to add four DHCP and DNS servers annually, which entails a single procurement process with subsequent configuration of each server.

Address block management also entails deletion of address blocks as well as moving blocks. Block deletions may result from the withdrawal from a major business market or consolidation of sites for example. The deleting of an address block within the hierarchy is a bottom-up process, the converse of the top-down allocation process. All downstream IP addresses, subnets, address pools, resource records and domains must first be decommissioned before the macro level block can be freed up for future assignment consideration. We'll discuss these "substeps" below, so the process of deleting a top or intermediate level block requires iterative execution of IP address and block deletion tasks, once for each address and subnet respectively. As with a modest to large allocation task, project planning resources may be required to verify deletions up the hierarchy. Table 10-2 highlights the relatively inexpensive block deletion task taken in isolation, which occurs only twice a year in our case.

Moving, or renumbering of address blocks combines the allocation process, which should be performed from a top down allocation to move (reassign) underlying subnets and IP addresses to the newly allocated block, and the deletion process, which frees up address space from the bottom up as addresses are moved away. In Table 10-3, we've combined the block modification task, which comprises about two of the three annual block change tasks and the block move task, comprising just one of the three annual block changes. Modifying block attributes is typically only a database update and is therefore trivial.

Movement of blocks, however, is a bit more involved. In our scenario, with one annual block move on average, a destination block needs to be identified and allocated, affected routers and DHCP/DNS servers provisioned, and the

TABLE 10-2: Block Deletion Cost Analysis

Block Deletion	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Identify block and downstream blocks and subnets	Engineering	0.10	$50	$5	2	$10
Verify downstream blocks and subnets are free	Engineering	1.00	$50	$50	2	$100
Delete block and verify	Engineering	0.10	$50	$5	2	$10
Join newly freed block with contiguous free block	Engineering	0.25	$50	$13	2	$25
TOTAL						**$120**

move executed. As the move progresses, monitoring and verification of successive IP address moves within downstream subnets enables deletion of the former addresses. A final audit validation, requiring 10 staff-hours, is conducted prior to the final deletion of the block and completion of the move. In all moves or changes, the IP inventory must be updated to reflect changes.

Considering what we've discussed so far, our total costs for block management are $24,670, most of which is comprised of block allocations for expansions.

Subnet Management

This basic task of subnet allocation involves the identification of a subnet that is available that rolls up within the address allocation plan for the given location and application, and assignment of the subnet in the IP address plan database. In addition, this task requires provisioning of the subnet on the appropriate router interface, defining and updating DHCP server configurations for address pool(s) with corresponding options and/or client class parameters to be defined on the allocated subnet, defining and updating DNS server configurations with domain updates (e.g., in-addr.arpa and ip6.arpa domains) and resource record updates for name servers and statically assigned addresses).

TABLE 10-3: Block Change Cost Analysis

Block Changes	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Identify block to move/change and downstream blocks and subnets	Engineering	0.50	$50	$25	2	$50
Modify block attributes	Engineering	0.25	$50	$13	2	$25
Identify destination block to which to move block	Engineering	0.50	$50	$25	1	$25
Address plan update	Engineering	0.50	$50	$25	2	$50
Router provisioning of new network(s)	Router Ops	0.25	$40	$10	1	$10
Basic DHCP and DNS server configuration	Server Ops	1.00	$40	$40	1	$40
Verify rolling move of IP subnets and blocks, and decommission moved blocks	Engineering	10.00	$50	$500	1	$500
Verify completion of all moved subnets and deleted former subnets	Engineering	1.00	$50	$50	1	$50
TOTAL						**$750**

The subnet allocation process illustrates the tight inter-relationship among address allocation, assignment, and DHCP and DNS server configuration tasks. Depending on your business processes, subnets may be allocated or reserved prior to address assignment and DHCP/DNS configuration. Nonetheless, this set of steps will typically be required to bring a subnet into production.

Devices to be assigned addresses on the subnet now and in the future will likely require name resolution information in DNS. At a minimum, this information applies to a forward domain for domain name-to-IP address lookup and a

TABLE 10-4: Subnet Allocation Cost Analysis

Subnet Allocation	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Define and allocate subnet	Engineering	0.20	$50	$10	38	$380
Router provisioning of new subnet	Router Ops	0.25	$40	$10	38	$380
DHCP pool definitions—define range, apply options and client classes	Engineering	0.50	$50	$25	38	$950
DHCP server configuration	Server Ops	0.50	$40	$20	38	$760
DNS server config updates for allocated subnets	DNS Eng	0.25	$50	$13	38	$475
Allocation deployment verification	Net Ops	0.50	$50	$25	38	$950
TOTAL						**$3,895**

reverse domain for the reverse lookup. Of course, these domains must exist and be configured on the DNS server. Depending on your domain topology, adding a new subnet to an existing location will likely utilize an existing domain, though this is certainly not necessarily the case. Likewise, adding a new subnet to a new location may or may not require a new domain declaration. In either case, a new domain may need to be defined and configured as a subdomain or a new zone on the appropriate DNS server. In the same way, the reverse domain corresponding to the subnet address may need to be added as well, unless a higher layer in-addr.arpa or ip6.arpa zone will house these resource records.

In our environment, 38 subnets will be allocated this year, based on the assumption of 10 percent of their current 380 subnets. The allocation process involves six basic tasks as shown in Table 10-4, each requiring a half hour or less.

Deleting a subnet may be required when closing a site or consolidating address space. Based on our initial assumptions, we expect to delete 19 subnets

TABLE 10-5: Subnet Deletion Cost Analysis

Subnet Deletion	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Identify subnet and associated IP addresses	Engineering	0.10	$50	$5	19	$95
Verify subnet IP addresses are free	Engineering	0.25	$50	$13	19	$238
Delete subnet and verify	Engineering	0.10	$50	$5	19	$95
Join newly freed subnet with contiguous free block	Engineering	0.25	$50	$13	19	$238
TOTAL						**$665**

in the coming year. Devices with IP addresses on each subnet to be deleted must first be moved or decommissioned such that the subnet is free of address assignments (other than perhaps subnet-serving routers). After all IP addresses have been verified as free, the subnet may be reclaimed into the free address space for future allocation. Subnet deletion costs are displayed in Table 10-5.

Moving a subnet could involve one of two results: movement of the subnet and its assigned IP addresses to another router or interface, preserving the current address assignment or movement to another router or interface, requiring a new subnet address. The first case requires consideration of address space rollup within the hierarchy but generally consists of modifying and verifying router provisioning compliance with the plan, as well as updates to routing tables and DHCP relay addresses as necessary.

Movement of a subnet that requires an address change due to a physical move, for example, when an office is moved, is inherently disruptive. The destination subnet may be allocated and provisioned on the destination router interface, along with the other tasks described above related to reserving static addresses and updating DHCP and DNS configurations. When each moved device plugs in, it will need to be manually re-addressed with the new address and/or obtain a DHCP lease on a pool relevant to the subnet.

In our case, of the 19 subnet changes this year, 14 will involve simple subnet attribute updates and 5 will involve disruptive moves. The corresponding tasks and costs are so numbered by occurrences per year in Table 10-6.

TABLE 10-6: Subnet Change Cost Analysis

Subnet Changes	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Identify subnet to move/ change and affected IP addresses	Engineering	0.50	$50	$25	19	$475
Modify subnet attributes	Engineering	0.25	$50	$13	14	$175
Identify destination subnet to which to move subnet	Engineering	0.25	$50	$13	5	$63
Address plan update	Engineering	0.25	$50	$13	5	$63
Router provisioning of new network(s)	Router Ops	0.25	$40	$10	5	$50
DHCP pool definitions for new subnet	Engineering	0.50	$50	$25	5	$125
DHCP server configuration	Server Ops	0.50	$40	$20	5	$100
DNS server configuration for destination subnet	DNS Eng	2.00	$50	$100	5	$500
Verify rolling move of IP addresses and decommission moved IP addresses	Engineering	2.00	$50	$100	5	$500
Verify completion of subnet with all IP addresses moved and old ones deleted	Engineering	0.50	$50	$25	5	$125
TOTAL						**$2,175**

TABLE 10-7: IP Address Assignment Cost Analysis

IP Address Assignment	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Identify available IP address	Engineering	0.05	$50	$3	7,500	$18,750
Update device configuration in DHCP server (M-DHCP)	Server Ops	0.10	$40	$10	1,875	$7,500
Static address assignment—email to owner then manual config	Engineering	0.10	$50	$13	5,625	$28,125
Update DNS for manually configured IP addresses	Server Ops	0.05	$40	$2	5,625	$11,250
Address assignment verification	Net Ops	0.05	$50	$3	7,500	$18,750
TOTAL						**$84,375**

Our total annual subnet management costs amount to a relatively modest $6,735, reflective of the relative payoff in diligent block allocation. If the work of capacity planning is performed well for block allocations, the subnet allocation and management process requires modest bookkeeping and little in the way of locating additional address space.

IP Address Assignment—Moves, Adds, Changes

Assigning, deleting, and reassigning IP addresses to individual hosts is usually the most frequent IP management activity in most organizations. This is typically associated with deployment, redeployment, or decommissioning of new devices, including routers, servers, printers, and the like. The IP address inventory database should be consulted to identify an available IP address. If possible, it would be useful to ping the IP address to be assigned just to verify accuracy of the inventory, though we'll discuss inventory assurance as a separate task. The IP address to be assigned should then be noted as assigned to the given device.

TABLE 10-8: IP Address Deletion Cost Analysis

IP Address Deletion	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Identify IP address to delete	Engineering	0.05	$50	$3	3,750	$9,375
Delete subnet and verify	Engineering	0.05	$50	$3	3,750	$9,375
TOTAL						**$18,750**

The actual physical IP address assignment may be performed by manually configuring the device or by using DHCP (in the DHCP case, we assume Manual DHCP (M-DHCP) would be used to assign the designated IP address to the corresponding host). In the non-DHCP case, the address must be entered directly on the device, so unless the IP address assigner is also responsible for the physical assignment, this process would entail an email or phone call to the device owner conveying the assigned IP address to be entered. When using M-DHCP, an entry in the appropriate DHCP server's configuration would be necessary to map the device's hardware address to the assigned IP address.

Most devices with IP addresses will require corresponding DNS resource records to enable their name resolution. Using the DHCP method of address assignment, the DHCP server can be configured to update a master DNS server upon assignment of the IP address. This update would affect the forward domain for domain name-to-IP address lookup and the reverse domain for the reverse lookup. A similar DNS update would be required if assigning the address manually. Updating DNS with this new host information may entail editing or updating the corresponding zone files on the server or by sending dynamic updates. We expect to perform 7,500 IP address assignments this year, with 1,875 of these performed via Manual DHCP and the remaining 5,625 requiring manual configuration. The corresponding cost estimates for these tasks are shown in Table 10-7.

De-assigning or deleting an IP address is relatively straightforward: delete the IP address in the IP inventory, remove the M-DHCP entry from DHCP if appropriate, and remove associated DNS resource records. Care must be taken to assure the address has been relinquished by the device before assigning it to another device. Thus assigning it with a state of "pending deletion" or something similar would alert other administrators not to assign that address to another device until confirmation is received of its availability. We expect 3,750 IP address deletions on average per year as illustrated in Table 10-8.

As with blocks and subnets, changes for IP addresses could involve simple attribute changes within the inventory or a more demanding move process. Of

TABLE 10-9: IP Address Change Cost Analysis

IP Address Changes	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Identify IP address to move/change	Engineering	0.05	$50	$3	3,750	$9,375
Modify IP address attributes	Engineering	0.10	$50	$5	2,813	$14,065
Identify destination IP address	Engineering	0.10	$50	$5	937	$4,685
Update device configuration in DHCP server (M-DHCP)	Server Ops	0.10	$40	$10	234	$936
Static address assignment—email to owner then manual config	Engineering	0.10	$50	$13	703	$3,515
Update DNS for manually configured IP addresses	Server Ops	0.10	$40	$4	703	$2,812
Address move verification	Net Ops	0.10	$50	$5	937	$4,685
TOTAL						**$40,073**

3,750 IP addresses requiring change, 2,813 (75 percent) will involve attribute updates. Of the remaining 937 addresses requiring movement, 234 will utilize Manual DHCP and 703 will require manual configuration. Moving the IP address is a combination of assigning an IP address on the destination subnet then deleting the current IP address on the current subnet after the move. Inventory assurance strategies can be employed to track the address move and to maintain database accuracy. These costs are summarized in Table 10-9.

Our individual IP address moves, adds, and changes cost about $143,198. While each address assignment task may take just minutes, this task is performed thousands of times throughout the year, yielding a rather astonishing total cost.

Inventory Assurance

The inventory assurance task seeks to maintain IP inventory database accuracy. Like any "inventory system," the IP inventory should reflect the block, subnet, and address assignments on the actual network. This information is relied upon for future planning, troubleshooting, and auditing. Inventory assurance also provides an opportunity to analyze why a discovered discrepancy exists and can be critical in detecting violations in network access policies, change control procedures, or provisioning and service level agreements. Finally, inventory assurance processes can help identify reclaimable address space to make the best use of address space and accurately model actual use in the inventory database.

The process of inventory assurance involves discovering network information, comparing discovered results with the inventory database or "plan of record," then following up by updating the database or otherwise investigating the discrepancy. Any changes to the inventory could also affect associated DHCP and/or DNS server configurations, as with subnet or address adds or deletes. In our case, we'll run SNMP sweeps of our routers once a week (2,496 times per year total), and subnet ping sweeps once a week (19,760 subnet sweeps per year). The results are then visually compared with the corresponding subnet and IP address in our spreadsheets. We're assuming about 15 minutes to compare each router's data and about 6 minutes per subnet. Each discrepancy requires the analyst to think and decide whether to accept the discrepancy, and update the database and associated DNS and DHCP configurations, or denote the IP address as an open issue that must be further investigated. Such investigation may involve analyzing discovery history of the IP address, identifying the owner of the discovered device, or even physically locating and inspecting the device. Inventory assurance costs are shown in Table 10-10.

Address Capacity Management

Inventory assurance discovery tasks provide enumeration of occupied IP addresses within a subnet. This information is helpful not only in maintaining IP inventory accuracy but in determining the overall IP address utilization across our subnets considering assigned vs. total (assigned + available) IP addresses. Recall that addresses on a subnet can be assigned manually, automatically via autoconfiguration, or via DHCP. Ping sweeps or similar host discovery methods enable detection of manual or autoconfigured addresses, while collection of DHCP lease information supplements these discovery methods, providing the added perspective of the DHCP server. Monitoring address pools in particular helps assure such pools are adequately sized to meet the IP address demands of each site.

The first step in the process is to collect DHCP lease files. We collect files from each of its 16 DHCP servers once each business day (270/year). This amounts to 4,320 lease files per year. These files are collected primarily for address utilization monitoring, but this information can also be used to validate

TABLE 10-10: Inventory Assurance Cost Analysis

Inventory Assurance	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Query routers via SNMP for subnets	Router Ops	0.05	$40	$2	2,496	$4,992
Compare discovery results with router-subnet inventory	Engineering	0.25	$50	$13	2,496	$31,200
Run ping sweep or other discovery method on subnet	Engineering	0.10	$50	$5	19,760	$98,800
Compare discovery results with IP address inventory	Engineering	0.05	$50	$3	19,760	$49,400
Assess discrepancies and update inventory or investigate cause	Engineering	0.25	$50	$13	1,113	$13,910
Update the inventory with accepted changes; make associated changes to DHCP and DNS if necessary	Engineering	0.10	$50	$5	890	$4,450
TOTAL						**$202,752**

the pool sizing parameters on each subnet within the IP inventory database. These two tasks, comparison of pool size and analysis of addresses assigned are combined in Table 10-11. Analysis of the results indicates that on average 5 percent of these lease files (216 occurrences) indicate additional capacity is required. These capacity enhancements can be provided by supplementing existing pools or adding new ones on existing subnets. The addition of pools requiring subnet allocation will be assumed a contributor to the 38 subnet allocations performed each year.

The last subtask within address capacity management involves analyzing several of these lease file "snapshots" to identify address utilization trends. Slated

TABLE 10-11: Address Capacity Management Cost Analysis

Address Capacity Management	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Collect DHCP lease files	Server Ops	0.10	$40	$4	4,320	$17,280
Analyze lease files to identify discrepancies in pool configuration vs. inventory as well as capacity utilization	Engineering	0.25	$50	$5	4,320	$54,000
Trigger block or pool allocation process for any pools at or nearing exhaustion	Engineering	0.10	$50	$5	216	$1,080
Update DHCP server configurations	Server Ops	0.25	$40	$10	216	$2,160
Analyze trends in utilization to initiate plans for address expansion or contraction	Engineering	2.00	$50	$100	12	$1,200
TOTAL						**$75,720**

for once a month, our staff analyze the last three months' counts per pool to identify pools where address demands are growing or shrinking. This can be helpful in identifying those pools requiring closer attention in the analysis of daily lease file collections in the coming month to minimize address depletions.

Auditing and Reporting

Auditing and reporting are key functions of any management system and are absolute requirements for any IP management system. In fact, as we look at the costs for manual audit tracking and reporting, we'll see that automating such tasks provides financial benefits as well. With that said, the intensity of auditing and reporting requirements varies across organizations, based upon organizational reporting policies and industry and government compliance laws and

TABLE 10-12: Auditing and Reporting Cost Analysis

Auditing and Reporting	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Analyze DHCP lease files and log history of IP-MAC assignments	Engineering	0.25	$50	$25	4,320	$54,000
Log all IPAM related changes made by staff	Engineering	0.02	$50	$1	17,361	$17,361
Run reports on demand	Engineering	1.00	$50	$50	100	$5,000
TOTAL						**$76,361**

regulations. At minimum, organizations need to be able to research which administrator performed a certain IP management function for troubleshooting, training, feedback, or accountability tracking. It's also very useful to track which device occupied a given IP address at a certain point in time. If a firewall log scan indicates that a particular IP address potentially violated an internal policy for example, the ability to map the suspect IP address to a device or user within the given time frame of the incident can save a tremendous amount of time.

For our example network, the auditing process entails updating a consolidated audit log with detected IP address assignments and associated MAC addresses for each DHCP lease file collected. In addition, an audit log for tracking administrator moves/adds/changes of IP blocks, subnets, addresses, DHCP and DNS server configurations, and upgrades should be maintained. Unless one person makes all such changes, this information is difficult to maintain manually. This generally relies on every person making such a change to remember to log the change manually or send an email to a centralized person or team for consolidated logging. If this information is consolidated, the running of reports is theoretically simple. Auditing and reporting costs are summarized in Table 10-12.

Server Upgrade Management

Upgrades may be required for DHCP and DNS servers due to security vulnerability fixes, bug fixes, or new feature enhancements. Let's assume we perform upgrades for each server three times a year on average, or a total of 120 total upgrades. Of these, 5 percent require a coincident operating system upgrade or

TABLE 10-13: Server Upgrades Cost Analysis

Server Upgrades	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Assess OS, DHCP, DNS version compatibility	Engineering	0.50	$50	$25	120	$3,000
Install upgrade on lab server	Engineering	8.00	$50	$400	3	$1,200
Obtain OS patch for compliance	Engineering	0.10	$50	$5	6	$30
Install OS patch	Server Ops	2.00	$40	$80	6	$480
Install DHCP/ DNS upgrade	Server Ops	4.00	$50	$200	120	$24,000
Verify successful installation and reachability	Engineering	1.00	$50	$50	120	$6,000
Update the inventory with current versions on the upgraded server	Engineering	0.50	$50	$25	120	$3,000
TOTAL						**$37,710**

patch for compliance with the DHCP/DNS upgrade. Upgrades should be installed and tested in a lab environment to verify expected behavior prior to upgrading production servers. This lab test and soak time varies depending on policies but we'll assume eight hours for installation and testing per upgrade. Table 10-13 highlights these costs.

IPv6 Implementation

You may still consider IPv6 among your longer-term projects, despite recent notifications from Regional Internet Registries indicating that public IPv4 space may be depleted within a few years. In our scenario, private address space is plentiful and we expect to retain our IPv4 public allocation. However, we have also allocated IPv6 space, both public and unicast local, in order to get ahead of the curve as IPv6 becomes more prevalent over the next decade. This will enable us to gain experience with the protocol and to support Internet reachability of relevant email and web servers to those users who ultimately will be forced to use IPv6 addresses after IPv4 exhaustion.

TABLE 10-14: IPv6 Migration Cost Analysis

IPv6 Deployment	Resource Type	Average Staff-Time Required (Hours)	Resource Cost	Cost Per Event	Events Per Year	Annualized Cost
IPv6 address management tools	Maint.	1.00	$25,000	$25,000	1	$25,000
IPv6 expertise/ staff	Engineering	0.50	$140,000	$70,000	1	$70,000
IPv6 training	Training	1.00	$30,000	$30,000	1	$30,000
TOTAL						**$125,000**

In terms of managing IPv6 address space, the model is in many ways quite similar, but also very different. Managing an IPv4-IPv6 coexistence and gradual migration will require adeptness at spreadsheet mathematics or use of a dual protocol IPAM system. In terms of this ROI example, we will lump our IPv6 expenses into a single table for one resource at half time, maintenance on an IPv6 tool (ignoring capital), and training for the resource and others within the support and IT organizations. In reality, IPv6 deployment results in replication of every task and expense table we've just reviewed, and we'll assume this has been inherently included. Table 10-14 covers only the incremental costs, though we haven't even included staff time spent on training and associated travel expenses in our analysis so it is rather conservative.

Outage Recovery Costs

IP address depletions, DHCP or DNS server outages, and errors in IP moves/adds/changes or DHCP/DNS server configuration can lead to the unavailability of these critical network services to end users. If address pools are diligently monitored, IP address depletions should be rare, though a few may occur due to planned or coincidental meetings in a given location causing a spike in demand. We'll assume that we experience no such outages due to our astute investment in proactive pool monitoring. If such an outage were to occur, however, those users attempting to obtain IP addresses after the pool becomes fully utilized will fail to do so. Unable to connect to the network, some may call the help desk complaining about the network being down. After collecting some information about the issue, the help desk staff would escalate to Engineering and Operations for troubleshooting and resolution.

Configuration errors at any level of the IP management process can create outage conditions resulting in the same trouble flow from end user to the help desk to Engineering and Operations. A miscalculated block or subnet allocation

could result in overlapping allocations, which can create routing loops or routing issues. Deleting a subnet or IP address prior to its relinquishment would stimulate complaints from affected end users. Likewise, erroneous DHCP and DNS server configurations can lead to improper name resolution with DNS, incorrect device initialization with DHCP or failure to load configuration files, which renders the DNS or DHCP service unavailable.

In our scenario, we expect no capacity related outages. Configuration errors are assumed to affect 1 in 200 (0.5 percent) IP management related changes. This amounts to 87 errors of the total 17,000+ IP management changes made within a year (based on all of the preceding tasks we've reviewed in this chapter). Each such outage is expected to last about half an hour, including the initiation of the outage, the detection by administrators or end users, reporting back to those responsible for resolution, isolation resolution of the problem and finally closure with those reporting the issue. Thus about 43.5 hours of outages ("outage-hours") are expected due to a configuration error.

DHCP and DNS server outages may occur due to server failure, power outage, or from an end user perspective, network reachability issues. Targeted at four nines availability (99.99 percent), We anticipate that each server will suffer up to 0.01 percent downtime or 0.88 hours per server or about 35 hours for all servers. Thus we expect about 43.5 + 35 = 78.5 outage-hours per year.

With about 40 servers servicing 17,000 employees, the average server outage will impact about 425 end users. However, due to redundancy, we'll assume only half are impacted, plus we'll apply a simple proportional probability of occurrence during working hours as about 24 percent (annual working hours/annual total hours). The net impact is 51 end users/outage-hour. Each outage will affect some Help Desk staff as well as Engineering and Operations staff members who will identify and resolve the outage on average within half an hour.

Considering these impacted resources per outage, we derive a cost per outage-hour of $1,882.50. This is the sum over each resource type's cost per hour times quantity impacted per outage. Given that configuration errors result in 43.5 outage-hours per year, the total cost of these errors is $1,882.50 × 43.5 = $81,889. Similarly, server outages at 35 outage-hours/year cost the organization $65,888. Total outage costs are thus $147,777. Table 10-15 summarizes the derivation of these costs.

IP Management System Administration Costs

System administration tasks generally entail backing up IP inventory data, DHCP and DNS server configurations, audit history, and related IP information. Depending on the system currently in use, such as an in-house developed system, ongoing development, testing and support costs need to be considered. New features may be required periodically in support of new DHCP or DNS features, for IPv6 support, or other enhancements. Even absent new development, support of the system to address bugs encountered or upgrade incompatibilities must be accounted for.

TABLE 10-15: Outages and Recovery Cost Analysis

Outages and Recovery	Affected Resource Cost/hr	Resources Impacted per Outage	Cost per Outage-Hour	Outages	Outage Hours	Annualized Cost
Customer Care/ Help Desk	$30	0.25	$7.50			
Eng/Ops	$45	2	$90.00			
End users	$35	51	$1,785.00			
Total cost per outage-hour			*$1,882.50*			
Configuration errors				*87*	*43.5*	$81,889
Address Pool Depletions				*0*	*0.0*	$0
Total Incidents				87		
Server outages (unavailability)					*35.0*	$65,888
Total outage hours per year					78.5	
TOTAL outage costs						**$147,777**

For a commercial off the shelf system, annual maintenance, training and support costs need to be considered. If considering a release upgrade, any costs associated with an upgrade should be considered. We've already addressed server upgrades, so in this context, upgrades refer to the product costs, associated software, hardware and staffing required to perform the upgrade of the management system. Some systems can be upgraded very quickly and easily while others are more cumbersome. The size and distribution of IP management components on your network also plays strongly into the upgrade costs. If possible, annualize this cost by determining how many such upgrades are planned over the next three years and averaging the total cost (divide total cost by 3 years) or map out costs on an annual basis for three years or the length of your planning horizon.

Assuming we currently use spreadsheets, the only system administration cost is the weekly backup of the spreadsheet and all DHCP and DNS server configurations to an off-site server, which takes about an hour a week and costs just over $2,000 annually. We've listed other cost components discussed in Table 10-16 for completeness if you are using a spreadsheet alternative, either an in-house system or a commercial system.

Cost Basis Summary

As summarized in Table 10-17, the total annual costs for execution of our IP management tasks are $842,003. Seemingly minor daily tasks certainly add up!

TABLE 10-16: IPAM System Administration Costs

System Administration	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Data repository backup	Server Ops	1.00	$40	$40	52	$2,080
Other system administration costs	Server Ops	0.00	$40	$0	52	$0
In-house system analyst staff	Analyst	0.00	$0	$0	0	$0
In-house system development staff	Dev.	0.00	$0	$0	0	$0
In-house system testing staff	Testing	0.00	$0	$0	0	$0
In-house system help desk	Help Desk	0.00	$0	$0	0	$0
Commercial system admin (add'l)	Server Ops	0.00	$0	$0	0	$0
Commercial system support costs	Dev.	0.00	$0	$0	0	$0
Commercial system upgrade costs	Dev.	0.00	$0	$0	0	$0
TOTAL						**$2,080**

The scope of IP management tasks is very broad so the total cost of IPAM operations is substantial. If an IPAM system can help automate tasks and reduce costs, particularly in the high expense areas of IP address assignment, IP inventory assurance, auditing and reporting, it may prove worthy. Table 10-17 summarizes the cost areas described so far.

We've grouped our IP management tasks into five task groups: Block Moves/Adds/Changes, Subnet Moves/Adds/Changes, IP Address Moves/Adds/Changes, Management Functions, and Outage Recovery. For each function listed, the pair of columns listed under *Cost By Task* indicates the annual cost of each task and its relative cost contribution as a percentage of total cost. For example, IP Address Changes cost $40,073 annually, representing 4.8 percent of the total IP management costs.

The next pair of columns highlights cost and cost contribution *By Task Group*. Thus the *Subnet Moves/Adds/Changes* group totals $6,735 annually, which is comprised of the sum of Subnet Allocation, Subnet Deletion, and Subnet Changes tasks. Viewing your costs in this manner provides immediate

TABLE 10-17: IPAM Costs Basis Summary

IPAM Function	By Task		By Task Group	
	Annual Cost	% of Total Cost	Annual Cost	% of Total Cost
Block Moves/Adds/Changes			$24,670	2.9%
Block allocation	$23,800	2.8%		
Block deletion	$120	0.0%		
Block changes	$750	0.1%		
Subnet Moves/Adds/Changes			$6,735	0.8%
Subnet allocation	$3,895	0.5%		
Subnet deletion	$665	0.1%		
Subnet changes	$2,175	0.3%		
IP Address Moves/Adds/Changes			$143,198	17.0%
IP Address assignment	$84,375	10.0%		
IP Address deletes	$18,750	2.2%		
IP Address changes	$40,073	4.8%		
Management Function Costs			$519,623	61.7%
Inventory assurance	$202,752	24.1%		
Capacity management	$75,720	9.0%		
Auditing and reporting	$76,361	9.1%		
Server upgrades	$37,710	4.5%		
System administration	$2,080	0.2%		
IPv6 support costs	$125,000	14.8%		
Outage Recovery Costs			$147,777	17.6%
Configuration errors	$81,889	9.7%		
Pool depletions	$0	0.0%		
Server outages	$65,888	7.8%		
Total IPAM Costs	**$842,003**	100.0%	**$842,003**	100.0%

insight into what IP management functions are candidates for cost reduction. It's interesting to note that in our scenario, the traditional IP management functional costs for block, subnet, and IP address moves, adds and changes amounts to only 20.7 percent of IP management costs. Management functions are costing us dearly at 61.7 percent of the total, though these critical functions highlight the "management" discipline required for managing IP address space.

To reduce costs in this area, we should seek an IP management solution that provides strong inventory assurance, IP address assignment, and IPv6 functionality. In ranking individual tasks by cost, these three tasks comprise the top three and together account for over half of the total cost. Outage costs due to configuration errors and server outages as well as auditing and reporting are the next costliest tasks. Thus computing your costs can help prioritize feature sets that can bring the greatest cost reduction in your environment.

SAVINGS WITH IPAM DEPLOYMENT

Now that we've enumerated the major IP management tasks and an example set of costs, let's consider the upside of the business case as how much we can reduce these costs. In general, these costs will not be completely eliminated, as implementing an IP management tool will not necessarily drive these costs to zero. Some level of oversight, control, and therefore manual effort will still likely be required, though the objective is to minimize this effort, especially for your high cost functions.

The interesting though somewhat speculative part of the business case is determining how much lingering manual effort will be required on an ongoing basis. Depending on the level of automation of the IPAM tool you may be considering, the average time per resource can be vastly diminished by 30 to 80 percent. For modest networks, a minimal investment in a point solution may be warranted to reduce 40 percent of key expense areas for a modest investment. On the other hand, for larger networks, a more integrated, multi-function system may be worth the investment to reduce manual expenses by 75 percent or more. If you can evaluate a vendor's product, you can measure the relative time savings to perform certain tasks. Table 10-18 illustrates a post-IPAM implementation cost for the IP address assignment function. Contrast Table 10-18 with Table 10-7, which illustrates the analogous pre-IPAM implementation costs. You should consider the new time and cost in a post-IPAM environment for each of the tables described earlier.

Armed with your priority list of features, evaluate various tools' strengths and weaknesses relative to automating your high cost functions. Some IPAM systems for example can provide a level of automation with the IP address assignment task. Let's assume that a tool under evaluation enables automated DHCP and DNS configuration based on parameters entered during address assignment. The task still involves identifying an available IP address, but once selected, the tool can create a corresponding DNS record update.

Implementation of this system would eliminate the need for Server Operations to update DNS for manually configured devices. This saves us $11,250 per this corresponding row in Table 10-7. If the tool also automates Manual DHCP entry in DHCP servers, this automation can save an additional $7,500. We've already contributed $18,750 to cost savings!

In our case, saving $18,750 would amount to a 2.2% cost savings, but this is only the first of the many IP management tasks that should be analyzed. The other tasks we discussed in this chapter should be considered with respect to savings that could be realized based on automation provided by the IP management system in question.

Be diligent in considering every task. Certain tasks, such as system administration, may actually cost more after implementing an IP management solution. Add in the additional administration staff costs including the annual support costs from the vendor. Just as we arrived at $842,003 as our base cost, consider each task with the new system in mind, and ideally in hand. A hands-on

TABLE 10-18: Post IPAM Implementation IP Address Assignment Cost Analysis

IP Address Assignment	Resource Type	Average Staff-Time Required (Hours)	Resource Hourly Cost	Cost Per Event	Events Per Year	Annualized Cost
Identify available IP address	Engineering	0.05	$50	$3	7,500	$18,750
Update device configuration in DHCP server (M-DHCP)	Server Ops	0.00	$40	$0	1,875	$0
Static address assignment—email to owner then manual config	Engineering	0.10	$50	$13	5,625	$28,125
Update DNS for manually configured IP addresses	Server Ops	0.00	$40	$0	5,625	$0
Address assignment verification	Net Ops	0.05	$50	$3	7,500	$18,750
TOTAL						**$65,625**

evaluation can facilitate a more accurate estimate of how the system would integrate with and automate related tasks. Table 10-19 summarizes our analysis of IP management "System A" in terms of current method costs, shown under the *Current Annual Costs* column against *Annual Costs* when using System A and the corresponding *Savings*.

Notice that system administration costs jump due to additional administrative tasks as well as annual support and training costs that are required with the system. Nevertheless, the total net savings if System A is deployed is estimated to be about $239,369. This represents the annual savings or return should we invest in System A.

BUSINESS CASE EXPENSES

As we just calculated our savings with IP management System A, we also derived our residual annual expenses required to continue performing IP management tasks with the new system in place. Next consider the investment required to

TABLE 10-19: Example IPAM Savings Analysis

IPAM Function	Current	Costs with IPAM System A	
	Annual Costs	Annual Costs	Savings
Block Moves/Adds/Changes			
Block allocation	$23,800	$9,400	$14,400
Block deletion	$120	$120	$0
Block changes	$750	$637	$114
Subnet Moves/Adds/Changes			
Subnet allocation	$3,895	$2,052	$1,843
Subnet deletion	$665	$665	$0
Subnet changes	$2,175	$938	$1,238
IP Address Moves/Adds/Changes			
IP address Assignment	$84,375	$65,625	$18,750
IP address Deletes	$18,750	$18,750	$0
IP address Changes	$40,073	$29,761	$10,313
Management Function Costs			
Inventory assurance	$202,752	$78,992	$123,760
Capacity management	$75,720	$42,024	$33,696
Auditing and reporting	$76,361	$76,361	$0
Server upgrades	$37,710	$37,710	$0
System administration	$2,080	$89,160	($87,080)
IPv6 support costs	$125,000	$70,000	$55,000
Outage Recovery Costs			
Configuration Errors	$81,889	$15,704	$66,185
Pool Depletions	$0	$0	$0
Server Outages	$65,888	$64,736	$1,152
Total IPAM Costs	**$842,003**	**$602,634**	**$239,369**

implement System A, including licensing, hardware, and implementation and training costs. Any vendor you consider for IP management will provide you a price quote on request for these expense items. Be sure to clarify whether new feature upgrades are included with support or cost an additional fee. This additional cost, if any, needs to be added to the expense side of the equation. As mentioned above, each tool is quite different in terms of feature set, so be sure to align the income (savings) side of the equation with the cost and features provided by the tool you're comparing on the expense side. After all, while using spreadsheets and manual DNS and DHCP configuration costs nothing in capital, the staffing costs to support required IP management tasks total over $842,000!

Continuing with our example, we determined we could reduce our net IP management task costs by $239,369 to $602,634 annually by deploying System A. Let's assume System A costs about $200,000, with implementation costs of

$30,000. Recall that we already accounted for the annual support costs for System A per the increased system administration costs per Table 10-19. Our first year investment then amounts to $230,000. The next step is to calculate the return on this investment.

NETTING IT OUT: BUSINESS CASE RESULTS

Table 10-20 nets out our "cash flows," which contribute to providing a return on the $230,000 investment. In year 1, our return is already slightly less than our investment, yielding a payback within one year. The payback period is calculated by dividing our investment into our cash flow per unit time. For example, IPAM Worldwide's investment ($230,000) divided by savings ($239,369 year) results in a payback of about a year, as shown in the following equation.

$$Payback\ Period = \frac{Investment}{Savings\ per\ unit\ time} = \frac{\$230{,}000}{\$239{,}369\ per\ year} = 1.0\ years$$

To calculate the payback period in months, we simply substitute our monthly savings in the denominator of our calculation:

$$Payback\ Period = \frac{Investment}{Savings\ per\ unit\ time} = \frac{\$230{,}000}{\left(\frac{\$239{,}369}{12}\right)\ per\ month} = 11.5\ months$$

This relatively rapid payback may make this investment decision quite simple. The other metric to consider is of course the return on investment (ROI) value itself, which is simply the cumulative savings over a given time span, typically three years, less the initial investment, divided by the investment amount.

TABLE 10-20: IPAM Business Case Results

	Year 1	Year 2	Year 3
IP management costs w/o IPAM	$842,003	$842,003	$842,003
IP management costs w/IPAM	–$602,634	–$602,634	–$602,634
Cost savings with IPAM (return)	$239,369	$239,369	$239,369
Cumulative Return	*$239,969*	*$478,739*	*$718,108*
IP management system investment	$200,000		
IP management system installation	$30,000		
Total cost (investment)	$230,000		
ROI (%)	212%		
ROI NPV @ 15%	166%		
Payback period (years)	1.0		
Payback period (months)	11.5		

$$ROI = \frac{Savings - Investment}{Investment} = \frac{(\$718{,}108 - \$230{,}000)}{\$230{,}0000} = 2.12 \times 100\% = 212\%$$

Our three year savings total over $718,000. The net return is derived by subtracting the initial investment, $718,000 – $230,000 = $488,000. This net return, divided by our $230,000 investment is 2.12 or 212%. This means that our net return has outpaced our investment by a ratio of 2.1:1 over three years. We can divide this result by 3 [years] to define our annualized ROI: 71 percent per year. Many organizations consider the time value of money and apply a discount rate to account for tomorrow's dollars being worth less than today's. Using a 15 percent rate, our three-year ROI is 166 percent. Different organizations have different investment criteria, so be cognizant of your standard policies for decision making.

CONCLUSION

A rigorous business case analysis can help you financially justify investment in an IPAM product. Quantifying your expenses under your current IPAM methodology using these techniques can also help you identify high cost areas. Knowledge of your high cost functions enables you to focus your search for an IPAM solution to one that can lower the cost of these functions, thereby maximizing your return on IPAM system investment.

Appendix A

IPv4 DHCP OPTIONS

Table A-1 lists the current set of defined IPv4 DHCP options. The Code column indicates the option code or number, and the name column lists the corresponding option name. Note that the Len (Length) column indicates the value of the Length field within the option. The total option length is this value plus two bytes, one byte for the code and one for the length field itself.

TABLE A-1: DHCP Options (Bootp Vendor Extensions and DHCP Options. [Online] www.iana.org/assignments/bootp-dhcp-parameters)

Code	Name	Len	Meaning	Reference
0	Pad	0	None	RFC 2132 (50)
1	Subnet Mask	4	Subnet Mask in "IP address" format	RFC 2132 (50)
2	Time Offset	4	Time Offset in Seconds from UTC (Deprecated by RFC 4833 which specifies use of options 100 & 101)	RFC 2132 (50)
3	Router	N	N/4* Router (default gateway) addresses	RFC 2132 (50)

Introduction to IP Address Management, By Timothy Rooney

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
4	Time Server	N	N/4 Timeserver addresses	RFC 2132 (50)
5	Name Server	N	N/4 IEN-116** name server addresses	RFC 2132 (50)
6	Domain Server	N	N/4 DNS server addresses	RFC 2132 (50)
7	Log Server	N	N/4 MIT Laboratory for Computer Science (LCS) UDP log server addresses	RFC 2132 (50)
8	Quotes Server	N	N/4 "Quote of the day" server addresses	RFC 2132 (50)
9	LPR Server	N	N/4 Line Printer server addresses	RFC 2132 (50)
10	Impress Server	N	N/4 Imagen Impress server addresses	RFC 2132 (50)
11	RLP Server	N	N/4 Resource Location Protocol server addresses	RFC 2132 (50)
12	Hostname	N	Client hostname string	RFC 2132 (50)
13	Boot File Size	2	Size of boot file in 512 byte blocks	RFC 2132 (50)
14	Merit Dump File	N	File pathname to which the client should dump its core image in the event of a client crash	RFC 2132 (50)
15	Domain Name	N	The DNS domain name of the client	RFC 2132 (50)
16	Swap Server	N	Swap server address	RFC 2132 (50)
17	Root Path	N	Path name for the client's root disk	RFC 2132 (50)
18	Extension File	N	Path name of a file containing vendor-extension information retrievable via TFTP	RFC 2132 (50)
19	Forward On/Off	1	Enable/Disable IP packet forwarding	RFC 2132 (50)
20	Source Routing On/Off	1	Enable/Disable IP packet forwarding for packets specifying nonlocal source routes	RFC 2132 (50)
21	Policy Filter	N	Specifies acceptable non-local next hops to which IP packets may be forwarded for packets specifying nonlocal source routes	RFC 2132 (50)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
22	Max Datagram Reassembly Size	2	The maximum size datagram the client should be ready to reassemble specified as a 16-bit unsigned integer	RFC 2132 (50)
23	Default IP TTL	1	Default IP time to live value for use in outgoing packets' IP header TTL field	RFC 2132 (50)
24	Path MTU Aging Timeout	4	The timeout in seconds when performing path maximum transmission unit (MTU) discovery in accordance with RFC 1191; MTU discovery helps minimize packet fragmentation along the path	RFC 2132 (50)
25	Path MTU Plateau Table	N	A table listing maximum transmission unit (MTU) sizes to use when performing path MTU discovery per RFC 1191	RFC 2132 (50)
26	Interface MTU	2	The value of the maximum transmission unit (MTU) for this device interface	RFC 2132 (50)
27	All Subnets are Local	1	Indicates whether all subnets within the client's network use the same maximum transmission unit (MTU) as the local subnet to which the client is connected	RFC 2132 (50)
28	Broadcast Address	4	Specifies the broadcast IP address for the client's subnet	RFC 2132 (50)
29	Mask Discovery	1	Specifies whether the client should perform subnet mask discovery or not	RFC 2132 (50)
30	Mask Supplier	1	Specifies whether the client should respond to other clients performing mask discovery	RFC 2132 (50)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
31	Router Discovery	1	Specifies whether the client should perform router discovery or not	RFC 2132 (50)
32	Router Solicitation Address	4	Specifies the IP address to which the client should direct router solicitation requests	RFC 2132 (50)
33	Static Route	N	Specifies a set of static routes the client should install in its routing cache; listed as "destination network—next hop router" pairings (Obsoleted by RFC 3442 defining the Classless static route option, 121).	RFC 2132 RFC 3442 (51)
34	Trailer Encapsulation	1	Specifies whether the client should attempt to negotiate the use of layer 2 frame trailers (like headers but at the end of the frame payload) in ARP messages	RFC 2132 (50)
35	ARP Timeout	4	ARP cache timeout in seconds	RFC 2132 (50)
36	Ethernet Encapsulation	1	Specifies whether the client should use Ethernet II or IEEE 802.3 on an Ethernet interface	RFC 2132 (50)
37	Default TCP TTL	1	Default TCP time to live value	RFC 2132 (50)
38	TCP Keepalive Time	4	TCP keepalive interval in seconds	RFC 2132 (50)
39	TCP Keepalive Garbage	1	Specifies whether the client should send an octet of "garbage" within TCP keepalive messages for compatibility with older implementations	RFC 2132 (50)
40	NIS Domain	N	Network Information Services domain	RFC 2132 (50)
41	NIS Servers	N	N/4 Network Information Services server addresses	RFC 2132 (50)
42	NTP Servers	N	N/4 Network Time Protocol server addresses	RFC 2132 (50)
43	Vendor Specific	N	Vendor specific information	RFC 2132 (50)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
44	NETBIOS Name Server	N	N/4 NETBIOS Name server (aka WINS server) addresses	RFC 2132 (50)
45	NBDD Server	N	N/4 NETBIOS Datagram Distribution (NBDD) server addresses	RFC 2132 (50)
46	NETBIOS Node Type	1	Specifies the client as a specific NETBIOS Node Type	RFC 2132 (50)
47	NETBIOS Scope	N	Specifies the NETBIOS scope for the client	RFC 2132 (50)
48	X Window Font Server	N	N/4 X Window Font server addresses	RFC 2132 (50)
49	X Window Display Manager	N	N/4 X Window display manager addresses	RFC 2132 (50)
50	Address Request	4	IP address requested by the client (within a Discover message)	RFC 2132 (50)
51	Address Time	4	IP address lease time requested by the client (within a Discover or Request message)	RFC 2132 (50)
52	Option Overload	1	Indicates that the "sname" and/or "file" DHCP header fields contain additional DHCP option information if options to return to the client exceed the normal option space in the message	RFC 2132 (50)
53	DHCP Message Type	1	DHCP message type as we discussed in Chapter 3 (Discover, Offer, etc.)	RFC 2132 (50)
54	DHCP Server Identifier	4	DHCP server identification provided in the Offer (and Request and optionally ACK, NAK) to identify the server, e.g., to distinguish among multiple offers	RFC 2132 (50)
55	Parameter List	N	List of DHCP option code numbers for parameters requested by the client	RFC 2132 (50)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
56	DHCP Error Message Text	N	Text containing an error message; can be used by the server in a Nak message to the client or by the client in a Decline message; e.g., this text could be included in logging details	RFC 2132 (50)
57	Maximum DHCP Message Size	2	The maximum DHCP message length the client is willing to accept	RFC 2132 (50)
58	Renewal (T1) Time	4	Interval from address assignment time to the time the client enters the Renewing state	RFC 2132 (50)
59	Rebinding (T2) Time	4	Interval from address assignment time to the time the client enters the Rebinding state	RFC 2132 (50)
60	Vendor Class Identifier	N	Used by clients to specify a vendor-specific identifier	RFC 2132 (50)
61	Client Id	N	Client Identifier	RFC 2132 (50)
62	Netware/IP Domain	N	Netware/IP Domain Name	RFC 2242 (52)
63	Netware/IP Option	N	Netware/IP sub Options	RFC 2242 (52)
64	NIS+ Domain	N	Network Information Services+ (NIS+) client domain name	RFC 2132 (50)
65	NIS+ Servers	N	N/4 Network Information Services+ (NIS+) server addresses	RFC 2132 (50)
66	TFTP Server Name	N	TFTP server name; can be used when the “sname” DHCP header field has been overloaded with other options	RFC 2132 (50)
67	Bootfile Name	N	Boot file name; can be used when the “file” DHCP header field has been overloaded with other options	RFC 2132 (50)
68	Home Agent	N	N/4 Mobile IP home agent addresses	RFC 2132 (50)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
69	SMTP Server	N	N/4 Simple Mail Transfer Protocol (SMTP) server addresses for outgoing e-mail	RFC 2132 (50)
70	POP3 Server	N	N/4 Post Office Protocol v3 (POP3) server addresses for incoming e-mail retrieval	RFC 2132 (50)
71	NNTP Server	N	N/4 Network News Transport Protocol (NNTP) server addresses	RFC 2132 (50)
72	WWW Server	N	N/4 World Wide Web (WWW) server addresses	RFC 2132 (50)
73	Finger Server	N	N/4 Finger server addresses; finger servers enable retrieval of host user information regarding login name, login duration, and more	RFC 2132 (50)
74	IRC Server	N	N/4 Internet Relay Chat (IRC) server addresses	RFC 2132 (50)
75	StreetTalk Server	N	N/4 StreetTalk server addresses; StreetTalk was a Banyan Vines user and resource directory	RFC 2132 (50)
76	STDA Server	N	N/4 StreetTalk Directory Assistance (STDA) server addresses; StreetTalk was a Banyan Vines user and resource directory	RFC 2132 (50)
77	User-Class	N	User Class Identifier	RFC 3004 (53)
78	SLP Directory Agent	N + 1	N/4 Service Location Protocol (SLP) Directory Agent IP address(es)	RFC 2610 (54)
79	SLP Service Scope	N	Service Location Protocol (SLP) service scope the SLP agent is configured to use	RFC 2610 (54)
80	Rapid Commit	0	Rapid Commit—requests a two-packet DHCP transaction instead of the normal four packet DORA process for mobility or overhead-constrained applications	RFC 4039 (55)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
81	Client FQDN	N	Fully Qualified Domain Name—defines the client's FQDN and whether the client or DHCP server should update DNS	RFC 4702 (56)
82	Relay Agent Information	N	Relay Agent Information—additional client information supplied by the intervening relay agent	RFC 3046 (57)
83	Internet Storage Name Service	N	Internet Storage Name Service (iSNS) server addresses and iSNS application information	RFC 4174 (58)
84	Unassigned	—	—	RFC 3679 (59)
85	NDS Servers	N	N/4 Novell Directory Services (NDS) Server IP addresses to contact for NDS client authentication and access the NDS directory repository	RFC 2241 (60)
86	NDS Tree Name	N	Novell Directory Services (NDS) tree name of the NDS repository the client should contact	RFC 2241 (60)
87	NDS Context	N	Novell Directory Services (NDS) initial context within the NDS repository the NDS client should use	RFC 2241 (60)
88	BCMCS Controller Domain Name	N	Broadcast and Multicast Server domain name (FQDN) list, used to construct follow-up SRV query(ies) (BCMCS is used in 3G wireless networks to enable mobiles to receive broadcast and multicast services)	RFC 4280 (61)
89	BCMCS Controller IPv4 address	N	N/4 Broadcast and Multicast Server (BCMCS) Controller IP address(es) (BCMCS is used in 3G wireless networks to enable mobiles to receive broadcast and multicast services)	RFC 4280 (61)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
90	Authentication	N	Authentication option used to communicate authentication information between the client and server in accordance with the DHCP authentication protocol	RFC3118 (43)
91	Client-last-transaction-time option	4	Seconds since the last DHCP transaction with the client on this lease as queried in a DHCP Lease Query message	RFC 4388 (62)
92	Associated-ip option	N	List of IP addresses associated with the client as queried in a DHCP Lease Query message	RFC 4388 (62)
93	PXE Client System	N	PXE client system architecture type(s) each encoded as 16-bit code, e.g., Intel x86PC, DEC Alpha, EFI x86-64, etc.	RFC 4578 (25)
94	PXE Client Network Interface	3	PXE client network interface identifier with individual octets encoded for interface type, interface major version number, and interface minor version number	RFC 4578 (25)
95	LDAP	N	Lightweight Directory Access Protocol servers; this option is used by Apple Computer though no governing RFC has been published	RFC 3679 (59)
96	Unassigned	—	—	RFC 3679 (59)
97	PXE Client Machine Identifier	N	PXE client machine identifier with encoded type and identifier value	RFC 4578 (25)
98	User Authentication Protocol	N	List of locations (URLs) for services capable of processing authentication requests encapsulated using Open Group's User Authentication Protocol (UAP)	RFC 2485 (63)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
99	Civic Location	—	Location of the server, network element closest to the client or the client itself as provided by the server encoded in country-specific civic (e.g., postal) format	RFC 4776 (64)
100	Time Zone	N	Time Zone encoded as IEEE 1003.1 TZ (POSIX)	RFC 4833 (65)
101	Time Zone database	N	Reference to a local (on the client) TZ database for lookup of time zone	RFC 4833 (65)
102–111	Unassigned	—	—	RFC 3679 (59)
112	Netinfo Address	N	NetInfo Parent Server Address; this option is used by Apple Computer though no governing RFC has been published; NetInfo is a distributed database of user and resource information for Apple devices.	RFC 3679 (59)
113	Netinfo Tag	N	NetInfo Parent Server Tag; this option is used by Apple Computer though no governing RFC has been published. NetInfo is a distributed database of user and resource information for Apple devices.	RFC 3679 (59)
114	URL	N	Uniform Resource Locator; this option is used by Apple Computer though no governing RFC has been published	RFC 3679 (59)
115	Unassigned	—	—	RFC 3679 (59)
116	Auto-Configure	1	Instructs the client to auto-configure a link local address (69.254.0.0/16) or not. This can be used by the DHCP server to inform the client that it has no IP addresses to assign and that the client may or may not auto-configure	RFC 2563 (66)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
117	Name Service Search	N	Lists one or more name services in priority order that the client should use for name resolution: DNS, NIS, NIS+, or WINS	RFC 2937 (67)
118	Subnet Selection	4	Identifies an IP subnet (address) from which to allocate an IP address to this client—overrides the GIAddr setting or DHCP server interface on which a broadcast Discover was received	RFC 3011 (68)
119	Domain Search	N	List of one or more domains for configuration of the client's resolver. If the application requests a resolution for a non-FQDN hostname, these domain(s) will successively be appended to the hostname prior to querying	RFC 3397 (69)
120	SIP Servers	N	A listing of one or more of either Session Initiation Protocol (SIP) server FQDN(s) or of SIP server IP address(es). SIP is a control protocol for management of multi-media calls or sessions.	RFC 3361 (70)
121	Classless Static Route	N	Specifies a set of static routes the client should install in its routing cache; listed as "<CIDR mask length>.<destination network>—next hop router" pairings. The destination network is enumerated only to significant octets, dropping local (non-subnet) portions; for example, 172.16.0.0/12 would be encoded as 12.172.16 and 10.0.0.0/18 as 18.10.0.0	RFC 3442 (51)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
122	CableLabs Client Configuration	N	Specifies resource (e.g., provisioning server, DHCP server, etc.) locations and parameters for use by cable multimedia terminal adapters (MTAs), which are customer premises devices operating over a DOCSIS cable network, providing VoIP and related multimedia services	RFC 3495 (71)
123	Location Configuration Information	16	Provides the client its Location Configuration Information (LCI), including latitude, longitude, altitude and resolution of each coordinate	RFC 3825 (72)
124	Vendor-Identifying Vendor Class	N	Enables specification of multiple vendor classes, each identified by IANA-assigned Enterprise Number (EN); this is useful to identify the hardware vendor, software vendor, application vendor, etc. supporting the device	RFC 3925 (73)
125	Vendor-Identifying Vendor-Specific Information	N	Set of DHCP options grouped by vendor as identified by IANA-assigned Enterprise Number (EN);	RFC 3925 (73)
126–127	Unassigned	—	—	RFC 3679 (59)
128	PXE—undefined (vendor specific)			RFC 4578 (25)
Over-loaded	Etherboot signature. 6 bytes: E4:45:74:68:00:00 DOCSIS “full security” server IP address TFTP Server IP address (for IP Phone software load)			
129	PXE—undefined (vendor specific)			RFC 4578 (25)
Over-loaded	Kernel options. Variable length string Call Server IP address			
130	PXE—undefined (vendor specific)			RFC 4578 (25)
Over-loaded	Ethernet interface. Variable length string. Discrimination string (to identify vendor)			
131	PXE—undefined (vendor specific)			RFC 4578 (25)
Over-loaded	Remote statistics server IP address			

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
132	PXE—undefined (vendor specific)			RFC 4578 (25)
Over-loaded	802.1P VLAN ID			
133	PXE—undefined (vendor specific)			RFC 4578 (25)
Over-loaded	802.1Q L2 Priority			
134	PXE—undefined (vendor specific)			RFC 4578 (25)
Over-loaded	Diffserv Code Point			
135	PXE—undefined (vendor specific)			RFC 4578 (25)
Over-loaded	HTTP Proxy for phone-specific applications			
136	PANA Agent	N	Identifies one or more IPv4 addresses of PANA (Protocol for carrying Authentication for Network Access) Authentication Agents for use by the client for authentication and authorization for network access service	RFC 5192 (74)
137	OPTION_V4_LOST	N	Location to Service Translation (LoST) server domain name; LoST protocol maps service identifiers and location information to service URLs	RFC 5223 (75)
138	CAPWAP Access Controller	N	Control and Provisioning of Wireless Access Points (CAPWAP) Access Controller IP address(es) to which the client may connect	RFC 5417 (76)
139	MoS Service IP addresses	N	IPv4 address(es) for servers providing particular types of IEEE 802.21 Mobility Service (MoS)	RFC 5678 (77)
140	MoS Service FQDNs	N	FQDN(s) for servers providing particular types of IEEE 802.21 Mobility Service (MoS)	RFC 5678 (77)
141–149	Unassigned			RFC 3942 (78)
150	TFTP server address (tentatively assigned—23 Jun 2005) Etherboot GRUB configuration path name			
151–174	Unassigned			RFC 3942 (78)

TABLE A-1: *Continued*

Code	Name	Len	Meaning	Reference
175	Etherboot (tentatively assigned—23 Jun 2005)			
176	IP Telephone (tentatively assigned—23 Jun 2005)			
177	Etherboot (tentatively assigned—23 Jun 2005)			
178–207	Unassigned			RFC 3942 (78)
208	PXE Magic (deprecated)	4	F1:00:74:7E	RFC 5071 (79)
209	PXE Configuration File	N	Configuration filename or file pathname for second stage PXE boot loading	RFC 5071 (79)
210	PXE Path Prefix	N	Configuration file path prefix to the filename specified in the PXE configuration file option (209)	RFC 5071 (79)
211	PXE Reboot Time	4	Number of seconds to wait to reboot if TFTP server is unreachable	RFC 5071 (79)
212-219	Unassigned			
220	Subnet Allocation Option (tentatively assigned—23 Jun 2005)			
221	Virtual Subnet Selection Option (tentatively assigned—23 Jun 2005)			
222-223	Unassigned			RFC 3942 (78)
224-254	Reserved (Private Use)			
255	End	0	None	RFC 2132 (50)

*The *N/4* notation refers to the use of "N" bytes to represent one or more IPv4 addresses, each of which is comprised of four bytes; thus for a length of N, the field would contain N/4 complete IPv4 addresses. This implies of course that N is a multiple of 4 in cases where the data type is IP address.
**IEN-116 = Internet Experiment Note 116; IENs were eventually merged with RFCs as TCP/IP went into production across ARPANET.

Appendix B

DHCPv6 OPTIONS

DHCPv6 options are used to convey information relevant to the associated DHCP message, including DUIDs and IAs. Options are listed within the DHCPv6 message and have the following general format shown in Figure B-1.

The currently defined set of DHCPv6 options are defined in Table B-1. Note that certain options may be nested, such as those associated with an IA.

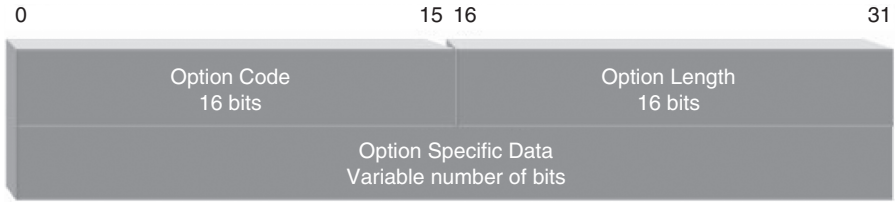

Figure B-1: DHCPv6 options format

Introduction to IP Address Management, By Timothy Rooney

TABLE B-1: DHCPv6 Options (Dynamic Host Configuration Protocol for IPv6 (DHCPv6). [Online] www.iana.org/assignments/dhcpv6-parameters)

Code	Name	Meaning	Reference
1	OPTION_ CLIENTID	Client Identifier (DUID of client)	RFC 3315 (21)
2	OPTION_ SERVERID	Server Identifier (DUID of server)	RFC 3315 (21)
3	OPTION_IA_NA	Identity Association for Non-temporary addresses—includes the IAID, T1 time, T2 time and additional options for the IA for non-temporary addresses	RFC 3315 (21)
4	OPTION_IA_TA	Identity Association for Temporary addresses—includes the IAID and additional options for this IA for temporary addresses	RFC 3315 (21)
5	OPTION_ IAADDR	IA Address option—specifies IPv6 addresses and associated preferred lifetime, valid lifetime and options associated with an IA_NA or IA_ TA. As such, this option may only appear as an option to the DHCPv6 message option OPTION_IA_TA or OPTION_IA_ NA.	RFC 3315 (21)
6	OPTION_ORO	Option Request Option—used by clients to list option codes for which values are requested or by servers in a Reconfigure message to indicate which options the client should request in its subsequent Renew or Information-Request message	RFC 3315 (21)
7	OPTION_ PREFERENCE	Preference setting by the server to facilitate client selection of DHCP server	RFC 3315 (21)
8	OPTION_ ELAPSED_TIME	The amount of time since the client began the current DHCP transaction in hundredths of a second. Clients are required to use this option	RFC 3315 (21)
9	OPTION_RELAY_ MSG	The DHCP message being relayed by a relay agent	RFC 3315 (21)
10	Unassigned		
11	OPTION_AUTH	Authentication information for use in reliably identifying the source of a DHCP message and to verify message integrity	RFC 3315 (21)

TABLE B-1: *Continued*

Code	Name	Meaning	Reference
12	OPTION_ UNICAST	Server unicast option indicates the IP address to which the client may unicast messages to this server	RFC 3315 (21)
13	OPTION_ STATUS_CODE	Status code option indicates a 2-byte status code and variable length status message. This option may be used as a DHCP message option or as an option within another DHCP message option.	RFC3315 (21)
14	OPTION_RAPID_ COMMIT	Rapid commit option enables a client to request a direct reply with an IP address and parameters, bypassing the Advertise and Request messages.	RFC3315 (21)
15	OPTION_USER_ CLASS	User class option—analogous to user class in DHCPv4 in assisting the server in making address assignment decisions	RFC 3315 (21)
16	OPTION_ VENDOR_ CLASS	Vendor class option—analogous to vendor class in DHCPv4 in conveying the vendor or manufacturer of the device or interface to assist the server in making address assignment decisions. The vendor class option includes the IANA-assigned Enterprise Number for the vendor	RFC 3315 (21)
17	OPTION_ VENDOR_OPTS	Vendor specific information—this option includes the IANA-assigned Enterprise Number as well as one or more options, each defined with option code, length and value	RFC 3315 (21)
18	OPTION_ INTERFACE_ID	Interface ID option—used by relay agents to convey the agent's interface ID on which the client message was received. This option may only appear in RELAY-FORW messages, and when it does, it is copied by the server to the RELAYREPL message	RFC 3315 (21)
19	OPTION_ RECONF_MSG	Reconfigure Message option for use in the Reconfigure message to inform the client which message to use to reconfigure; either Renew or Information-Request	RFC 3315 (21)

TABLE B-1: *Continued*

Code	Name	Meaning	Reference
20	OPTION_ RECONF_ ACCEPT	Reconfigure Accept option—the client populates this option if it is willing to accept Reconfigure messages from the server	RFC 3315 (21)
21	OPTION_SIP_ SERVER_D	SIP Servers Domain Names option listing domain names of the SIP outbound proxy servers that the client can use	RFC 3319 (80)
22	OPTION_SIP_ SERVER_A	SIP Servers IPv6 Address List option lists the IPv6 addresses of the SIP outbound proxy servers that the client can use	RFC 3319 (80)
23	OPTION_DNS_ SERVERS	DNS Recursive Name Server Option—lists IPv6 address(es) of DNS recursive name servers to which DNS queries may be sent by the client resolver in order of preference	RFC 3646 (81)
24	OPTION_ DOMAIN_LIST	Domain Search List option—provides a domain search list for client use when resolving hostnames via DNS	RFC 3646 (81)
25	OPTION_IA_PD	Identity Association for Prefix Delegation—includes the IAID, T1 time, T2 time and additional options for the IA_PD, including the associated prefix(es) defined as option code 26.	RFC 3633 (82)
26	OPTION_ IAPREFIX	IA_PD Prefix option—specifies the IPv6 prefixes associated with the IA_PD, along with associated options and preferred and valid lifetimes. This option may only appear as an option to the DHCPv6 message option OPTION_IA_PD. The prefix is specified with an 8-bit prefix length and a 128-bit IPv6 prefix.	RFC 3633 (82)
27	OPTION_NIS_ SERVERS	Network Information Service (NIS) Servers—ordered list of NIS servers by IPv6 address available to the client	RFC 3898 (83)
28	OPTION_NISP_ SERVERS	Network Information Service v2 (NIS+) Servers—ordered list of NIS+ servers by IPv6 address available to the client	RFC 3898 (83)

TABLE B-1: *Continued*

Code	Name	Meaning	Reference
29	OPTION_NIS_ DOMAIN_ NAME	Network Information Service (NIS) domain name—NIS domain name to be used by the client	RFC 3898 (83)
30	OPTION_NISP_ DOMAIN_ NAME	Network Information Service v2 (NIS+) domain name—NIS+ domain name to be used by the client.	RFC 3898 (83)
31	OPTION_SNTP_ SERVERS	Simple Network Time Protocol (SNTP) servers—ordered list of SNTP servers by IPv6 address available to the client	RFC 4075 (84)
32	OPTION_ INFORMATION_ REFRESH_TIME	Information Refresh Option—specifies the upper bound of the number or seconds from the current time that a client should wait before refreshing information received from the DHCPv6 server, particularly for stateless DHCPv6 scenarios.	RFC 4242 (85)
33	OPTION_BCMCS_ SERVER_D	Broadcast and Multicast Service (BCMCS) Domain Name List—list of one or more FQDNs corresponding to BCMCS server(s). (BCMCS is used in 3G wireless networks to enable mobiles to receive broadcast and multicast services).	RFC 4280 (61)
34	OPTION_BCMCS_ SERVER_A	Broadcast and Multicast Service (BCMCS) IPv6 Address List—list of one or more IPv6 address(es) corresponding to BCMCS server(s). (BCMCS is used in 3G wireless networks to enable mobiles to receive broadcast and multicast services).	RFC 4280 (61)
35	Unassigned		
36	OPTION_ GEOCONF_ CIVIC	Geographical location in civic (e.g., postal) format. This option can be provided by the server to relate the location of the server, the closest network element (e.g., router) to the client or the client itself. The location information includes an ISO 3166 country code (US, DE, JP, etc.) and country-specific location information such as state, province, county, city, block, group of streets and more.	RFC 4776 (64)

TABLE B-1: *Continued*

Code	Name	Meaning	Reference
37	OPTION_ REMOTE_ID	Relay Agent Remote ID option—remote identity inserted by the relay agent in RELAY-FORW message to the DHCPv6 server. This is useful in service provider environments where the "edge" device facing the subscriber device, inserts an identifier for the subscriber connection prior to relaying to the DHCPv6 server	RFC 4649 (86)
38	OPTION_ SUBSCRIBER_ ID	Relay Agent Subscriber ID option—subscriber identity inserted by the relay agent in RELAY-FORW message to the DHCPv6 server. This is useful in service provider environments where the "edge" device facing the subscriber device, inserts an identifier for the subscriber from which the message originated, prior to relaying to the DHCPv6 server	RFC 4580 (87)
39	OPTION_ CLIENT_FQDN	FQDN option—indicates whether the client or the DHCP server should update DNS with the AAAA record corresponding to the assigned IPv6 address and the FQDN provided in this option. The DHCP server always updates the PTR record.	RFC 4704 (88)
40	OPTION_PANA_ AGENT	This option provides one or more IPv6 address(es) associated with PANA (Protocol for carrying Authentication for Network Access) Authentication Agents that a client can use.	RFC 5192 (74)
41	OPTION_NEW_ POSIX_ TIMEZONE	Time zone to be used by the client in IEEE 1003.1 format (POSIX—portable operating system interface). This format enables textual representation of time zone and daylight savings time information.	RFC 4833 (65)

TABLE B-1: *Continued*

Code	Name	Meaning	Reference
42	OPTION_NEW_TZDB_TIMEZONE	Time zone (TZ) database entry referred to by entry name. The client must have a copy of the TZ database, which it queries for the corresponding entry to determine its time zone	RFC 4833 (65)
43	OPTION_ERO	Relay Agent Echo Request option—used by relay agents in the RELAY_FORW message to request that the DHCPv6 sever echo back certain requested relay agent options, even if not supported on the server. (DHCPv4 servers always echo back relay agent option (82) information but this is not required in DHCPv6, hence this option for relay agents requiring such echo back).	RFC 4994 (89)
44	OPTION_LQ_QUERY	The Query option is used in the LEASEQUERY message to identify the query information being requested. This option includes the Query type (by IA address or client ID option), link address to which the query applies and query options.	RFC 5007 (90)
45	OPTION_CLIENT_DATA	Client Data—this option contains the query response information for the requested client data within a LEASEQUERY-REPLY message. At a minimum this option includes the client identifier (OPTION_CLIENTID), the IA address or prefix (OPTION_IAADDR and/or OPTION_IAPREFIX) and client last transaction time (OPTION_CLT_TIME)	RFC 5007 (90)
46	OPTION_CLT_TIME	Client Last Transaction Time—indicates the number of seconds since the server last communicated with the client referenced by the lease query. This option is encapsulated within the OPTION_CLIENT_DATA option within a LEASEQUERY-REPLY message.	RFC 5007 (90)

TABLE B-1: *Continued*

Code	Name	Meaning	Reference
47	OPTION_LQ_ RELAY_ DATA	Relay data—used in a LEASEQUERY-REPLY message to provide the relay agent information associated with the client information requested. This option includes the relay agent address from which the client's relay information was received along with the complete relayed message	RFC 5007 (90)
48	OPTION_LQ_ CLIENT_ LINK	Client link—identifies one or more links on which the queried client has DHCPv6 bindings. The queried client can be identified by address or client ID	RFC 5007 (90)
49	OPTION_MIP6_ HNINF	Mobile IPv6 Home Network Information—used by the client to identify its target home network to the server (in an Information Request message)	draft-ietf-mip6-hiopt-17.txt (91)
50	OPTION_MIP6_ RELAY	Mobile IPv6 Relay Agent—used by a relay agent to identify home network information via a Relay-Forw message)	draft-ietf-mip6-hiopt-17.txt (91)
51	OPTION_V6_ LOST	Location to Service Translation (LoST) server domain name; LoST protocol maps service identifiers and location information to service URLs	RFC 5223 (75)
52	OPTION_ CAPWAP_AC_ V6	Control and Provisioning of Wireless Access Points (CAPWAP) Access Controller IPv6 address(es) to which the client may connect	RFC 5417 (76)
53	OPTION_RELAY_ ID	DHCPv6 Bulk LeaseQuery—requests lease and prefix delegation bindings for a specified relay agent identified by its DUID in this option	RFC 5460 (92)
54	OPTION-IPv6_ Addresss-MoS	List of IPv6 address(es) for servers providing particular types of IEEE 802.21 Mobility Service (MoS)	RFC 5678 (77)
55	OPTION-IPv6_ FQDN-MoS	List of FQDN(s) for servers providing particular types of IEEE 802.21 Mobility Service (MoS)	RFC 5678 (77)
56	OPTION_NTP_ SERVER	Network Time Protocol (NTP) or Simple NTP (SNTP) Server Location as unicast or multicast address(es) or as FQDN	draft-ietf-ntp-dhcpv6-ntp-opt-06 (122)
57–255	Unassigned		

Appendix C

DNS RESOURCE RECORD SUMMARY

TABLE C-1: Resource Record and Query Type Summary (Domain Name System (DNS) Parameters. [Online] www.iana.org/assignments/dns-parameters)

RRType (or QType)	RR Purpose (i.e., Rdata contents)	RR Type ID	IETF Status	Defining Document
A	IPv4 address for a given hostname	1	Standard	RFC 1035 (23)
AAAA	IPv6 address for a given hostname	28	Draft Standard	RFC 3596 (93)
A6	IPv6 address or portion thereof for iterative IPv6 address resolution for a given hostname	38	Experimental	RFC 2874 (94)
AFSDB	Server hostname for a given AFS and DCE domain	18	Experimental	RFC 1183 (95)
APL	Address prefix lists for a given domain	42	Experimental	RFC 3123 (96)

Introduction to IP Address Management, By Timothy Rooney

TABLE C-1: *Continued*

RRType (or QType)	RR Purpose (i.e., Rdata contents)	RR Type ID	IETF Status	Defining Document
ATMA	Asynchronous Transfer Mode (ATM) address for a host	34	Not Submitted	ATM Name System Specification by the ATM Forum (97)
CERT	Certificate or Certificate Revocation List	37	Standards Track	RFC 4398 (98)
CNAME	Alias host name for a host	5	Standard	RFC 1035 (23)
DHCID	Associates a DHCP client's identity with a DNS name	49	Standards Track	RFC 4701 (99)
DLV	Authoritative zone signature for a trust anchor	32769	Informational (DNSSEC)	RFC 4431 (100)
DNAME	Alias domain name	39	Proposed Standard	RFC 2672 (101)
DNSKEY	Zone public key	48	Standards Track (DNSSEC)	RFC 4034 (47)
DS	Signature for delegated child zone	43	Standards Track (DNSSEC)	RFC 4034 (47)
GID	Group ID	102	RESERVED	IANA-Reserved
GPOS	Lat/Long/Altitude for a given host—superseded by LOC	27	Experimental	RFC 1712 (102)
HINFO	CPU and OS information for a host	13	Standard	RFC 1035 (23)
HIP	Host Identity Protocol	55	Experimental	RFC 5205 (103)
IPSECKEY	Public key for a given DNS name for use with IPSec	45	Proposed Standard	RFC 4025 (104)
ISDN	Integrated Services Digital Network (ISDN) address and subaddress for a given host	20	Experimental	RFC 1183 (95)
KEY	Superseded by DNSKEY within DNSSEC but still used by SIG(0) and TKEY	25	Proposed Standard	RFC 2536 (105)
KX	Intermediary domain to obtain a key for a host in given domain	36	Informational	RFC 2230 (106)
LOC	Lat/Long/Altitude and precision for a given host	29	Uncommon	RFC 1876 (107)

TABLE C-1: *Continued*

RRType (or QType)	RR Purpose (i.e., Rdata contents)	RR Type ID	IETF Status	Defining Document
MB	Mailbox name for a given email ID	7	Experimental	RFC 1035 (23)
MD	Mail delivery host for a given domain	3	Obsolete	RFC 1035 (23)
MF	Host that will accept mail for forwarding to a given domain	4	Obsolete	RFC 1035 (23)
MG	Mail group mailbox name for a given email ID	8	Experimental	RFC 1035 (23)
MINFO	Mailbox names for sending account requests or error reports for a given mailbox name	14	Experimental	RFC 1035 (23)
MR	Alias for a mailbox name	9	Experimental	RFC 1035 (23)
MX	Mail Exchanger for email host resolution	15	Standard	RFC 1035 (23)
NAPTR	Uniform Resource Identifier for a generic string—used for DDDS, ENUM applications	35	Standards Track	RFC 3761 (108)
NS	Name server for a given domain name	2	Standard	RFC 1035 (23)
NSAP	Network Services Access Point address for a host	22	Uncommon	RFC 1706 (109)
NSAP-PTR	Hostname for a given NSAP address	23	Uncommon	RFC 1706 (109)
NSEC	Authenticated confirmation or denial of existence of a resource record set for DNSSEC	47	Standards Track (DNSSEC)	RFC 4034 (47)
NSEC3	Authenticated denial of existence of a resource record set for DNSSEC (without trivial zone enumeration obtainable with NSEC)	50	Standards Track (DNSSEC)	RFC 5155 (110)
NSEC3 PARAM	NSEC3 parameters used to calculate hashed owner names	51	Standards Track (DNSSEC)	RFC 5155 (110)
NULL	Up to 65535 bytes of anything for a given host	10	Experimental	RFC 1035 (23)
NXT	Superseded by NSEC	30	Obsolete (DNSSEC)	RFC 3755 (111)

TABLE C-1: *Continued*

RRType (or QType)	RR Purpose (i.e., Rdata contents)	RR Type ID	IETF Status	Defining Document
PTR	Hostname for a given IPv4 or IPv6 address	12	Standard	RFC 1035 (23)
PX	X.400 mapping for a given domain name	26	Uncommon	RFC 2163 (112)
RP	Email address and TXT record pointer for more info for a host	17	Experimental	RFC 1183 (95)
RRSIG	Signature for a resource record set of a given domain name, class and RR Type	46	Standards Track (DNSSEC)	RFC 4034 (47)
RT	Proxy hostname for a given host that is not always connected	21	Experimental	RFC 1183 (95)
SIG	Superseded by RRSIG within DNSSEC; used by SIG(0) and TKEY	24	Proposed Standard	RFC 2536 (105)
SOA	Authority information for a zone	6	Standard	RFC 1035 (23)
SPF	Sender Policy Framework—enables a domain owner to indentify hosts authorized to send emails from the domain	99	Experimental	RFCs 4408 (113), 4409 (114)
SRV	Host providing specified services in a domain	33	Standards Track	RFC 2782 (115)
SSHFP	Secure Shell Fingerprints—enables verification of SSH host keys using DNSSEC	44	Standards Track	RFC 4255 (116)
TXT	Arbitrary text associated with a host	16	Standard	RFC 1035 (23)
UID	User ID	101	RESERVED	IANA-Reserved
UINFO	User Info	100	RESERVED	IANA-Reserved
UNSPEC	Unspecified	103	RESERVED	IANA-Reserved
WKS	Services available via a given protocol at a specified IP address for a host—SRV RR more commonly used today	11	Standard	RFC 1035 (23)
X25	X.25 Packet Switched Data Network (PSDN)	19	Experimental	RFC 1183 (95)

GLOSSARY

The key terms used throughout this book are summarized in this appendix.

DHCP Dynamic Host Configuration Protocol automates IP address assignment to network hosts or devices. DHCP technology applies to both IPv4 and IPv6 address assignment, though "DHCP" typically refers to assignment of IPv4 addresses.

A-DHCP Automatic DHCP, infinite lease

D-DHCP Dynamic DHCP

M-DHCP Manual DHCP, fixed IP address-MAC address

DHCPv6 DHCP specifically for IPv6 addresses, not version 6 of the DHCP protocol.

DNS Domain Name System, the distributed database of Internet name, address, and other information.

DNSSEC DNS Security extensions provide resolution data origin authentication (the source truly published this data), data integrity verification (the data was not modified en route), and authenticated denial of existence of data (the requested data truly does not exist in this zone).

FCAPS An initialism of the five major functional areas of network management, namely Fault, Configuration, Accounting, Performance and Security as described in ITU Telecommunications Management Network standards.

Host An end device that communicates on an IP network, such as a server, laptop, VoIP phone, etc. We contrast an end device with network infrastructure devices such as routers and switches.

IP Internet Protocol, the network layer used across the Internet and all IP networks. IP generically refers to all IP versions, while IPv4 and IPv6 denote respective IP versions.

IPAM IP address management, the disciplined approach to managing IP address space and associated DHCP and DNS services.

Introduction to IP Address Management, By Timothy Rooney

ITIL® Information Technology Infrastructure Library, developed by the UK Office of Government and Commerce (OGC), is a best practices framework with the perspective of the information technology (IT) organization as a service provider to the enterprise.

KSK Key Signing Key, used within DNSSEC to sign a zone signing key (ZSK), which in turn signs DNS zone data. The public KSK is published in a DNSKEY resource record within the corresponding secure zone. Resolvers or recursive servers configured to trust this zone's data must have a copy of the public KSK (or that of a parent zone's or lookaside validator's zone) configured as trust-anchors within their respective configurations.

NAT Network Address Translation—a gateway or firewall that changes (translates) an IP address within the IP packet header prior to forwarding; commonly used in enterprise networks to translate internal private IP addresses to external public IP addresses.

TCP Transmission Control Protocol, the connection-oriented transport layer protocol within the TCP/IP protocol suite.

UDP User Datagram Protocol, the connectionless transport layer protocol within the TCP/IP protocol suite.

ZSK Zone Signing Key, used within DNSSEC to sign zone information, meaning each of the resource record sets within the zone.

BIBLIOGRAPHY

1. **ISO/IEC.** *Information technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. Genera, Switzerland: ISO/IEC, November, 1994 (Second Edition). ISO/IEC 7498-1:1994(E).
2. **The ISC Domain Survey.** [Online] www.isc.org/solutions/survey.
3. **Postel, J.** *DoD Standard Internet Protocol*. s.l.: IETF, January, 1980. RFC 760.
4. —. *Internet Protocol*. s.l.: IETF, September, 1981. RFC 791.
5. **Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., Lear, E.** *Address Allocation for Private Internets*. s.l.: IETF, February, 1996. RFC 1918.
6. **Hinden, R.** *Applicability Statement for the Implementation of Classless Inter-Domain Routting (CIDR)*. s.l.: IETF, September, 1993. RFC 1517.
7. **Rekhter, Y., Li, T.** *An Architecture for IP Address Allocation with CIDR*. s.l.: IETF, September, 1993. RFC 1518.
8. **Fuller, V., Li, T., Yu, J., Varadhan, K.** *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*. s.l.: IETF, September, 1993. RFC 1519.
9. **IANA.** *Special-Use IPv4 Addresses*. s.l.: IETF, September, 2002. RFC 3330.
10. **Cotton, M., Vegoda, L.** *Special Use IPv4 Addresses*. s.l.: IETF, August, 2009. draft-iana-rfc3330bis-11.txt.
11. **Hinden, R., Deering, S.** *IP Version 6 Addressing Architecture*. s.l.: IETF, February, 2006. RFC 4291.
12. **Internet Assigned Numbers Authority (IANA).** Internet Protocol Version 6 Address Space. www.iana.org. [Online] [Cited: October 19, 2009.] http://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml.
13. **Hinden, R., Deering, S., Nordmark, E.** *IPv6 Global Unicast Address Format*. s.l.: IETF, August, 2003. RFC 3587.
14. **Hinden, R., Haberman, B.** *Unique Local IPv6 Unicast Addresses*. s.l.: IETF, October, 2005. RFC 4193.
15. **Microsoft.** IPv6 Address Autoconfiguration. *www.microsoft.com*. [Online] [Cited: October 19, 2009.] http://msdn.microsoft.com/en-us/library/aa917171.aspx.
16. **Loughney, J. Ed.** *IPv6 Node Requirements*. s.l.: IETF, April, 2006. RFC 4294.
17. **Hubbard, K., Kosters, M., Conrad, D., Karrrenberg, D., Postel, J.** *Internet Registry IP Allocation Guidelines*. s.l.: IETF, November, 1996. RFC 2050.

Introduction to IP Address Management, By Timothy Rooney

18. **Huitema, C.** *The H Ratio for Address Assignment Efficiency*. s.l.: IETF, November, 1994. RFC 1715.
19. **Durand, A., Huitema, C.** *The Host-Density Ratio for Address Assignment Efficiency: An update on the H ratio*. s.l.: IETF, November, 2001. RFC 3194.
20. **Internet Systems Consortium.** *dhcpd.conf man page*. Redwood City, CA: Internet Systems Consortium, Inc. ("ISC"), 2009.
21. **Droms, R. Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., Carney, M.** *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. s.l.: IETF, July, 2003. RFC 3315.
22. **Narten, T., Draves, R.** *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. s.l.: IETF, January, 2001. RFC 3041.
23. **Mockapetris, P. V.** *Domain Names—Implementation and Specification*. s.l.: IETF, November, 1987. RFC 1035.
24. **Rooney, Timothy.** *IP Address Management: Pricinciples and Practice*. s.l.: IEEE Press/John Wiley and Sons, 2010.
25. **Johnston, M., Venaas, S. Ed.** *Dynamic Host Configuration Protocol (DHCP) Options for Intel Preboot eXecution Environment (PXE)*. s.l.: IETF, November, 2006. RFC 4578.
26. **Mealling, M.** *Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database*. s.l.: IETF, October, 2002. RFC 3403.
27. —. *Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm*. s.l.: IETF, October, 2002. RFC 3402.
28. **Mendel, Thomas.** *IP Address Management Market Overview*. s.l.: Forrester Research, June, 2004. Market Overview.
29. **Blanchet, M.** *A Flexible Method for Managing the Assignment of Bits of an IPv6 Address Block*. s.l.: IETF, April, 2003. RFC 3531.
30. **Cherenson, Andrew.** *nslookup man page (BIND distribution)*. Redwood City, CA: Internet Systems Consortium, Inc. ("ISC").
31. **Internet Systems Consortium.** *dig man page (with BIND distribution)*. Redwood City, CA: Internet Systems Consortium ("ISC").
32. **Rooney, Tim.** *Automating IP Address Management Workflow*. Santa Clara, CA: BT INS, Inc., August, 2005.
33. **Huston, Geoff.** IPv4 Address Report. [Online] [Cited: October 28, 2009.] http://ipv4.potaroo.net.
34. **Durand, A., Ihren, J.** *DNS IPv6 Transport Operational Guidelines*. s.l.: IETF, September, 2004. RFC 3901.
35. **Chown, T., Venaas, S., Strauf, C.** *Dynamic Host Configuration Protocol (DHCP): IPv4 and IPv6 Dual-Stack Issues*. s.l.: IETF, May, 2006. RFC 4477.
36. **Tsirtsis, G., Srisuresh, P.** *Network Address Translation—Protocol Translation (NAT-PT)*. s.l.: IETF, February, 2000. RFC 2766.
37. **Nordmark, E.** *Stateless IP/ICMP Translation Algorithm (SIIT)*. s.l.: IETF, February, 2000. RFC 2765.
38. **Leech, M., Ganis, M. Lee, Y., Kuris, R., Koblas, D., Jones, L.** *SOCKS Protocol Version 5*. s.l.: IETF, March, 1996. RFC 1928.
39. **Kitamura, H.** *A SOCKS-Based IPv6/IPv4 Gateway Mechanism*. s.l.: IETF, April, 2001. RFC 3089.

40. **Internet Systems Consortium.** *dhcpd.conf man page*. Redwood City, CA: Internet Systems Consortium, 2009.
41. **Cisco Systems, Inc.** *Cisco Network Admission Control (NAC)*. San Jose, CA: Cisco Systems, Inc., 2009.
42. **Microsoft Corporation.** *Introduction to Nework Access Protection*. Redmond, WA: Microsoft Corporation, 2008.
43. **Droms, R., Arbaugh, W.** Eds. *Authentication for DHCP Messages*. s.l.: IETF, June, 2001. RFC 3118.
44. **Atkins, D., Austein, R.** *Threat Analysis of the Domain Name System (DNS)*. s.l.: IETF, August, 2004. RFC 3833.
45. **Eastlake, D. 3rd.** *Domain Name System Security Extensions*. s.l.: IETF, March, 1999. RFC 2535.
46. **Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.** *DNS Security Introduction and Requirements*. s.l. : IETF, March, 2005. RFC 4033.
47. —. *Resource Records for DNS Security Extensions*. s.l.: IETF, March, 2005. RFC 4034.
48. —. *Protocol Modifications for the DNS Security Extensions*. s.l.: IETF, March, 2005. RFC 4035.
49. **Internet Systems Consortium.** *BIND 9 Administrator Reference Manual*. Redwood City, CA: Internet Systems Consortium, Inc. (“ISC”), 2009.
50. **Alexander, S., Droms, R.** *DHCP Options and BOOTP Vendor Extensions*. s.l. : IETF, March, 1997. RFC 2132.
51. **Lemon, T., Cheshire, S., Volz, B.** *The Classless Static Route Option for Dynamic Host Configuration Protocol (DHCP) Version 4*. s.l.: IETF, December, 2002. RFC 3442.
52. **Droms, R., Fong, K.** *NetWare/IP Domain Name and Information*. s.l.: IETF, November, 1997. RFC 2242.
53. **Stump, G., Droms, R., Gu, Y., Vyaghrapuri, R., Demirtjis, A., Beser, B., Privat, J.** *The User Class Option for DHCP*. s.l.: IETF, November, 2000. RFC 3004.
54. **Perkins, C., Guttman, E.** *DHCP Options for Service Location Protocol*. s.l.: IETF, June, 1999. RFC 2610.
55. **Park, S., Kim, P., Volz, B.** *Rapid Commit Option for the Dynamic Host Configuration Protocol version 4 (DHCPv4)*. s.l.: IETF, March, 2005. RFC 4039.
56. **Stapp, M., Volz, B., Rekhter, Y.** *The Dynamic Host Configuration Protocol (DHCP) Client Fully Qualified Domain Name (FQDN) Option*. s.l.: IETF, October, 2006. RFC 4702.
57. **Patrick, M.** *DHCP Relay Agent Information Option*. s.l.: IETF, January, 2001. RFC 3046.
58. **Monia, C., Tseng, J., Gibbons, K.** *The IPv4 Dynamic Host Configuration Protocol (DHCP) Option for the Internet Storage Name Service*. s.l.: IETF, September, 2005. RFC 4174.
59. **Droms, R.** *Unused Dynamic Host Configuration Protocol (DHCP) Option Codes*. s.l.: IETF, January, 2004. RFC 3679.
60. **Provan, D.** *DHCP Options for Novell Directory Services*. s.l.: IETF, November, 1997. RFC 2241.

61. **Chowdhury, K., Yegani, P., Madour, L.** *Dynamic Host Configuration Protocol (DHCP) Options for Broadcast and Multicast Control Servers*. s.l.: IETF, November, 2005. RFC 4280.
62. **Woundy, R., Kinnear, K.** *Dynamic Host Configuration Protocol (DHCP) Leasequery*. s.l.: IETF, February, 2006. RFC 4388.
63. **Drach, S.** *DHCP Option for The Open Group's User Authentication Protocol*. s.l.: IETF, January, 1999. RFC 2485.
64. **Schulzrinne, H.** *Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information*. s.l.: IETF, November, 2006. RFC 4776.
65. **Lear, E., Eggert, P.** *Timezone Options for DHCP*. s.l.: IETF, April, 2007. RFC 4833.
66. **Troll, R.** *DHCP Option to Disable Stateless Auto-Configuration in IPv4 Clients*. s.l.: IETF, May, 1999. RFC 2563.
67. **Smith, C.** *The Name Service Search Option for DHCP*. s.l.: IETF, September, 2000. RFC 2937.
68. **Waters, G.** *The IPv4 Subnet Selection Optoin for DHCP*. s.l.: IETF, November, 2000. RFC 3011.
69. **Aboba, B., Cheshire, S.** *Dynamic Host Configuration Protocol (DHCP) Domain Search Option*. s.l.: IETF, November, 2002. RFC 3397.
70. **Schulzrinne, H.** *Dynamic Host Configuration Protocol (DHCP-for-IPv4) Option for Session Initiation Protocol (SIP) Servers*. s.l.: IETF, August, 2002. RFC 3361.
71. **Beser, B., Duffy, P. Ed.** *Dynamic Host Configuration Protocol (DHCP) Option for CableLabs Client Configuration*. s.l.: IETF, March, 2003. RFC 3495.
72. **Polk, J., Schnizlein, J., Linsner, M.** *Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information*. s.l.: IETF, July, 2004. RFC 3825.
73. **Littlefield, J.** *Vendor-Identifying Vendor Options for Dynamic Host Configuration Protocol Version 4 (DHCPv4)*. s.l.: IETF, October, 2004. RFC 3925.
74. **Morand, L., Yegin, A., Kumar, S., Madanapalli, S.** *DHCP Options for Protocol for Carrying Authentication for Network Access (PANA) Authentication Agents*. s.l.: IETF, May, 2008. RFC 5192.
75. **Schulzrinne, H., Polk, J., Tschofenig, H.** *Discovering Location-to-Service Translation (LoST) Servers Using the Dynamic Host Configuration Protocol (DHCP)*. s.l.: IETF, August, 2008. RFC 5223.
76. **Calhoun, P.** *Control And Provisioning of Wireless Access Points (CAPWAP) Access Controller DHCP Option*. s.l.: IETF, March, 2009. RFC 5417.
77. **Bajko, G., Das, S.** *Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Options for IEEE 802.21 Mobility Services (MoS) Discovery*. s.l.: IETF, December, 2009. RFC 5678.
78. **Volz, B.** *Reclassifying Dynamic Host Configuration Protocol Version 4 (DHCPv4) Options*. s.l.: IETF, November, 2004. RFC 3942.
79. **Hankins, D.** *Dynamic Host Configuration Protocol Options Used by PXELINUX*. s.l.: IETF, December, 2007. RFC 5071.
80. **Schulzrinne, H., Volz, B.** *Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers*. s.l.: IETF, July, 2003. RFC 3319.

81. **Droms, R.** Ed. *DNS Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. s.l.: IETF, December, 2003. RFC 3646.
82. **Troan, O., Droms, R.** *IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) Version 6*. s.l.: IETF, December, 2003. RFC 3633.
83. **Kalusivalingam, V.** *Network Information Service (NIS) Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. s.l.: IETF, October, 2004. RFC 3898.
84. —. *Simple Network Time Protocol (SNTP) Configuration Option for DHCPv6*. s.l.: IETF, May, 2005. RFC 4075.
85. **Venaas, S., Chown, T., Volz, B.** *Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. s.l.: IETF, November, 2005. RFC 4242.
86. **Volz, B.** *Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Relay Agent Remote-ID Option*. s.l.: IETF, August, 2006. RFC 4649.
87. —. *Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Relay Agent Subscriber-ID Option*. s.l.: IETF, June, 2006. RFC 4580.
88. —. *The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option*. s.l.: IETF, October, 2006. RFC 4704.
89. **Zeng, S., Volz, B., Kinnear, K., Brzozowski, J.** *DHCPv6 Relay Agent Echo Request Option*. s.l.: IETF, September, 2007. RFC 4994.
90. **Brzozowski, J., Kinnear, K., Volz, B., Zeng, S.** *DHCPv6 Leasequery*. s.l.: IETF, September, 2007. RFC 5007.
91. **Jang, H., Yegin, A., Chowdhury, K., Choi, J.** *DHCP Options for Home Information Discovery in MIPv6*. s.l.: IETF, May, 2008. draft-ietf-mip6-hiopt-17.
92. **Stapp, M.** *DHCPv6 Bulk Leasequery*. s.l.: IETF, February, 2009. RFC 5460.
93. **Thomson, S., Huitema, C., Ksinant, V., Souissi, M.** *DNS Extensions to Support IP Version 6*. s.l.: IETF, October, 2003. RFC 3596.
94. **Crawford, M., Huitema, C.** *DNS Extensions to Support IPv6 Address Aggregation and Renumbering*. s.l.: IETF, July, 2000. RFC 2874.
95. **Everhart, C. F., Mamakos, L. A., Ullmann, R., Mockapetris, P. V.** *New DNS RR Definitions*. s.l.: IETF, October, 1990. RFC 1183.
96. **Koch, P.** *A DNS RR Type for Lists of Address Prefixes*. s.l.: IETF, June, 2001. RFC 3123.
97. **Committee, ATM Forum Technical.** *ATM Name System, V2.0*. s.l.: ATM Forum, 2000. AF-DANS-0152.000.
98. **Josefsson, S.** *Storing Certificates in the Domain Name System (DNS)*. s.l.: IETF, March, 2006. RFC 4398.
99. **Stapp, M., Lemon, T., Gustafsson, A.** *A DNS Resource Record (RR) for Encoding Dynamic Host Configuration Protocol (DHCP) Information (DHCID RR)*. s.l.: IETF, October, 2006. RFC 4701.
100. **Andrews, M., Weiler, S.** *The DNSSEC Lookaside Validation (DLV) DNS Resource Record*. s.l.: IETF, February, 2006. RFC 4431.
101. **Crawford, M.** *Non-Terminal DNS Name Redirection*. s.l.: IETF, August, 1999. RFC 2672.
102. **Farrell, C., Schulze, M., Pleitner, S., Baldoni, D.** *DNS Encoding of Geographical Location*. s.l.: IETF, November, 1994. RFC 1712.

103. **Nikander, P., Laganier, J.** *Host Identity Protocol (HIP) Domain Name System (DNS) Extensions*. s.l.: IETF, April, 2008. RFC 5205.
104. **Richardson, M.** *A Method for Storing IPsec Keying Material in DNS*. s.l.: IETF, March, 2005. RFC 4025.
105. **Eastlake, D.** 3rd**,** *DSA KEYs and SIGs in the Domain Name System (DNS)*. s.l.: IETF, March, 1999. RFC 2536.
106. **Atkinson, R.** *Key Exchange Delegation Record for the DNS*. s.l.: IETF, November, 1997. RFC 2230.
107. **Davis, C., Vixie, P., Goodwin, T., Dickinson, I.** *A Means for Expressing Location Information in the Domain Name System*. s.l.: IETF, January, 1996. RFC 1876.
108. **Faltstrom, P., Mealling, M.** *The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)*. s.l.: IETF, April, 2004. RFC 3761.
109. **Manning, B., Colella, R.** *DNS NSAP Resource Records*. s.l.: IETF, October, 1994. RFC 1706.
110. **Laurie, B., Sisson, G., Arends, R., Blacka, D.** *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*. s.l.: IETF, March, 2008. RFC 5155.
111. **Weiler, S.** *Legacy Resolver Compatibility for Delegation Signer (DS)*. s.l.: IETF, May, 2004. RFC 3755.
112. **Allocchio, C.** *Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM)*. s.l.: IETF, January, 1998. RFC 2163.
113. **Wong, M., Schlitt, W.** *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1*. s.l.: IETF, April, 2006. RFC 4408.
114. **Gellens, R., Klensin, J.** *Message Submission for Mail*. s.l.: IETF, April, 2006. RFC 4409.
115. **Gulbrandsen, A., Vixie, P., Esibov, L.** *A DNS RR for Specifying the Location of Services (DNS SRV)*. s.l.: IETF, February, 2000. RFC 2782.
116. **Schlyter, J., Griffin, W.** *Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints*. s.l.: IETF, January, 2006. RFC 4255.
117. **Rooney, Tim.** *IPv4-to-IPv6 Transition and Co-Existence Strategies*. Santa Clara, CA: BT INS, Inc., March, 2009.
118. —. *IPv6 Addressing and Management Challenges*. Santa Clara, CA: BT INS, Inc., March, 2008.
119. **Office of Government and Commerce (OGC).** ITIL(R) Home. *Official ITIL(R) Website*. [Online] APM Group Limited. [Cited: October 25, 2009.] http://www.itil-officialsite.com/home/home.asp.
120. **Kolkman, O., Gieben, R.** *DNSSEC Operational Practices*. s.l.: IETF, September, 2006. RFC 4641.
121. **Delgrossi, L., Berger, L. Eds.** *Internet Stream Protocol Version 2 (ST2) Protocol Specification—Version ST2+*. s.l.: IETF, August, 1995. RFC 1819.
122. **Gayraud, R., Lourdelet, B.,** *Network Time Protocol (NTP) Server Option for IPv6*. s.l.: IETF, January, 2010, draft-ietf-ntp-dhcpv6-ntp-opt-06.

INDEX

Introduction to IP Address Management, By Timothy Rooney