TCP/IP

TODD LAMMLE

Bestselling author of
CCNA Routing and Switching Complete Study Guide: Exams 100-105, 200-105, 200-125

SYBEX
A Wiley Brand

TCP/IP

Todd Lammle

SYBEX
A Wiley Brand

Senior Acquisitions Editor: Kenyon Brown
Development Editor: Kim Wimpsett
Technical Editors: Todd Montgomery
Production Editor: Christine O'Connor
Copy Editor: Judy Flynn
Editorial Manager: Mary Beth Wakefield
Production Manager: Kathleen Wisor
Executive Editor : Jim Minatel
Book Designers: Judy Fung and Bill Gibson
Proofreader: Josh Chase, Word One New York
Indexer: John Sleeva
Project Coordinator, Cover: Brent Savage
Cover Designer: Wiley
Cover Image: Getty Images Inc./Jeremy Woodhouse

Published simultaneously in Canada

Manufactured in the United States of America

ISBN: 978-1-119-47170-7 (ebk.)
ISBN: 978-1-119-47270-4 (ebk.)

Manufactured in the United States of America

Library of Congress Control Number: 2016942433

Contents

List of Tables

List of Illustrations

Acknowledgments

There are many people that work to put a book together, and as an author, I dedicated an enormous amount of time to write this book, but it would have never been published without the dedicated, hard work of many other people.

Kenyon Brown, my acquisitions editor, is instrumental to my success in the world of Cisco certification. Ken, I look forward to our continued progress together in both the print and video markets!

Christine O’Connor, my production editor, and Judy Flynn, my copyeditor, were my rock and foundation for formatting and intense editing of every page in this book. This amazing team gives me the confidence to help keep me moving during the difficult and very long days, week after week. How Christine stays so organized with all my changes, as well as making sure every figure is in the right place in the book is still a mystery to me! You’re amazing, Christine! Thank you! Judy understands my writing style so well now, after doing at least a dozen books with me, that she even sometimes finds a technical error that may have slipped through as I was going through the material. Thank you, Judy, for doing such a great job! I truly thank you both.

About the Author

Todd Lammle is the authority on Cisco certification and internetworking and is Cisco certified in most Cisco certification categories. He is a world-renowned author, speaker, trainer, and consultant. Todd has three decades of experience working with LANs, WANs, and large enterprise licensed and unlicensed wireless networks, and lately he's been implementing large Cisco Firepower networks. His years of real-world experience are evident in his writing; he is not just an author but an experienced networking engineer with very practical experience working on the largest networks in the world, at such companies as Xerox, Hughes Aircraft, Texaco, AAA, Cisco, and Toshiba, among many others. Todd has published over 60 books, including the very popular *CCNA: Cisco Certified Network Associate Study Guide*, *CCNA Wireless Study Guide*, *CCNA Data Center Study Guide*, and *SSFIPS (Firepower)*, all from Sybex. He runs an international consulting and training company based in Colorado, Texas, and San Francisco.

You can reach Todd through his webesite at www.lammle.com/ccna.

Introduction

Welcome to the exciting world of Cisco certification! If you've picked up this book because you want to improve yourself and your life with a better, more satisfying, and secure job, you've done the right thing. Whether you're striving to enter the thriving, dynamic IT sector or seeking to enhance your skill set and advance your position within it, being Cisco certified can seriously stack the odds in your favor to help you attain your goals!

Cisco certifications are powerful instruments of success that also markedly improve your grasp of all things internetworking. If you're considering studying for the ICND1 Exam 100-105, this book complements the *CCENT ICND1 Study Guide: Exam 100-105, Third Edition*. For more information, visit http://www.wiley.com/WileyCDA/WileyTitle/productCd-1119288789.html.

Although it's now common knowledge that Cisco rules routing and switching, the fact that it also rocks the voice, data center, and service provider worlds is also well-recognized. And Cisco certifications reach way beyond the popular but less extensive certifications like those offered by CompTIA and Microsoft to equip you with indispensable insight into today's vastly complex networking realm. Essentially, by deciding to become Cisco certified, you're proudly announcing that you want to become an unrivaled networking expert —a goal that this book will get you well on your way to achieving. Congratulations in advance on the beginning of your brilliant future!

NOTE

For up-to-the-minute updates covering additions or modifications to the Cisco certification exams, as well as additional study tools, videos, review questions, and bonus materials, be sure to visit the Todd Lammle websites and forum at www.lammle.com/ccna.

What Does This Book Cover?

You will learn the following concepts in this book:

Chapter 1: Introduction to TCP/IP This chapter provides you with the background necessary for success on the exam as well as in the real world with a thorough presentation of TCP/IP. This in-depth chapter covers the very beginnings of the Internet Protocol stack and goes all the way to IP addressing and understanding the difference between a network address and a broadcast address before finally ending with network troubleshooting.

Chapter 2: Easy Subnetting You'll actually be able to subnet a network in your head after reading this chapter if you really want to! And you'll find plenty of help in this chapter as long as you don't skip the written labs and review questions at the end.

Chapter 3: VLSMs, Summarization, and Troubleshooting TCP/IP Here, you'll find out all about variable length subnet masks (VLSMs) and how to design a network using VLSMs. This chapter will finish with summarization techniques and configurations. As with Chapter 2, plenty of help is there for you if you don't skip the written lab and review questions.

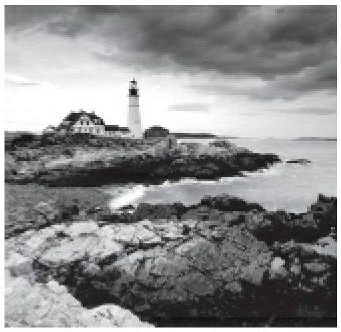
After reading the book, I can't stress this point enough, however: It's critical that you have some hands-on experience with Cisco routers.

If you can get a hold of some basic routers and switches, you're set. However, if you can't, I highly recommend you read *CCENT ICND1 Study Guide: Exam 100-105, Third Edition* (visit http://www.wiley.com/WileyCDA/WileyTitle/productCd-1119288789.html for more information) or, for more in-depth coverage, read *CCNA Routing and Switching Complete Study Guide: Exam 100-105, Exam 200-105, and Exam 200-125, Second Edition*. Visit http://www.wiley.com/WileyCDA/WileyTitle/productCd-1119288282.html for more information.

I've worked hard to provide hundreds of configuration examples in both books to help network administrators, or people who want to become network administrators, learn the skills they need to pass the CCENT and CCNA R/S exams.

Be sure to check the announcements section of my forum at www.lammle.com/ccna to find out how to download bonus material I created specifically for this book.

Chapter 1
Introduction to TCP/IP



The *Transmission Control Protocol/Internet Protocol (TCP/IP)* suite was designed and implemented by the Department of Defense (DoD) to ensure and preserve data integrity as well as maintain communications in the event of catastrophic war. So it follows that if designed and implemented correctly, a TCP/IP network can be a secure, dependable and resilient one. In this chapter, I'll cover the protocols of TCP/IP, and throughout this book, you'll learn how to create a solid TCP/IP network with Cisco routers and switches.

We'll begin by exploring the DoD's version of TCP/IP, then compare that version and its protocols with the OSI reference model that we discussed earlier.

Once you understand the protocols and processes used at the various levels of the DoD model, we'll take the next logical step by delving into the world of IP addressing and the different classes of IP addresses used in networks today.

NOTE

Subnetting is so vital, it will be covered in its own chapter, Chapter 2, "Easy Subnetting."

Because having a good grasp of the various IPv4 address types is critical to understanding IP addressing, subnetting, and variable length subnet masks (VLSMs), we'll explore these key topics in detail, ending this chapter by discussing the various types of IPv4 addresses.

NOTE

To find up-to-the-minute updates for this chapter, please see www.lammle.com/ccna or the book’s web page via www.sybex.com/go/ccna.

Introducing TCP/IP

TCP/IP is at the very core of all things networking, so I really want to ensure that you have a comprehensive and functional command of it. I'll start by giving you the whole TCP/IP backstory, including its inception, and then move on to describe the important technical goals as defined by its original architects. And of course I'll include how TCP/IP compares to the theoretical OSI model.

A Brief History of TCP/IP

TCP first came on the scene way back in 1973, and in 1978, it was divided into two distinct protocols: TCP and IP. Later, in 1983, TCP/IP replaced the Network Control Protocol (NCP) and was authorized as the official means of data transport for anything connecting to ARPAnet, the Internet's ancestor. The DoD's Advanced Research Projects Agency (ARPA) created this ancient network way back in 1957 in a cold war reaction to the Soviet's launching of *Sputnik*. Also in 1983, ARPA was redubbed DARPA and divided into ARPAnet and MILNET until both were finally dissolved in 1990.

It may be counterintuitive, but most of the development work on TCP/IP happened at UC Berkeley in Northern California, where a group of scientists were simultaneously working on the Berkeley version of UNIX, which soon became known as the Berkeley Software Distribution (BSD) series of UNIX versions. Of course, because TCP/IP worked so well, it was packaged into subsequent releases of BSD Unix and offered to other universities and institutions if they bought the distribution tape. So basically, BSD Unix bundled with TCP/IP began as shareware in the world of academia. As a result, it became the foundation for the tremendous success and unprecedented growth of today's Internet as well as smaller, private and corporate intranets.

As usual, what started as a small group of TCP/IP aficionados evolved, and as it did, the US government created a program to test any new published standards and make sure they passed certain criteria. This was to protect TCP/IP's integrity and to ensure that no developer changed anything too dramatically or added any proprietary features. It's this very quality—this open-systems approach to the TCP/IP family of protocols—that sealed its popularity because this quality guarantees a solid connection between myriad hardware and software platforms with no strings attached.

TCP/IP and the DoD Model

The DoD model is basically a condensed version of the OSI model that comprises four instead of seven layers:

- Process/Application layer
- Host-to-Host layer or Transport layer
- Internet layer
- Network Access layer or Link layer

Figure 1.1 offers a comparison of the DoD model and the OSI reference model. As you can see, the two are similar in concept, but each has a different number of layers with different names. Cisco may at times use different names for the same layer, such as both "Host-to-Host" and Transport" at the layer above the Internet layer, as well as "Network Access" and "Link" used to describe the bottom layer.

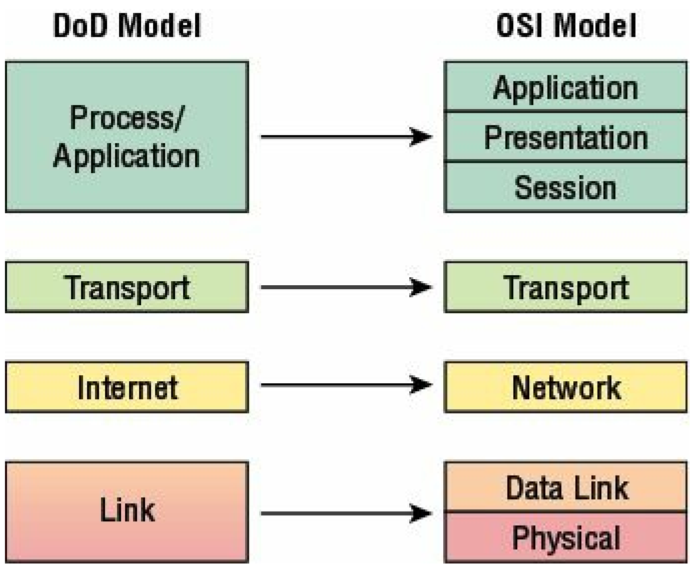


Figure 1.1 The DoD and OSI models

When the different protocols in the IP stack are discussed, the layers of the OSI and DoD models are interchangeable. In other words, be prepared for the exam objectives to call the Host-to-Host layer the

Transport layer!

A vast array of protocols join forces at the DoD model's *Process/Application layer*. These processes integrate the various activities and duties spanning the focus of the OSI's corresponding top three layers (Application, Presentation, and Session). We'll focus on a few of the most important applications found in the CCNA objectives. In short, the Process/Application layer defines protocols for node-to-node application communication and controls user-interface specifications.

The *Host-to-Host layer or Transport layer* parallels the functions of the OSI's Transport layer, defining protocols for setting up the level of transmission service for applications. It tackles issues like creating reliable end-to-end communication and ensuring the error-free delivery of data. It handles packet sequencing and maintains data integrity.

The *Internet layer* corresponds to the OSI's Network layer, designating the protocols relating to the logical transmission of packets over the entire network. It takes care of the addressing of hosts by giving them an IP (Internet Protocol) address and handles the routing of packets among multiple networks.

At the bottom of the DoD model, the *Network Access layer or Link layer* implements the data exchange between the host and the network. The equivalent of the Data Link and Physical layers of the OSI model, the Network Access layer oversees hardware addressing and defines protocols for the physical transmission of data. The reason TCP/IP became so popular is because there were no set physical layer specifications, so it could run on any existing or future physical network!

The DoD and OSI models are alike in design and concept and have similar functions in similar layers. Figure 1.2 shows the TCP/IP protocol suite and how its protocols relate to the DoD model layers.

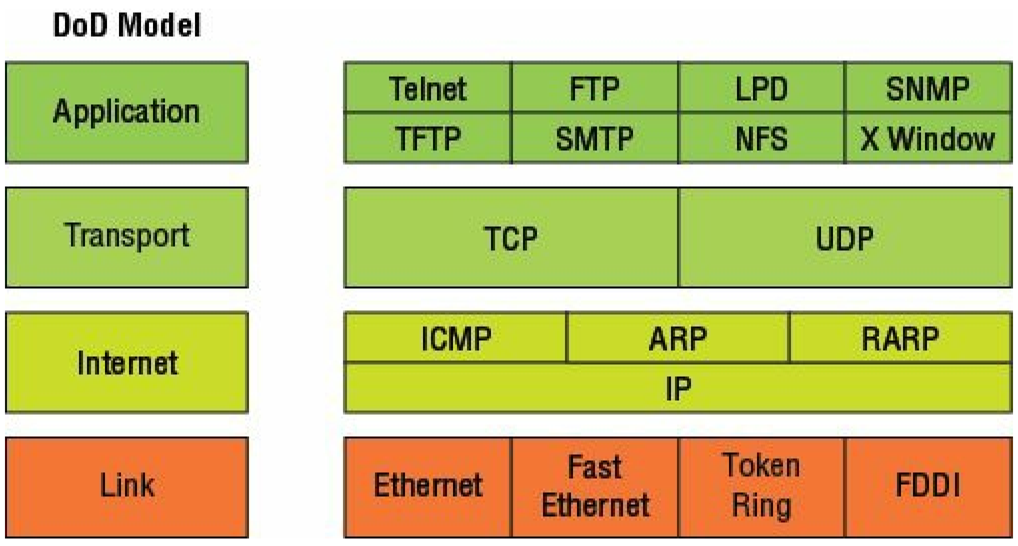


Figure 1.2 The TCP/IP protocol suite

In the following sections, we will look at the different protocols in more detail, beginning with those found at the Process/Application layer.

The Process/Application Layer Protocols

Coming up, I'll describe the different applications and services typically used in IP networks, and although there are many more protocols defined here, we'll focus in on the protocols most relevant to the CCNA objectives. Here's a list of the protocols and applications we'll cover in this section:

- Telnet
- SSH
- FTP
- TFTP
- SNMP
- HTTP
- HTTPS
- NTP
- DNS
- DHCP/BootP
- APIPA

Telnet

Telnet was one of the first Internet standards, developed in 1969, and is the chameleon of protocols—its specialty is terminal emulation. It allows a user on a remote client machine, called the Telnet client, to access the resources of another machine, the Telnet server, in order to access a command-line interface. Telnet achieves this by pulling a fast one on the Telnet server and making the client machine appear as though it were a terminal directly attached to the local network. This projection is actually a software image—a virtual terminal that can interact with the chosen remote host. A drawback is that there are no encryption techniques available within the Telnet protocol, so everything must be sent in clear text, including passwords! Figure 1.3 shows an example of a Telnet client trying to connect to a Telnet server.

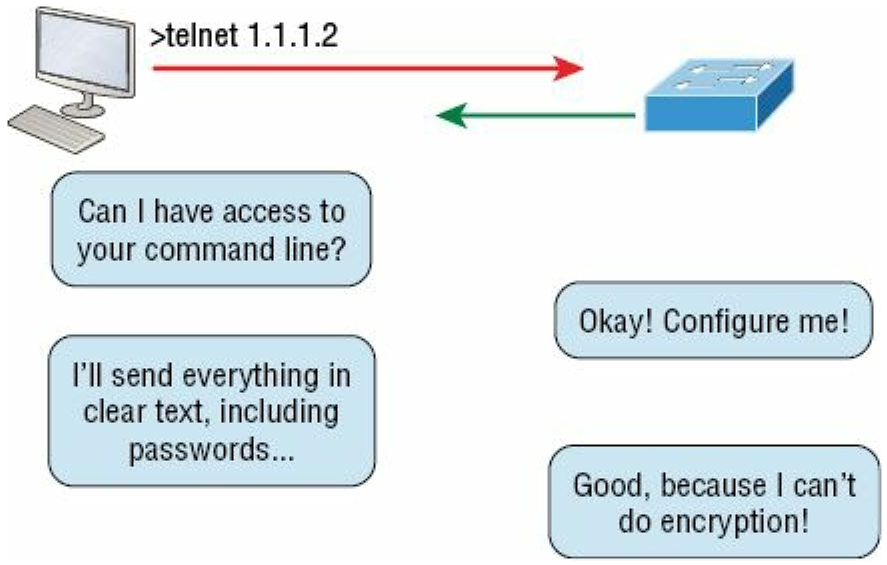

>telnet 1.1.1.2
Can I have access to your command line?
Okay! Configure me!
I'll send everything in clear text, including passwords...
Good, because I can't do encryption!

Figure 1.3 Telnet

These emulated terminals are of the text-mode type and can execute defined procedures such as displaying menus that give users the opportunity to choose options and access the applications on the duped server. Users begin a Telnet session by running the Telnet client software and then logging into the Telnet server. Telnet uses an 8-bit, byte-oriented data connection over TCP, which makes it very thorough. It's still in use today because it is so simple and easy to use, with very low overhead, but again, with everything sent in clear text, it's not recommended in production.

Secure Shell (SSH)

Secure Shell (SSH) protocol sets up a secure session that's similar to Telnet over a standard TCP/IP connection and is employed for doing things like logging into systems, running programs on remote systems, and moving files from one system to another. And it does all of this while maintaining an encrypted connection. Figure 1.4 shows a SSH client trying to connect to a SSH server. The client must send the data encrypted!

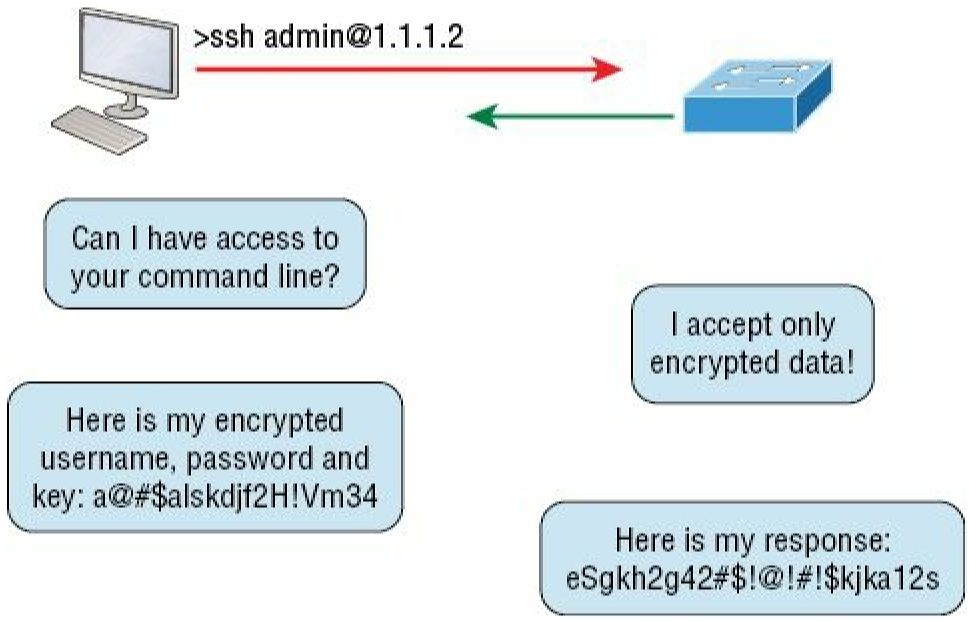

>ssh admin@1.1.1.2
Can I have access to your command line?
I accept only encrypted data!
Here is my encrypted username, password and key: a@#$alskdjf2H!Vm34
Here is my response: eSgkh2g42#$!@!#!$kjka12s

Figure 1.4 Secure Shell

You can think of it as the new-generation protocol that's now used in place of the antiquated and very unused rsh and rlogin—even Telnet.

File Transfer Protocol (FTP)

File Transfer Protocol (FTP) actually lets us transfer files, and it can accomplish this between any two machines using it. But FTP isn't just a protocol; it's also a program. Operating as a protocol, FTP is used by applications. As a program, it's employed by users to perform file tasks by hand. FTP also allows for access to both directories and files and can accomplish certain types of directory operations, such as relocating into different ones (Figure 1.5).

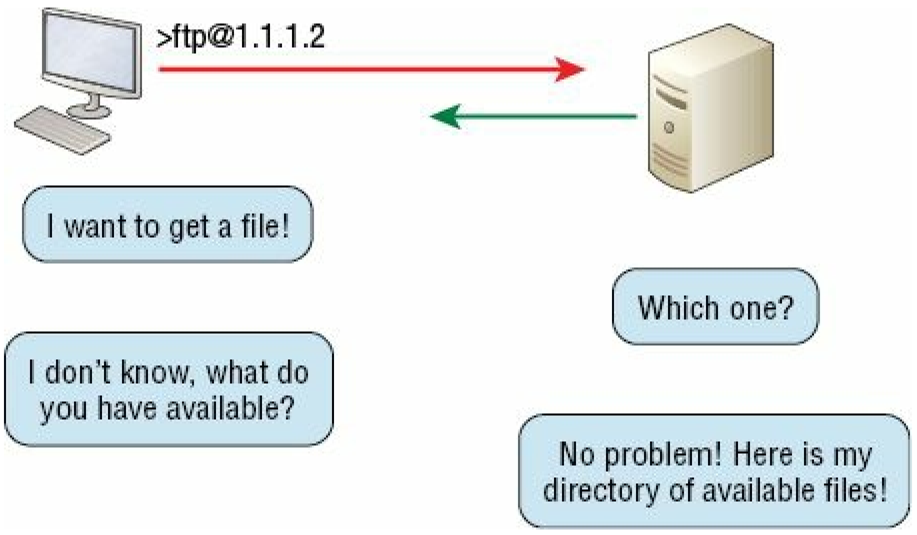

>ftp@1.1.1.2
I want to get a file!
Which one?
I don't know, what do you have available?
No problem! Here is my directory of available files!

Figure 1.5 FTP

But accessing a host through FTP is only the first step. Users must then be subjected to an authentication login that's usually secured with passwords and usernames implemented by system administrators to restrict access. You can get around this somewhat by adopting the username *anonymous*, but you'll be limited in what you'll be able to access.

Even when employed by users manually as a program, FTP's functions are limited to listing and manipulating directories, typing file contents, and copying files between hosts. It can't execute remote files as programs.

Trivial File Transfer Protocol (TFTP)

Trivial File Transfer Protocol (TFTP) is the stripped-down, stock version of FTP, but it's the protocol of choice if you know exactly what you want and where to find it because it's fast and so easy to use!

But TFTP doesn't offer the abundance of functions that FTP does because it has no directory-browsing abilities, meaning that it can only send and receive files (Figure 1.6). Still, it's heavily used for managing file systems on Cisco devices.

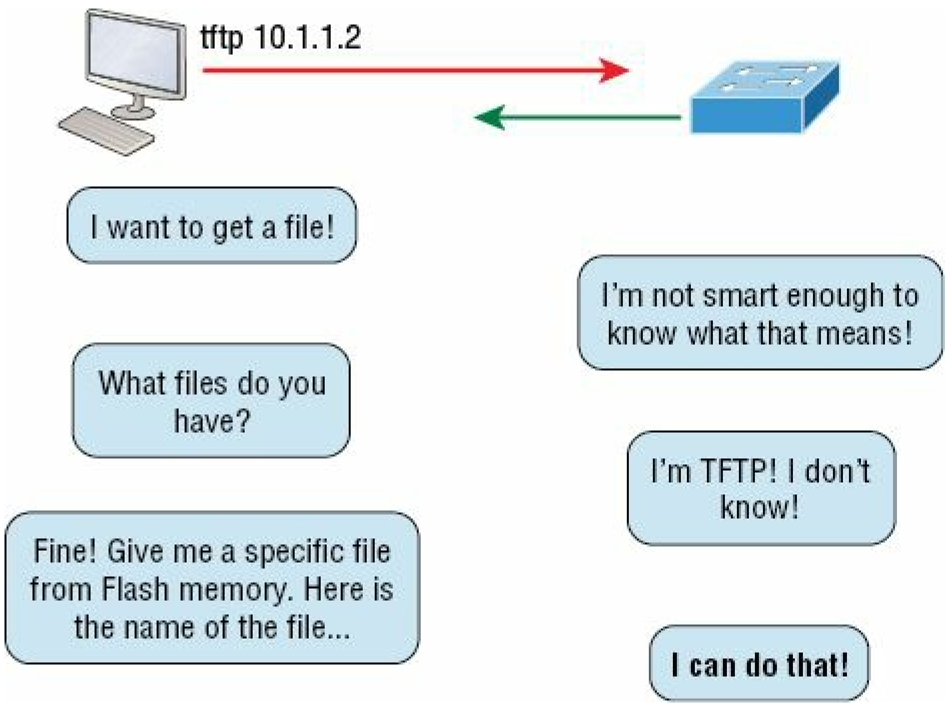

tftp 10.1.1.2
I want to get a file!
I'm not smart enough to know what that means!
What files do you have?
I'm TFTP! I don't know!
Fine! Give me a specific file from Flash memory. Here is the name of the file...
I can do that!

Figure 1.6 TFTP

This compact little protocol also skimps in the data department, sending much smaller blocks of data than FTP. Also, there's no authentication as with FTP, so it's even more insecure, and few sites support it because of the inherent security risks.

Real World Scenario

When Should You Use FTP?

Let's say everyone at your San Francisco office needs a 50 GB file emailed to them right away. What do you do? Many email servers would reject that email due to size limits (a lot of ISPs don't allow files larger than 5 MB or 10 MB to be emailed), and even if there are no size limits on the server, it would still take a while to send this huge file. FTP to the rescue!

If you need to give someone a large file or you need to get a large file from someone, FTP is a nice choice. To use FTP, you would need to set up an FTP server on the Internet so that the files can be shared.

Besides resolving size issues, FTP is faster than email. In addition, because it uses TCP and is connection-oriented, if the session dies, FTP can sometimes start up where it left off. Try that with your email client!

Simple Network Management Protocol (SNMP)

Simple Network Management Protocol (SNMP) collects and manipulates valuable network information, as you can see in Figure 1.7. It gathers data by polling the devices on the network from a network management station (NMS) at fixed or random intervals, requiring them to disclose certain information, or even asking for certain information from the device. In addition, network devices can inform the NMS station about problems as they occur so the network administrator is alerted.

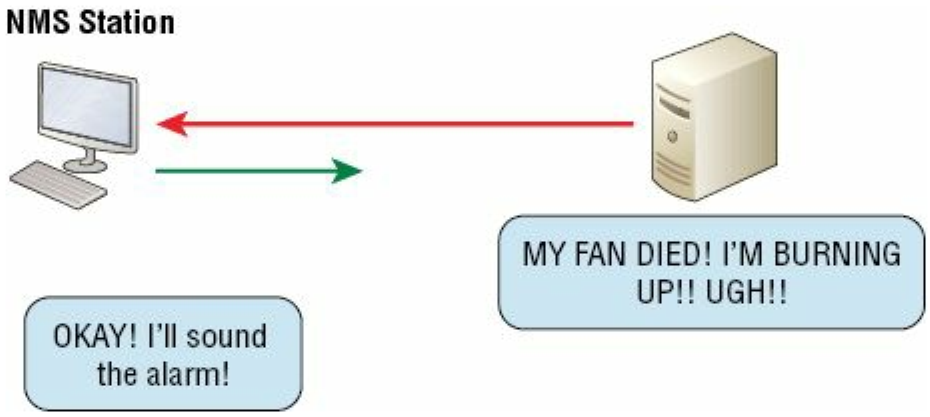

NMS Station
MY FAN DIED! I'M BURNING UP!! UGH!!
OKAY! I'll sound the alarm!

Figure 1.7 SNMP

When all is well, SNMP receives something called a *baseline*—a report delimiting the operational traits of a healthy network. This protocol can also stand as a watchdog over the network, quickly notifying managers of any sudden turn of events. These network watchdogs are called *agents*, and when aberrations occur, agents send an alert called a *trap* to the management station.

SNMP Versions 1, 2, and 3

SNMP versions 1 and 2 are pretty much obsolete. This doesn't mean you won't see them in a network now and then, but you'll only come across v1 rarely, if ever. SNMPv2 provided improvements, especially in performance. But one of the best additions was called GETBULK, which allowed a host to retrieve a large amount of data at once. Even so, v2 never really caught on in the networking world and SNMPv3 is now the standard. Unlike v1, which used only UDP, v3 uses both TCP and UDP and added even more security, message integrity, authentication, and encryption.

Hypertext Transfer Protocol (HTTP)

All those snappy websites comprising a mélange of graphics, text, links, ads, and so on rely on the *Hypertext Transfer Protocol (HTTP)* to make it all possible (Figure 1.8). It's used to manage communications between web browsers and web servers and opens the right resource when you click a link, wherever that resource may actually reside.

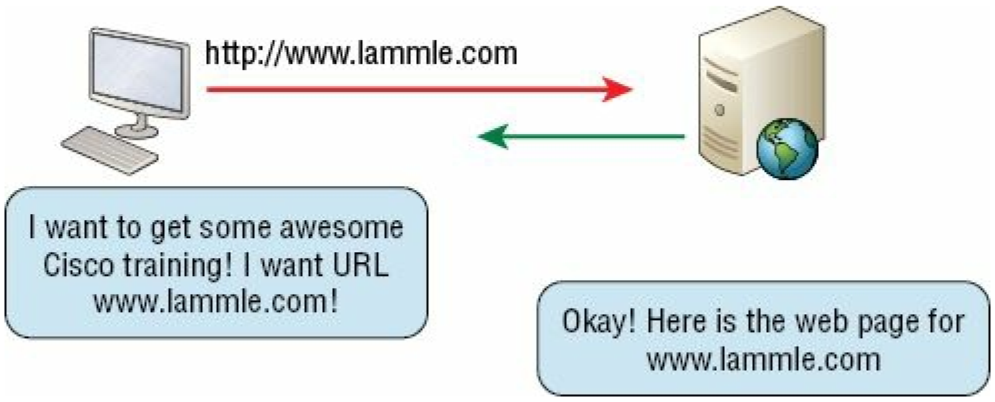

http://www.lammle.com
I want to get some awesome Cisco training! I want URL www.lammle.com!
Okay! Here is the web page for www.lammle.com

Figure 1.8 HTTP

In order for a browser to display a web page, it must find the exact server that has the right web page, plus the exact details that identify the information requested. This information must be then be sent back to the browser. Nowadays, it's highly doubtful that a web server would have only one page to display!

Your browser can understand what you need when you enter a Uniform Resource Locator (URL), which we usually refer to as a web address, such as, for example, http://www.lammle.com/forum and http://www.lammle.com/blog.

So basically, each URL defines the protocol used to transfer data, the name of the server, and the particular web page on that server.

Hypertext Transfer Protocol Secure (HTTPS)

Hypertext Transfer Protocol Secure (HTTPS) is also known as Secure Hypertext Transfer Protocol. It uses Secure Sockets Layer (SSL). Sometimes you'll see it referred to as SHTTP or S-HTTP, which were slightly different protocols, but since Microsoft supported HTTPS, it became the de facto standard for securing web communication. But no matter—as indicated, it's a secure version of HTTP that arms you with a whole bunch of security tools for keeping transactions between a web browser and a server secure.

It's what your browser needs to fill out forms, sign in, authenticate, and encrypt an HTTP message when you do things online like make a reservation, access your bank, or buy something.

Network Time Protocol (NTP)

Kudos to Professor David Mills of the University of Delaware for coming up with this handy protocol that's used to synchronize the clocks on our computers to one standard time source (typically, an atomic clock). *Network Time Protocol (NTP)* works by synchronizing devices to ensure that all computers on a given network agree on the time (Figure 1.9).

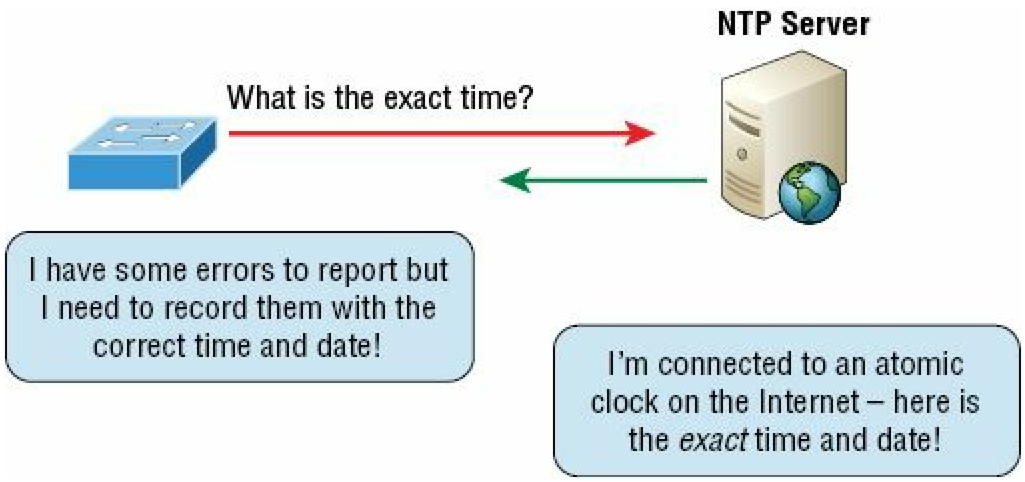

NTP Server
What is the exact time?
I have some errors to report but I need to record them with the correct time and date!
I'm connected to an atomic clock on the Internet – here is the *exact* time and date!

Figure 1.9 NTP

This may sound pretty simple, but it's very important because so many of the transactions done today are time and date stamped. Think about databases—a server can get messed up pretty badly and even crash if it's out of sync with the machines connected to it by even mere seconds! You can't have a transaction entered by a machine at, say, 1:50 a.m. when the server records that transaction as having occurred at 1:45 a.m. So basically, NTP works to prevent a "back to the future *sans* DeLorean" scenario from bringing down the network—very important indeed!

Domain Name Service (DNS)

Domain Name Service (DNS) resolves hostnames—specifically, Internet names, such as www.lammle.com. But you don't have to actually use DNS. You just type in the IP address of any device you want to communicate with and find the IP address of a URL by using the Ping program. For example, >ping www.cisco.com will return the IP address resolved by DNS.

An IP address identifies hosts on a network and the Internet as well, but DNS was designed to make our lives easier. Think about this: What would happen if you wanted to move your web page to a different service provider? The IP address would change and no one would know what the new one is. DNS allows you to use a domain name to specify an IP address. You can change the IP address as often as you want and no one will know the difference.

To resolve a DNS address from a host, you'd typically type in the URL from your favorite browser, which would hand the data to the Application layer interface to be transmitted on the network. The application would look up the DNS address and send a UDP request to your DNS server to resolve the name (Figure 1.10).

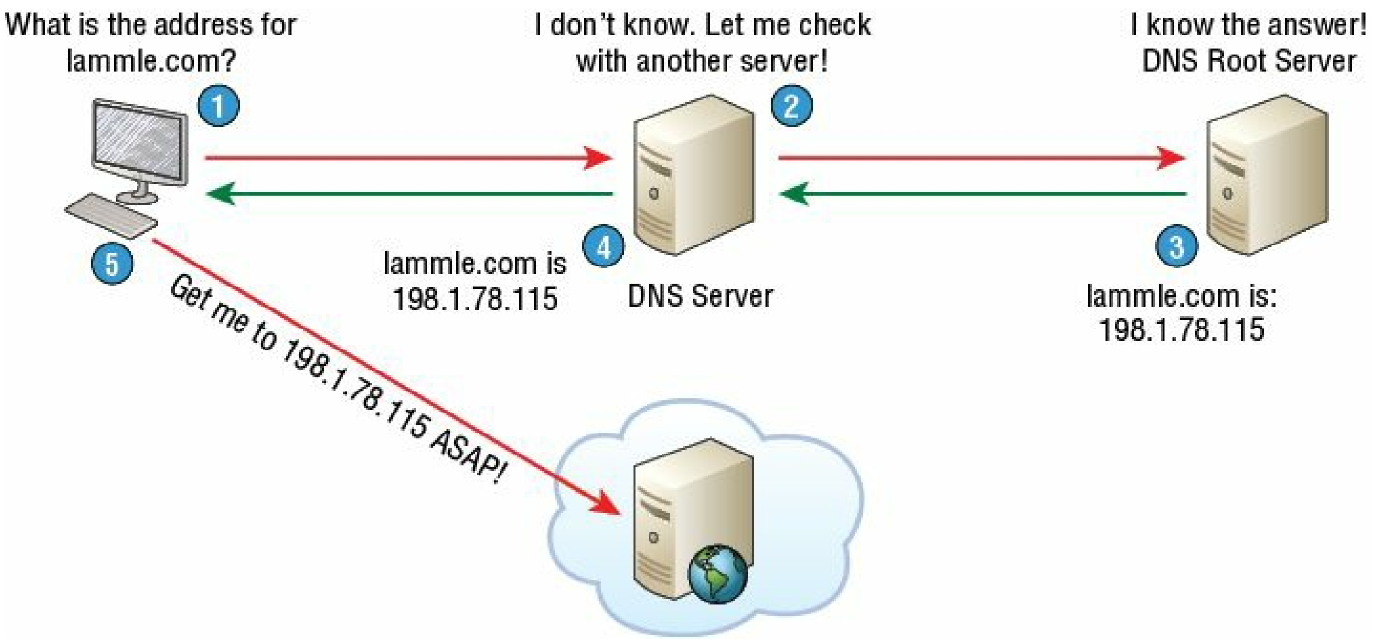

What is the address for lammle.com?
1
I don't know. Let me check with another server!
2
I know the answer! DNS Root Server
3
lammle.com is: 198.1.78.115
4
lammle.com is 198.1.78.115
DNS Server
5
Get me to 198.1.78.115 ASAP!

Figure 1.10 DNS

If your first DNS server doesn't know the answer to the query, then the DNS server forwards a TCP request to its root DNS server. Once the query is resolved, the answer is transmitted back to the originating host, which means the host can now request the information from the correct web server.

DNS is used to resolve a *fully qualified domain name (FQDN)*—for example, www.lammle.com or todd.lammle.com. An FQDN is a hierarchy that can logically locate a system based on its domain identifier.

If you want to resolve the name *todd*, you either must type in the FQDN of todd.lammle.com or have a device such as a PC or router add the suffix for you. For example, on a Cisco router, you can use the command ip domain-name lammle.com to append each request with the lammle.com domain. If you don't do that, you'll have to type in the FQDN to get DNS to resolve the name.

An important thing to remember about DNS is that if you can ping a device with an IP address but cannot use its FQDN, then you might have some type of DNS configuration failure.

Dynamic Host Configuration Protocol (DHCP)/Bootstrap Protocol (BootP)

Dynamic Host Configuration Protocol (DHCP) assigns IP addresses to hosts. It allows for easier administration and works well in small to very large network environments. Many types of hardware can be used as a DHCP server, including a Cisco router.

DHCP differs from BootP in that BootP assigns an IP address to a host but the host's hardware address must be entered manually in a BootP table. You can think of DHCP as a dynamic BootP. But remember that BootP is also used to send an operating system that a host can boot from. DHCP can't do that.

But there's still a lot of information a DHCP server can provide to a host when the host is requesting an IP address from the DHCP server. Here's a list of the most common types of information a DHCP server can provide:

- IP address
- Subnet mask
- Domain name
- Default gateway (routers)
- DNS server address
- WINS server address

A client that sends out a DHCP Discover message in order to receive an IP address sends out a broadcast at both layer 2 and layer 3.

- The layer 2 broadcast is all *F*s in hex, which looks like this: ff:ff:ff:ff:ff:ff.
- The layer 3 broadcast is 255.255.255.255, which means all networks and

all hosts.

DHCP is connectionless, which means it uses User Datagram Protocol (UDP) at the Transport layer, also known as the Host-to-Host layer, which we'll talk about later.

Seeing is believing, so here's an example of output from my analyzer showing the layer 2 and layer 3 broadcasts:

```
Ethernet II, Src: 0.0.0.0 (00:0b:db:99:d3:5e),Dst:
Broadcast(ff:ff:ff:ff:ff:ff)

Internet Protocol, Src: 0.0.0.0 (0.0.0.0),Dst:
255.255.255.255(255.255.255.255)
```

The Data Link and Network layers are both sending out "all hands" broadcasts saying, "Help—I don't know my IP address!"

Figure 1.11 shows the process of a client/server relationship using a DHCP connection.

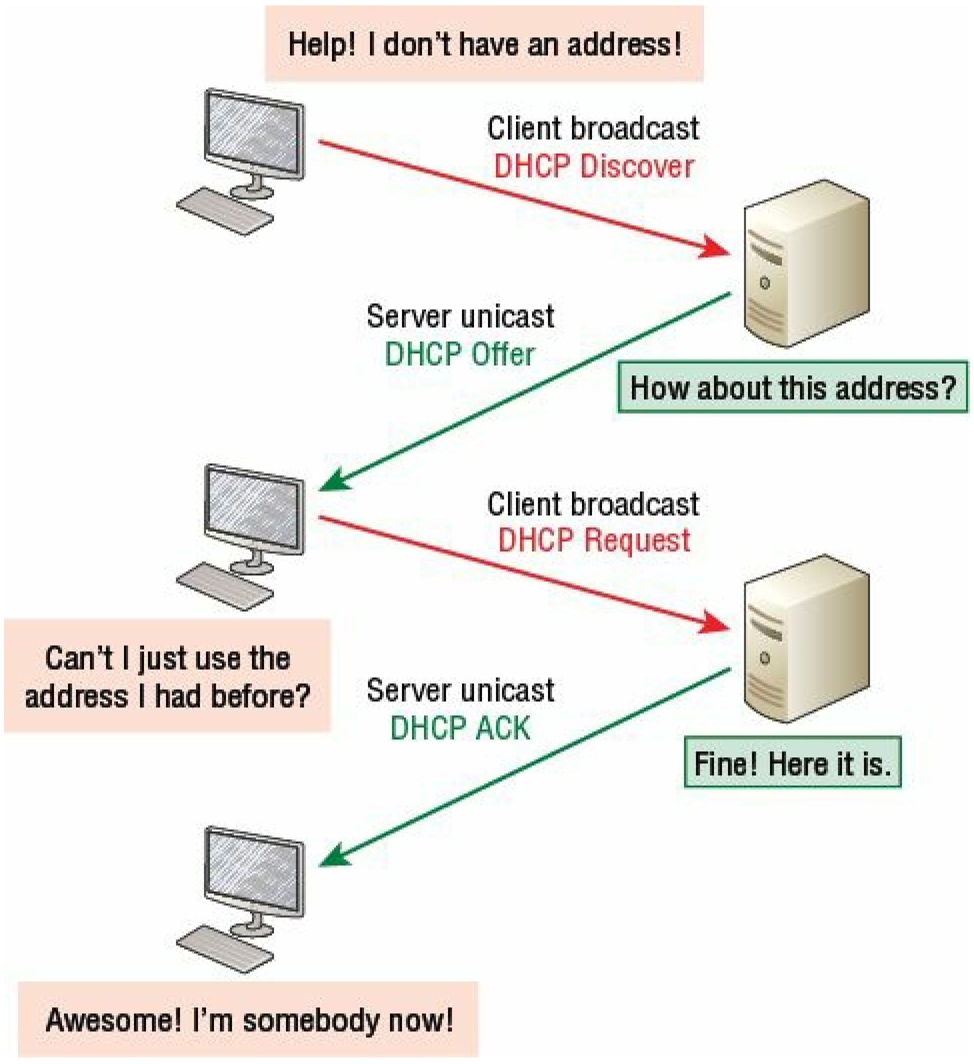


Figure 1.11 DHCP client four-step process

This is the four-step process a client takes to receive an IP address from a DHCP server:

1. The DHCP client broadcasts a DHCP Discover message looking for a DHCP server (Port 67).
2. The DHCP server that received the DHCP Discover message sends a layer 2 unicast DHCP Offer message back to the host.
3. The client then broadcasts to the server a DHCP Request message asking for the offered IP address and possibly other information.
4. The server finalizes the exchange with a unicast DHCP Acknowledgment message.

DHCP Conflicts

A DHCP address conflict occurs when two hosts use the same IP address. This sounds bad, and it is! We'll never even have to discuss this problem once we get to the chapter on IPv6!

During IP address assignment, a DHCP server checks for conflicts using the Ping program to test the availability of the address before it's assigned from the pool. If no host replies, then the DHCP server assumes that the IP address is not already allocated. This helps the server know that it's providing a good address, but what about the host? To provide extra protection against that terrible IP conflict issue, the host can broadcast for its own address!

A host uses something called a gratuitous ARP to help avoid a possible duplicate address. The DHCP client sends an ARP broadcast out on the local LAN or VLAN using its newly assigned address to solve conflicts before they occur.

So, if an IP address conflict is detected, the address is removed from the DHCP pool (scope), and it's really important to remember that the address will not be assigned to a host until the administrator resolves the conflict by hand!

Automatic Private IP Addressing (APIPA)

Okay, so what happens if you have a few hosts connected together with a switch or hub and you don't have a DHCP server? You can add IP information by hand, known as *static IP addressing*, but later Windows operating systems provide a feature called Automatic Private IP Addressing

(APIPA). With APIPA, clients can automatically self-configure an IP address and subnet mask—basic IP information that hosts use to communicate—when a DHCP server isn't available. The IP address range for APIPA is 169.254.0.1 through 169.254.255.254. The client also configures itself with a default Class B subnet mask of 255.255.0.0.

But when you're in your corporate network working and you have a DHCP server running, and your host shows that it's using this IP address range, it means that either your DHCP client on the host is not working or the server is down or can't be reached due to some network issue. Believe me—I don't know anyone who's seen a host in this address range and has been happy about it!

Now, let's take a look at the Transport layer, or what the DoD calls the Host-to-Host layer.

The Host-to-Host or Transport Layer Protocols

The main purpose of the Host-to-Host layer is to shield the upper-layer applications from the complexities of the network. This layer says to the upper layer, "Just give me your data stream, with any instructions, and I'll begin the process of getting your information ready to send."

Coming up, I'll introduce you to the two protocols at this layer:

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

In addition, we'll look at some of the key host-to-host protocol concepts, as well as the port numbers.

NOTE

Remember, this is still considered layer 4, and Cisco really likes the way layer 4 can use acknowledgments, sequencing, and flow control.

Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) takes large blocks of information from an application and breaks them into segments. It numbers and sequences each segment so that the destination's TCP stack can put the segments back into the order the application intended. After these segments are sent on the transmitting host, TCP waits for an acknowledgment of the receiving end's TCP virtual circuit session, retransmitting any segments that aren't acknowledged.

Before a transmitting host starts to send segments down the model, the sender's TCP stack contacts the destination's TCP stack to establish a connection. This creates a *virtual circuit*, and this type of communication is known as *connection-oriented*. During this initial handshake, the two TCP layers also agree on the amount of information that's going to be sent before the recipient's TCP sends back an acknowledgment. With everything agreed upon in advance, the path is paved for reliable communication to take place.

TCP is a full-duplex, connection-oriented, reliable, and accurate protocol, but establishing all these terms and conditions, in addition to error checking, is no small task. TCP is very complicated, and so not surprisingly, it's costly in terms of network overhead. And since today's networks are much more reliable than those of yore, this added reliability is often unnecessary. Most programmers use TCP because it removes a lot of programming work, but for real-time video and VoIP, *User Datagram Protocol (UDP)* is often better because using it results in less overhead.

TCP Segment Format

Since the upper layers just send a data stream to the protocols in the Transport layers, I'll use Figure 1.12 to demonstrate how TCP segments a data stream and prepares it for the Internet layer. When the Internet layer receives the data stream, it routes the segments as packets through an internetwork. The segments are handed to the receiving host's Host-to-Host layer protocol, which rebuilds the data stream for the upper-layer applications or protocols.

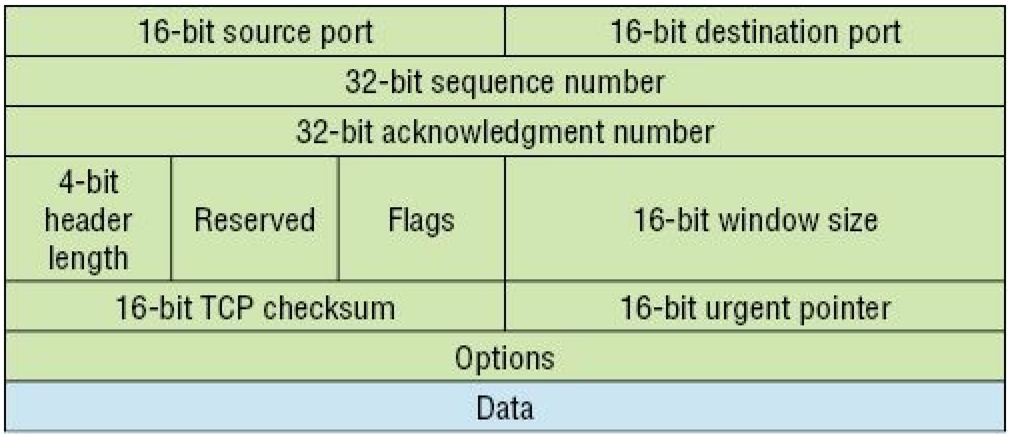


Figure 1.12 TCP segment format

Figure 1.12 shows the TCP segment format and shows the different fields within the TCP header. This isn't important to memorize for the Cisco exam objectives, but you need to understand it well because it's really good foundational information.

The TCP header is 20 bytes long, or up to 24 bytes with options. You need to understand what each field in the TCP segment is in order to build a strong educational foundation:

Source port This is the port number of the application on the host sending the data, which I'll talk about more thoroughly a little later in this chapter.

Destination port This is the port number of the application requested on the destination host.

Sequence number A number used by TCP that puts the data back in the correct order or retransmits missing or damaged data during a process called sequencing.

Acknowledgment number The value is the TCP octet that is expected next.

Header length The number of 32-bit words in the TCP header, which indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits in length.

Reserved Always set to zero.

Code bits/flags Controls functions used to set up and terminate a session.

Window The window size the sender is willing to accept, in octets.

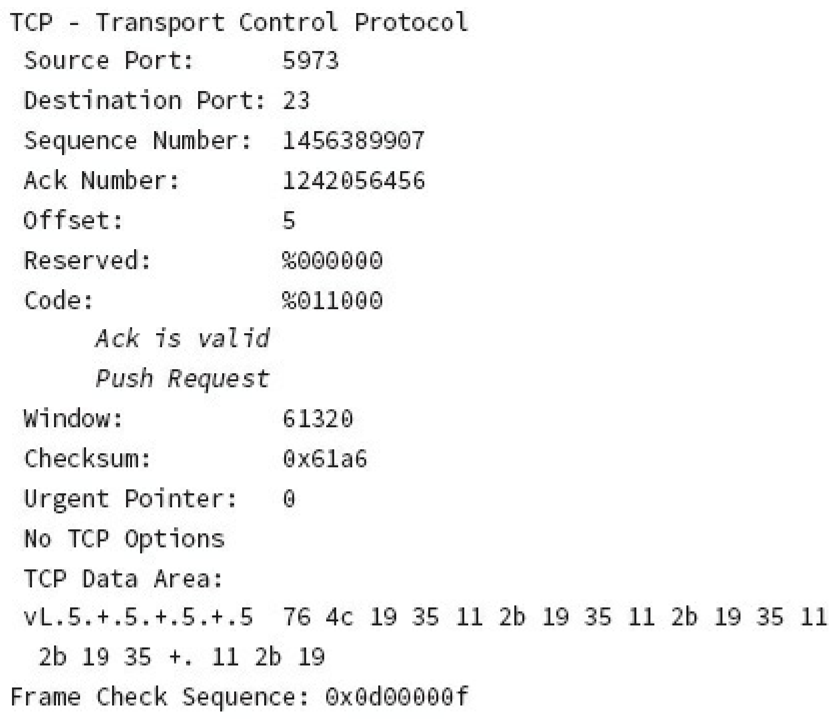
Checksum The cyclic redundancy check (CRC), used because TCP doesn't trust the lower layers and checks everything. The CRC checks the header and data fields.

Urgent A valid field only if the Urgent pointer in the code bits is set. If so, this value indicates the offset from the current sequence number, in octets, where the segment of non-urgent data begins.

Options May be 0, meaning that no options have to be present, or a multiple of 32 bits. However, if any options are used that do not cause the option field to total a multiple of 32 bits, padding of 0s must be used to make sure the data begins on a 32-bit boundary. These boundaries are known as words.

Data Handed down to the TCP protocol at the Transport layer, which includes the upper-layer headers.

Let's take a look at a TCP segment copied from a network analyzer:



```
TCP - Transport Control Protocol
 Source Port:       5973
 Destination Port: 23
 Sequence Number:   1456389907
 Ack Number:        1242056456
 Offset:            5
 Reserved:          %000000
 Code:              %011000
      Ack is valid
      Push Request
 Window:            61320
 Checksum:          0x61a6
 Urgent Pointer:    0
 No TCP Options
 TCP Data Area:
 vL.5.+.5.+.5.+.5  76 4c 19 35 11 2b 19 35 11 2b 19 35 11
  2b 19 35 +. 11 2b 19
Frame Check Sequence: 0x0d00000f
```

Did you notice that everything I talked about earlier is in the segment? As you can see from the number of fields in the header, TCP creates a lot of overhead. Again, this is why application developers may opt for efficiency over reliability to save overhead and go with UDP instead. It's also defined at the Transport layer as an alternative to TCP.

User Datagram Protocol (UDP)

User Datagram Protocol (UDP) is basically the scaled-down economy model of TCP, which is why UDP is sometimes referred to as a thin protocol. Like a thin person on a park bench, a thin protocol doesn't take up a lot of room—or in this case, require much bandwidth on a network.

UDP doesn't offer all the bells and whistles of TCP either, but it does do a fabulous job of transporting information that doesn't require reliable delivery, using far less network resources. (UDP is covered thoroughly in Request for Comments 768.)

So clearly, there are times that it's wise for developers to opt for UDP rather than TCP, one of them being when reliability is already taken care of at the Process/Application layer. Network File System (NFS) handles its own reliability issues, making the use of TCP both impractical and redundant. But ultimately, it's up to the application developer to opt for using UDP or TCP, not the user who wants to transfer data faster!

UDP does *not* sequence the segments and does not care about the order in which the segments arrive at the destination. UDP just sends the segments off and forgets about them. It doesn't follow through, check up on them, or even allow for an acknowledgment of safe arrival—complete abandonment. Because of this, it's referred to as an unreliable protocol. This does not mean that UDP is ineffective, only that it doesn't deal with reliability issues at all.

Furthermore, UDP doesn't create a virtual circuit, nor does it contact the destination before delivering information to it. Because of this, it's also considered a *connectionless* protocol. Since UDP assumes that the application will use its own reliability method, it doesn't use any itself. This presents an application developer with a choice when running the Internet Protocol stack: TCP for reliability or UDP for faster transfers.

It's important to know how this process works because if the segments arrive out of order, which is commonplace in IP networks, they'll simply be passed up to the next layer in whatever order they were received. This can result in some seriously garbled data! On the other hand, TCP sequences the segments so they get put back together in exactly the right order, which is something UDP just can't do.

UDP Segment Format

Figure 1.13 clearly illustrates UDP's markedly lean overhead as compared to TCP's hungry requirements. Look at the figure carefully—can you see that UDP doesn't use windowing or provide for acknowledgments in the UDP header?

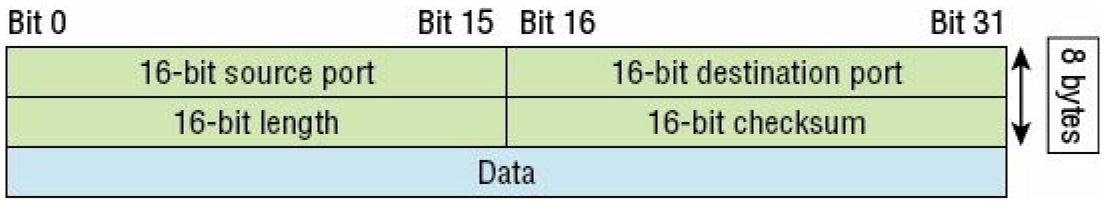


Figure 1.13 UDP segment

It's important for you to understand what each field in the UDP segment is:

Source port Port number of the application on the host sending the data

Destination port Port number of the application requested on the destination host

Length Length of UDP header and UDP data

Checksum Checksum of both the UDP header and UDP data fields

Data Upper-layer data

UDP, like TCP, doesn’t trust the lower layers and runs its own CRC. Remember that the Frame Check Sequence (FCS) is the field that houses the CRC, which is why you can see the FCS information.

The following shows a UDP segment caught on a network analyzer:

```
UDP - User Datagram Protocol

  Source Port:       1085

  Destination Port: 5136

  Length:            41

  Checksum:          0x7a3c

  UDP Data Area:

  ..Z......00 01 5a 96 00 01 00 00 00 00 00 11 0000 00

...C..2._C._C  2e 03 00 43 02 1e 32 0a 00 0a 00 80 43 00 80

Frame Check Sequence: 0x00000000
```

Notice that low overhead! Try to find the sequence number, ack number, and window size in the UDP segment. You can’t because they just aren’t there!

Key Concepts of Host-to-Host Protocols

Since you’ve now seen both a connection-oriented (TCP) and connectionless (UDP) protocol in action, it’s a good time to summarize the two here. Table 1.1highlights some of the key concepts about these two protocols for you to memorize.

Table 1.1 Key features of TCP and UDP

TCP	**UDP**
Sequenced	Unsequenced
Reliable	Unreliable

Connection-oriented	Connectionless
Virtual circuit	Low overhead
Acknowledgments	No acknowledgment
Windowing flow control	No windowing or flow control of any type

And if all this isn't quite clear yet, a telephone analogy will really help you understand how TCP works. Most of us know that before you speak to someone on a phone, you must first establish a connection with that other person no matter where they are. This is akin to establishing a virtual circuit with the TCP protocol. If you were giving someone important information during your conversation, you might say things like, "You know? or "Did you get that?" Saying things like this is a lot like a TCP acknowledgment—it's designed to get you verification. From time to time, especially on mobile phones, people ask, "Are you still there?" People end their conversations with a "Goodbye" of some kind, putting closure on the phone call, which you can think of as tearing down the virtual circuit that was created for your communication session. TCP performs these types of functions.

Conversely, using UDP is more like sending a postcard. To do that, you don't need to contact the other party first, you simply write your message, address the postcard, and send it off. This is analogous to UDP's connectionless orientation. Since the message on the postcard is probably not a matter of life or death, you don't need an acknowledgment of its receipt. Similarly, UDP does not involve acknowledgments.

Let's take a look at another figure, one that includes TCP, UDP, and the applications associated to each protocol: Figure 1.14 (discussed in the next section).

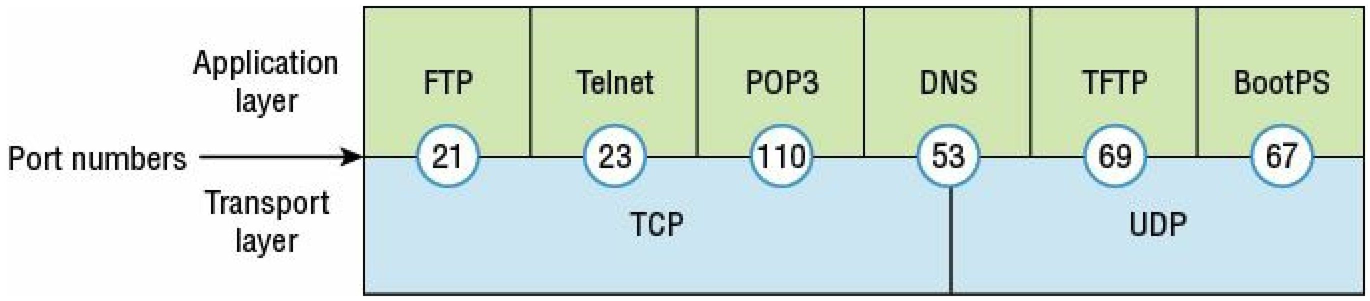


Figure 1.14 Port numbers for TCP and UDP

Port Numbers

TCP and UDP must use *port numbers* to communicate with the upper layers because these are what keep track of different conversations crossing the network simultaneously. Originating-source port numbers are dynamically assigned by the source host and will equal some number starting at 1024. Port number 1023 and below are defined in RFC 3232 (or just see www.iana.org), which discusses what we call well-known port numbers.

Virtual circuits that don't use an application with a well-known port number are assigned port numbers randomly from a specific range instead. These port numbers identify the source and destination application or process in the TCP segment.

NOTE

The Requests for Comments (RFCs) form a series of notes about the Internet (originally the ARPAnet) started in 1969. These notes discuss many aspects of computer communication, focusing on networking protocols, procedures, programs, and concepts, but they also include meeting notes, opinions, and sometimes even humor. You can find the RFCs by visiting www.iana.org.

Figure 1.14 illustrates how both TCP and UDP use port numbers. I'll cover the different port numbers that can be used next:

- Numbers below 1024 are considered well-known port numbers and are defined in RFC 3232.
- Numbers 1024 and above are used by the upper layers to set up sessions with other hosts and by TCP and UDP to use as source and destination addresses in the segment.

TCP Session: Source Port

Let's take a minute to check out analyzer output showing a TCP session I captured with my analyzer software session now:

```
TCP - Transport Control Protocol
 Source Port:      5973
 Destination Port: 23
 Sequence Number:  1456389907
 Ack Number:       1242056456
 Offset:           5
 Reserved:         %000000
 Code:             %011000
      Ack is valid
      Push Request
 Window:           61320
 Checksum:         0x61a6
 Urgent Pointer:   0
 No TCP Options
 TCP Data Area:
 vL.5.+.5.+.5.+.5  76 4c 19 35 11 2b 19 35 11 2b 19 35 11
  2b 19 35 +. 11 2b 19
Frame Check Sequence: 0x0d00000f
```

Notice that the source host makes up the source port, which in this case is 5973. The destination port is 23, which is used to tell the receiving host the purpose of the intended connection (Telnet).

By looking at this session, you can see that the source host makes up the source port by using numbers from 1024 to 65535. But why does the source make up a port number? To differentiate between sessions with different hosts because how would a server know where information is coming from if it didn't have a different number from a sending host? TCP and the upper layers don't use hardware and logical addresses to understand the sending host's address as the Data Link and Network layer protocols do. Instead, they use port numbers.

TCP Session: Destination Port

You'll sometimes look at an analyzer and see that only the source port is above 1024 and the destination port is a well-known port, as shown in the following trace:

```
TCP - Transport Control Protocol
 Source Port:      1144
 Destination Port: 80 World Wide Web HTTP
 Sequence Number:  9356570
 Ack Number:       0
 Offset:          7
 Reserved:        %000000
 Code:            %000010
      Synch Sequence
 Window:          8192
 Checksum:        0x57E7
 Urgent Pointer:  0
 TCP Options:
  Option Type: 2 Maximum Segment Size
    Length:    4
    MSS:       536
  Option Type: 1 No Operation
  Option Type: 1 No Operation
  Option Type: 4
    Length:    2
    Opt Value:
  No More HTTP Data
Frame Check Sequence: 0x43697363
```

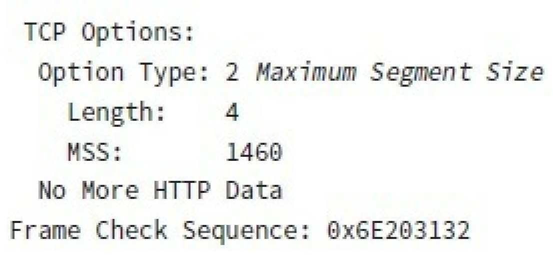
And sure enough, the source port is over 1024, but the destination port is 80, indicating an HTTP service. The server, or receiving host, will change the destination port if it needs to.

In the preceding trace, a “SYN” packet is sent to the destination device. This Synch (as shown in the output) sequence is what’s used to inform the remote destination device that it wants to create a session.

TCP Session: Syn Packet Acknowledgment

The next trace shows an acknowledgment to the SYN packet:

```
TCP - Transport Control Protocol
 Source Port:      80 World Wide Web HTTP
 Destination Port: 1144
 Sequence Number:  2873580788
 Ack Number:       9356571
 Offset:           6
 Reserved:         %000000
 Code:             %010010
      Ack is valid
      Synch Sequence
 Window:           8576
 Checksum:         0x5F85
 Urgent Pointer:   0
```



```
TCP Options:
  Option Type: 2 Maximum Segment Size
    Length:    4
    MSS:       1460
  No More HTTP Data
Frame Check Sequence: 0x6E203132
```

Notice the *Ack is valid*, which means that the source port was accepted and the device agreed to create a virtual circuit with the originating host.

And here again, you can see that the response from the server shows that the source is 80 and the destination is the 1144 sent from the originating host—all’s well!

Table 1.2 gives you a list of the typical applications used in the TCP/IP suite by showing their well-known port numbers and the Transport layer protocols used by each application or process. It’s really key to memorize this table.

Table 1.2 Key protocols that use TCP and UDP

TCP	**UDP**
Telnet 23	SNMP 161
SMTP 25	TFTP 69
HTTP 80	DNS 53
FTP 20, 21	BooTPS/DHCP 67
DNS 53	
HTTPS 443	NTP 123
SSH 22	
POP3 110	
IMAP4 143	

Notice that DNS uses both TCP and UDP. Whether it opts for one or the other depends on what it’s trying to do. Even though it’s not the only application that can use both protocols, it’s certainly one that you should make sure to remember in your studies.

NOTE

What makes TCP reliable is sequencing, acknowledgments, and flow control (windowing). UDP does not have reliability.

Okay—I want to discuss one more item before we move down to the Internet layer—session multiplexing. Session multiplexing is used by both TCP and UDP and basically allows a single computer, with a single IP address, to have multiple sessions occurring simultaneously. Say you go to www.lammle.com and are browsing and then you click a link to another page. Doing this opens another session to your host. Now you go to www.lammle.com/forum from another window and that site opens a window as well. Now you have three sessions open using one IP address because the Session layer is sorting the separate requests based on the Transport layer port number. This is the job of the Session layer: to keep application layer data separate!

The Internet Layer Protocols

In the DoD model, there are two main reasons for the Internet layer’s existence: routing and providing a single network interface to the upper layers.

None of the other upper- or lower-layer protocols have any functions relating to routing—that complex and important task belongs entirely to the Internet layer. The Internet layer’s second duty is to provide a single network interface to the upper-layer protocols. Without this layer, application programmers would need to write “hooks” into every one of their applications for each different Network Access protocol. This would not only be a pain in the neck, but it would lead to different versions of each application—one for Ethernet, another one for wireless, and so on. To prevent this, IP provides one single network interface for the upper-layer protocols. With that mission accomplished, it’s then the job of IP and the various Network Access protocols to get along and work together.

All network roads don't lead to Rome—they lead to IP. And all the other protocols at this layer, as well as all those at the upper layers, use it. Never forget that. All paths through the DoD model go through IP. Here's a list of the important protocols at the Internet layer that I'll cover individually in detail coming up:

- Internet Protocol (IP)
- Internet Control Message Protocol (ICMP)
- Address Resolution Protocol (ARP)

Internet Protocol (IP)

Internet Protocol (IP) essentially is the Internet layer. The other protocols found here merely exist to support it. IP holds the big picture and could be said to "see all," because it's aware of all the interconnected networks. It can do this because all the machines on the network have a software, or logical, address called an IP address, which we'll explore more thoroughly later in this chapter.

For now, understand that IP looks at each packet's address. Then, using a routing table, it decides where a packet is to be sent next, choosing the best path to send it upon. The protocols of the Network Access layer at the bottom of the DoD model don't possess IP's enlightened scope of the entire network; they deal only with physical links (local networks).

Identifying devices on networks requires answering these two questions: Which network is it on? And what is its ID on that network? The first answer is the *software address*, or *logical address*. You can think of this as the part of the address that specifies the correct street. The second answer is the hardware address, which goes a step further to specify the correct mailbox. All hosts on a network have a logical ID called an IP address. This is the software, or logical, address and contains valuable encoded information, greatly simplifying the complex task of routing. (IP is discussed in RFC 791.)

IP receives segments from the Host-to-Host layer and fragments them into datagrams (packets) if necessary. IP then reassembles datagrams back into segments on the receiving side. Each datagram is assigned the IP address of the sender and that of the recipient. Each router or switch (layer 3 device) that receives a datagram makes routing decisions based on the packet's destination IP address.

Figure 1.15 shows an IP header. This will give you a picture of what the IP protocol has to go through every time user data that is destined for a remote network is sent from the upper layers.

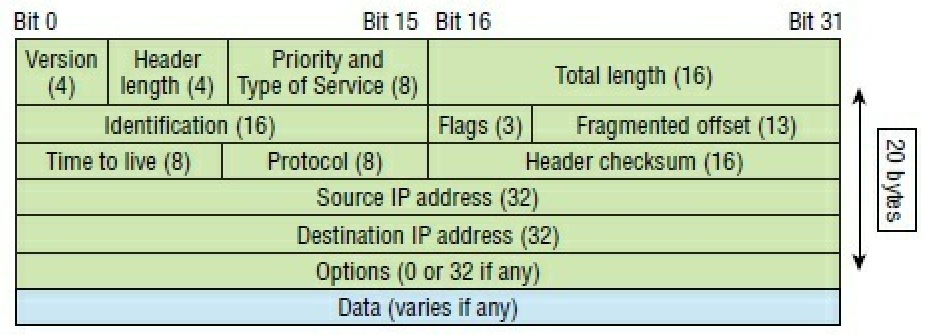


Figure 1.15 IP header

The following fields make up the IP header:

Version IP version number.

Header length Header length (HLEN) in 32-bit words.

Priority and Type of Service Type of Service tells how the datagram should be handled. The first 3 bits are the priority bits, now called the differentiated services bits.

Total length Length of the packet, including header and data.

Identification Unique IP-packet value used to differentiate fragmented packets from different datagrams.

Flags Specifies whether fragmentation should occur.

Fragment offset Provides fragmentation and reassembly if the packet is too large to put in a frame. It also allows different maximum transmission units (MTUs) on the Internet.

Time To Live The time to live (TTL) is set into a packet when it is originally generated. If it doesn't get to where it's supposed to go before the TTL expires, boom—it's gone. This stops IP packets from continuously circling the network looking for a home.

Protocol Port of upper-layer protocol; for example, TCP is port 6 or UDP is port 17. Also supports Network layer protocols, like ARP and ICMP, and can be referred to as the Type field in some analyzers. We'll talk about this field more in a minute.

Header checksum Cyclic redundancy check (CRC) on header only.

Source IP address 32-bit IP address of sending station.

Destination IP address 32-bit IP address of the station this packet is destined for.

Options Used for network testing, debugging, security, and more.

Data After the IP option field, will be the upper-layer data.

Here's a snapshot of an IP packet caught on a network analyzer. Notice that all the header information discussed previously appears here:

```
IP Header - Internet Protocol Datagram

 Version:              4

 Header Length:        5

 Precedence:           0

 Type of Service:      %000

 Unused:               %00

 Total Length:         187

 Identifier:           22486

 Fragmentation Flags: %010 Do Not Fragment

 Fragment Offset:      0

 Time To Live:         60

 IP Type:              0x06 TCP

 Header Checksum:      0xd031

 Source IP Address:    10.7.1.30

 Dest. IP Address:     10.7.1.10

 No Internet Datagram Options
```

The Type field is typically a Protocol field, but this analyzer sees it as an IP Type field. This is important. If the header didn't carry the protocol information for the next layer, IP wouldn't know what to do with the data carried in the packet. The preceding example clearly tells IP to hand the segment to TCP.

Figure 1.16 demonstrates how the Network layer sees the protocols at the Transport layer when it needs to hand a packet up to the upper-layer protocols.

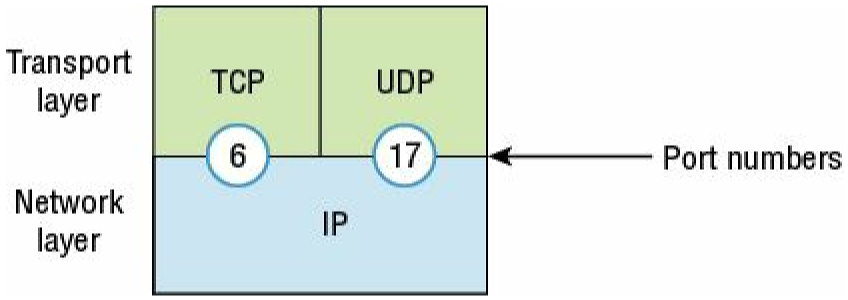


Figure 1.16 The Protocol field in an IP header

In this example, the Protocol field tells IP to send the data to either TCP port 6 or UDP port 17. But it will be UDP or TCP only if the data is part of a data stream headed for an upper-layer service or application. It could just as easily be destined for Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), or some other type of Network layer protocol.

Table 1.3 is a list of some other popular protocols that can be specified in the Protocol field.

Table 1.3 Possible protocols found in the Protocol field of an IP header

Protocol	**Protocol Number**
ICMP	1
IP in IP (tunneling)	4
TCP	6
UDP	17
EIGRP	88
OSPF	89
IPv6	41
GRE	47
Layer 2 tunnel (L2TP)	115

NOTE

You can find a complete list of Protocol field numbers at www.iana.org/assignments/protocol-numbers.

Internet Control Message Protocol (ICMP)

Internet Control Message Protocol (ICMP) works at the Network layer and is used by IP for many different services. ICMP is basically a management protocol and messaging service provider for IP. Its messages are carried as IP datagrams. RFC 1256 is an annex to ICMP, which gives hosts extended capability in discovering routes to gateways.

ICMP packets have the following characteristics:

- They can provide hosts with information about network problems.
- They are encapsulated within IP datagrams.

The following are some common events and messages that ICMP relates to:

Destination unreachable If a router can't send an IP datagram any further, it uses ICMP to send a message back to the sender, advising it of the situation. For example, take a look at Figure 1.17, which shows that interface e0 of the Lab_B router is down.

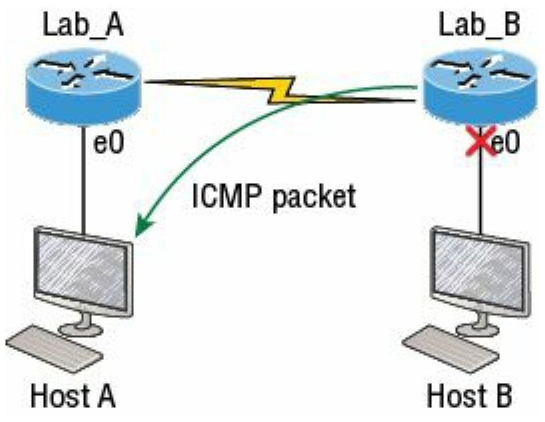


Figure 1.17 ICMP error message is sent to the sending host from the remote router.

When Host A sends a packet destined for Host B, the Lab_B router will send an ICMP destination unreachable message back to the sending device, which is Host A in this example.

Buffer full/source quench If a router's memory buffer for receiving incoming datagrams is full, it will use ICMP to send out this message alert until the congestion abates.

Hops/time exceeded Each IP datagram is allotted a certain number of routers, called hops, to pass through. If it reaches its limit of hops before arriving at its destination, the last router to receive that datagram deletes it. The executioner router then uses ICMP to send an obituary message, informing the sending machine of the demise of its datagram.

Ping Packet Internet Groper (Ping) uses ICMP echo request and reply messages to check the physical and logical connectivity of machines on an internetwork.

Traceroute Using ICMP time-outs, Traceroute is used to discover the path a packet takes as it traverses an internetwork.

NOTE

Traceroute is usually just called trace. Microsoft Windows uses tracert to allow you to verify address configurations in your internetwork.

The following data is from a network analyzer catching an ICMP echo request:

```
Flags:          0x00
  Status:         0x00
  Packet Length: 78
  Timestamp:      14:04:25.967000 12/20/03
Ethernet Header
  Destination: 00:a0:24:6e:0f:a8
  Source:      00:80:c7:a8:f0:3d
  Ether-Type:  08-00 IP
IP Header - Internet Protocol Datagram
  Version:             4
  Header Length:       5
  Precedence:          0
  Type of Service:     %000
  Unused:              %00
  Total Length:        60
  Identifier:          56325
  Fragmentation Flags: %000
  Fragment Offset:     0
  Time To Live:        32
  IP Type:             0x01 ICMP
  Header Checksum:     0x2df0
  Source IP Address:   100.100.100.2
  Dest. IP Address:    100.100.100.1
  No Internet Datagram Options
ICMP - Internet Control Messages Protocol
  ICMP Type:       8 Echo Request
  Code:            0
  Checksum:        0x395c
  Identifier:      0x0300
  Sequence Number: 4352
  ICMP Data Area:
  abcdefghijklmnop  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70
  qrstuvwabcdefghi  71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69
Frame Check Sequence: 0x00000000
```

Notice anything unusual? Did you catch the fact that even though ICMP works at the Internet (Network) layer, it still uses IP to do the Ping request? The Type field in the IP header is 0x01, which specifies that the data we're carrying is owned by the ICMP protocol. Remember, just as all roads lead to

Rome, all segments or data *must* go through IP!

NOTE

The Ping program uses the alphabet in the data portion of the packet as a payload, typically around 100 bytes by default, unless, of course, you are pinging from a Windows device, which thinks the alphabet stops at the letter *W* (and doesn't include *X*, *Y*, or *Z*) and then starts at *A* again. Go figure!

The only fields are destination hardware address, source hardware address, and Ether-Type. The only frame that uses an Ether-Type field exclusively is an Ethernet_II frame.

We'll move on soon, but before we get into the ARP protocol, let's take another look at ICMP in action. Figure 1.18 shows an internetwork—it has a router, so it's an internetwork, right?

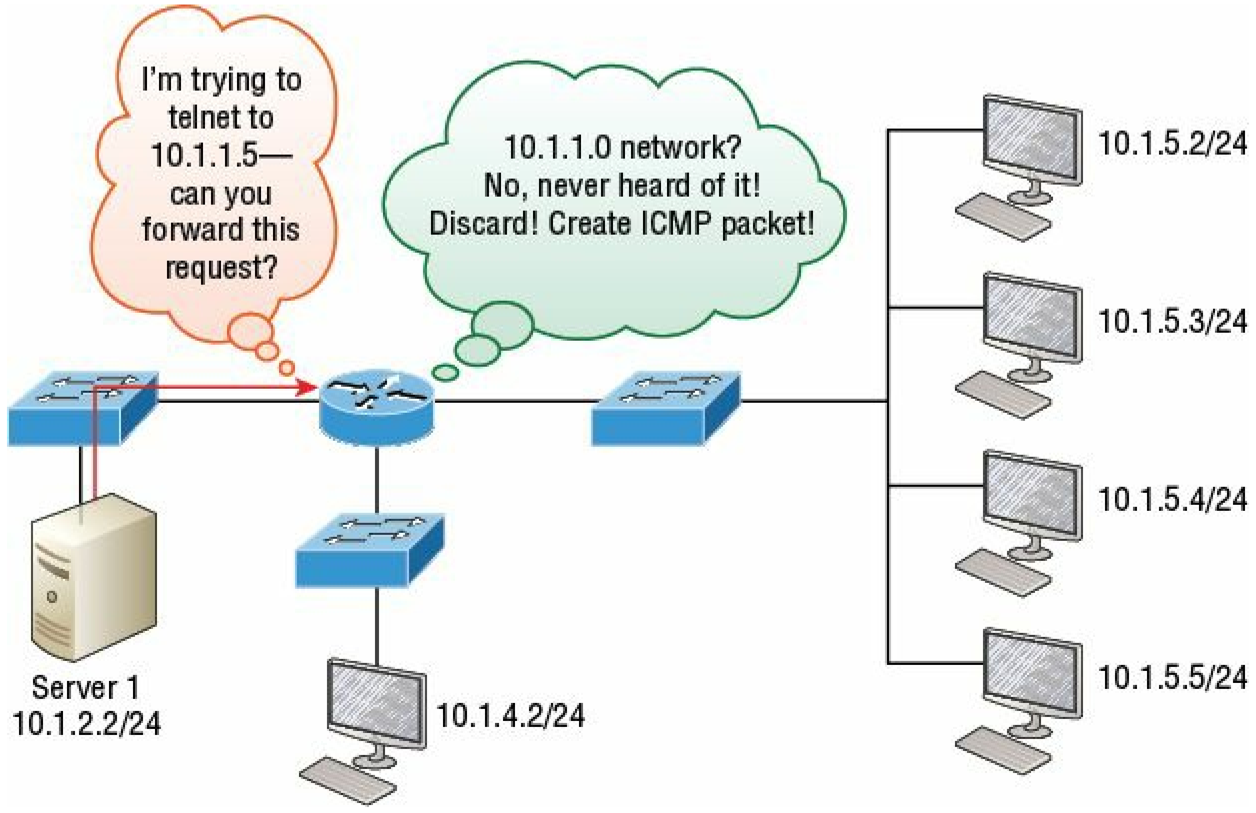

I'm trying to telnet to 10.1.1.5—can you forward this request?
10.1.1.0 network? No, never heard of it! Discard! Create ICMP packet!
10.1.5.2/24
10.1.5.3/24
10.1.5.4/24
10.1.5.5/24
Server 1 10.1.2.2/24
10.1.4.2/24

Figure 1.18 ICMP in action

Server 1 (10.1.2.2) telnets to 10.1.1.5 from a DOS prompt. What do you think

Server 1 will receive as a response? Server 1 will send the Telnet data to the default gateway, which is the router, and the router will drop the packet because there isn't a network 10.1.1.0 in the routing table. Because of this, Server 1 will receive an ICMP destination unreachable back from the router.

Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) finds the hardware address of a host from a known IP address. Here's how it works: When IP has a datagram to send, it must inform a Network Access protocol, such as Ethernet or wireless, of the destination's hardware address on the local network. Remember that it has already been informed by upper-layer protocols of the destination's IP address. If IP doesn't find the destination host's hardware address in the ARP cache, it uses ARP to find this information.

As IP's detective, ARP interrogates the local network by sending out a broadcast asking the machine with the specified IP address to reply with its hardware address. So basically, ARP translates the software (IP) address into a hardware address—for example, the destination machine's Ethernet adapter address—and from it, deduces its whereabouts on the LAN by broadcasting for this address. Figure 1.19 shows how an ARP broadcast looks to a local network.

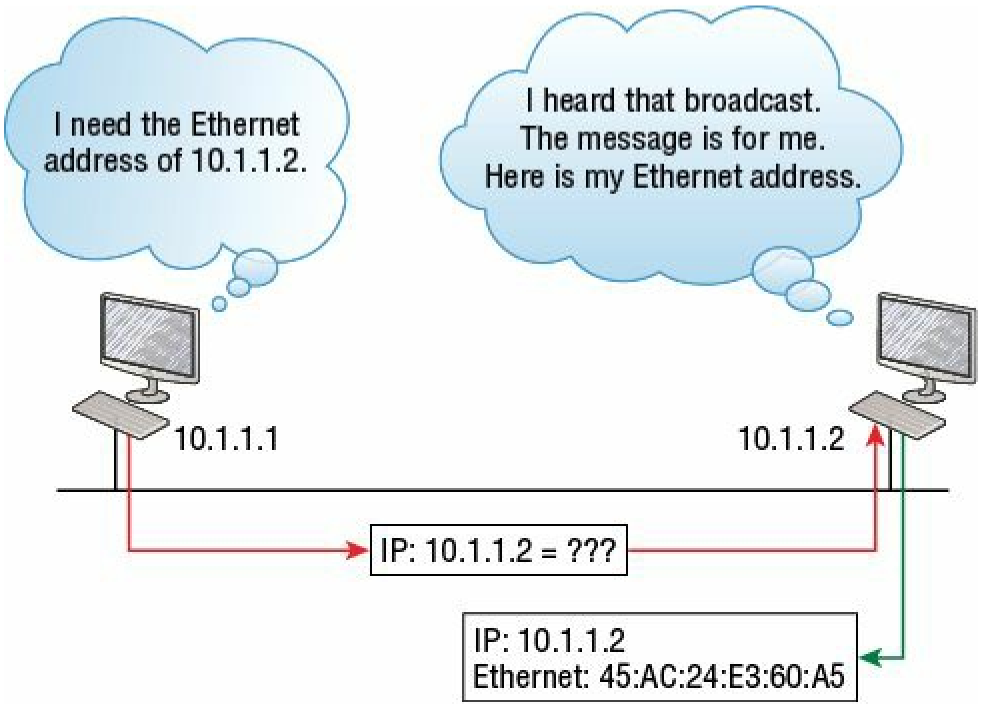

I need the Ethernet address of 10.1.1.2.
I heard that broadcast. The message is for me. Here is my Ethernet address.
10.1.1.1
10.1.1.2
IP: 10.1.1.2 = ???
IP: 10.1.1.2
Ethernet: 45:AC:24:E3:60:A5

Figure 1.19 Local ARP broadcast

NOTE

ARP resolves IP addresses to Ethernet (MAC) addresses.

The following trace shows an ARP broadcast—notice that the destination hardware address is unknown and is all *F*s in hex (all 1s in binary)—and is a hardware address broadcast:

```
  Flags:          0x00
  Status:         0x00
  Packet Length: 64
  Timestamp:      09:17:29.574000 12/06/03
Ethernet Header
  Destination:    FF:FF:FF:FF:FF:FF Ethernet Broadcast
  Source:         00:A0:24:48:60:A5
  Protocol Type: 0x0806 IP ARP
ARP - Address Resolution Protocol
  Hardware:                 1 Ethernet (10Mb)
  Protocol:                 0x0800 IP
  Hardware Address Length: 6
  Protocol Address Length: 4
  Operation:                1 ARP Request
  Sender Hardware Address: 00:A0:24:48:60:A5
  Sender Internet Address: 172.16.10.3
  Target Hardware Address: 00:00:00:00:00:00 (ignored)
  Target Internet Address: 172.16.10.10
Extra bytes (Padding):
  ................ 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
```

```
   0A 0A 0A 0A 0A

Frame Check Sequence: 0x00000000
```

IP Addressing

One of the most important topics in any discussion of TCP/IP is IP addressing. An *IP address* is a numeric identifier assigned to each machine on an IP network. It designates the specific location of a device on the network.

An IP address is a software address, not a hardware address—the latter is hard-coded on a network interface card (NIC) and used for finding hosts on a local network. IP addressing was designed to allow hosts on one network to communicate with a host on a different network regardless of the type of LANs the hosts are participating in.

Before we get into the more complicated aspects of IP addressing, you need to understand some of the basics. First I'm going to explain some of the fundamentals of IP addressing and its terminology. Then you'll learn about the hierarchical IP addressing scheme and private IP addresses.

IP Terminology

Throughout this chapter you're being introduced to several important terms that are vital to understanding the Internet Protocol. Here are a few to get you started:

Bit A bit is one digit, either a 1 or a 0.

Byte A byte is 7 or 8 bits, depending on whether parity is used. For the rest of this chapter, always assume a byte is 8 bits.

Octet An octet, made up of 8 bits, is just an ordinary 8-bit binary number. In this chapter, the terms *byte* and *octet* are completely interchangeable.

Network address This is the designation used in routing to send packets to a remote network—for example, 10.0.0.0, 172.16.0.0, and 192.168.10.0.

Broadcast address The address used by applications and hosts to send information to all nodes on a network is called the broadcast address. Examples of layer 3 broadcasts include 255.255.255.255, which is any network, all nodes; 172.16.255.255, which is all subnets and hosts on network 172.16.0.0; and 10.255.255.255, which broadcasts to all subnets and hosts on network 10.0.0.0.

The Hierarchical IP Addressing Scheme

An IP address consists of 32 bits of information. These bits are divided into four sections, referred to as octets or bytes, with each containing 1 byte (8 bits). You can depict an IP address using one of three methods:

- Dotted-decimal, as in 172.16.30.56
- Binary, as in 10101100.00010000.00011110.00111000
- Hexadecimal, as in AC.10.1E.38

All these examples represent the same IP address. Pertaining to IP addressing, hexadecimal isn't used as often as dotted-decimal or binary, but you still might find an IP address stored in hexadecimal in some programs.

The 32-bit IP address is a structured or hierarchical address, as opposed to a flat or nonhierarchical address. Although either type of addressing scheme could have been used, *hierarchical addressing* was chosen for a good reason. The advantage of this scheme is that it can handle a large number of addresses, namely 4.3 billion (a 32-bit address space with two possible values for each position—either 0 or 1—gives you 2^{32}, or 4,294,967,296). The disadvantage of the flat addressing scheme, and the reason it's not used for IP addressing, relates to routing. If every address were unique, all routers on the Internet would need to store the address of each and every machine on the Internet. This would make efficient routing impossible, even if only a fraction of the possible addresses were used!

The solution to this problem is to use a two- or three-level hierarchical addressing scheme that is structured by network and host or by network, subnet, and host.

This two- or three-level scheme can also be compared to a telephone number. The first section, the area code, designates a very large area. The second section, the prefix, narrows the scope to a local calling area. The final segment, the customer number, zooms in on the specific connection. IP addresses use the same type of layered structure. Rather than all 32 bits being treated as a unique identifier, as in flat addressing, a part of the address is designated as the network address and the other part is designated as either the subnet and host or just the node address.

Next, we'll cover IP network addressing and the different classes of address we can use to address our networks.

Network Addressing

The *network address* (which can also be called the network number) uniquely identifies each network. Every machine on the same network shares that network address as part of its IP address. For example, in the IP address 172.16.30.56, 172.16 is the network address.

The *node address* is assigned to, and uniquely identifies, each machine on a network. This part of the address must be unique because it identifies a particular machine—an individual—as opposed to a network, which is a group. This number can also be referred to as a *host address*. In the sample IP address 172.16.30.56, the 30.56 specifies the node address.

The designers of the Internet decided to create classes of networks based on network size. For the small number of networks possessing a very large number of nodes, they created the rank *Class A network*. At the other extreme is the *Class C network*, which is reserved for the numerous networks with a small number of nodes. The class distinction for networks between very large and very small is predictably called the *Class B network*.

Subdividing an IP address into a network and node address is determined by the class designation of one's network. Figure 1.20 summarizes the three classes of networks used to address hosts—a subject I'll explain in much greater detail throughout this chapter.

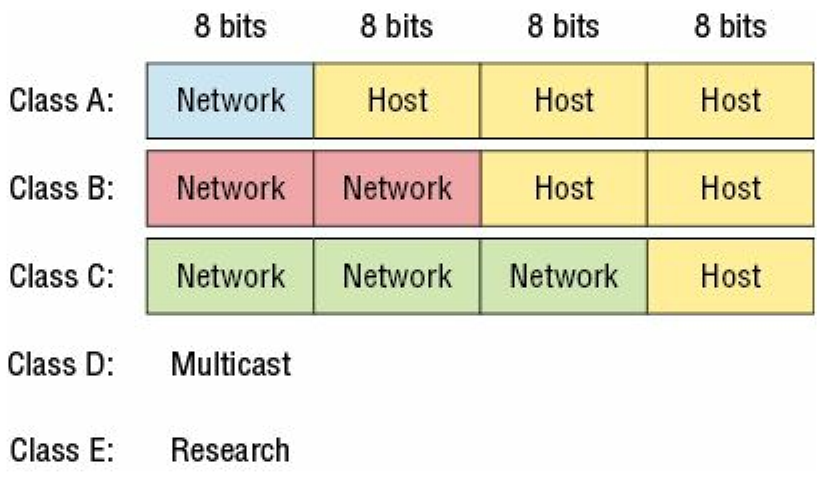


Figure 1.20 Summary of the three classes of networks

To ensure efficient routing, Internet designers defined a mandate for the leading-bits section of the address for each different network class. For example, since a router knows that a Class A network address always starts with a 0, the router might be able to speed a packet on its way after reading only the first bit of its address. This is where the address schemes define the difference between a Class A, a Class B, and a Class C address. Coming up, I'll discuss the differences between these three classes, followed by a discussion of the Class D and Class E addresses. Classes A, B, and C are the only ranges that are used to address hosts in our networks.

Network Address Range: Class A

The designers of the IP address scheme decided that the first bit of the first byte in a Class A network address must always be off, or 0. This means a Class A address must be between 0 and 127 in the first byte, inclusive.

Consider the following network address:

```
0 xxxxxxx
```

If we turn the other 7 bits all off and then turn them all on, we'll find the Class A range of network addresses:

```
00000000 = 0

01111111 = 127
```

So, a Class A network is defined in the first octet between 0 and 127, and it can't be less or more. Understand that 0 and 127 are not valid in a Class A network because they're reserved addresses, which I'll explain soon.

Network Address Range: Class B

In a Class B network, the RFCs state that the first bit of the first byte must always be turned on but the second bit must always be turned off. If you turn the other 6 bits all off and then all on, you will find the range for a Class B network:

```
10000000 = 128

10111111 = 191
```

As you can see, a Class B network is defined when the first byte is configured from 128 to 191.

Network Address Range: Class C

For Class C networks, the RFCs define the first 2 bits of the first octet as always turned on, but the third bit can never be on. Following the same process as the previous classes, convert from binary to decimal to find the range. Here's the range for a Class C network:

```
11000000 = 192

11011111 = 223
```

So, if you see an IP address that starts at 192 and goes to 223, you'll know it is a Class C IP address.

Network Address Ranges: Classes D and E

The addresses between 224 to 255 are reserved for Class D and E networks. Class D (224–239) is used for multicast addresses and Class E (240–255) for scientific purposes, but I'm not going into these types of addresses because they are beyond the scope of knowledge you need to gain from this book.

Network Addresses: Special Purpose

Some IP addresses are reserved for special purposes, so network administrators can't ever assign these addresses to nodes. Table 1.4lists the members of this exclusive little club and the reasons why they're included in it.

Table 1.4 Reserved IP addresses

Address	**Function**
Network address of all 0s	Interpreted to mean "this network or segment."
Network address of all 1s	Interpreted to mean "all networks."
Network 127.0.0.1	Reserved for loopback tests. Designates the local node and allows that node to send a test packet to itself without generating network traffic.
Node address of all 0s	Interpreted to mean "network address" or any host on a specified network.
Node address of all 1s	Interpreted to mean "all nodes" on the specified network; for example, 128.2.255.255 means "all nodes" on network 128.2 (Class B address).
Entire IP address set to all 0s	Used by Cisco routers to designate the default route. Could also mean "any network."

Entire IP address set to all 1s (same as 255.255.255.255)	Broadcast to all nodes on the current network; sometimes called an “all 1s broadcast” or local broadcast.

Class A Addresses

In a Class A network address, the first byte is assigned to the network address and the three remaining bytes are used for the node addresses. The Class A format is as follows:

network.node.node.node

For example, in the IP address 49.22.102.70, the 49 is the network address and 22.102.70 is the node address. Every machine on this particular network would have the distinctive network address of 49.

Class A network addresses are 1 byte long, with the first bit of that byte reserved and the 7 remaining bits available for manipulation (addressing). As a result, the maximum number of Class A networks that can be created is 128. Why? Because each of the 7 bit positions can be either a 0 or a 1, thus 2^7, or 128.

To complicate matters further, the network address of all 0s (0000 0000) is reserved to designate the default route (see Table 1.4in the previous section). Additionally, the address 127, which is reserved for diagnostics, can’t be used either, which means that you can really only use the numbers 1 to 126 to designate Class A network addresses. This means the actual number of usable Class A network addresses is 128 minus 2, or 126.

The IP address 127.0.0.1 is used to test the IP stack on an individual node and cannot be used as a valid host address. However, the loopback address creates a shortcut method for TCP/IP applications and services that run on the same device to communicate with each other.

Each Class A address has 3 bytes (24-bit positions) for the node address of a machine. This means there are 2^{24}—or 16,777,216—unique combinations and, therefore, precisely that many possible unique node addresses for each

Class A network. Because node addresses with the two patterns of all 0s and all 1s are reserved, the actual maximum usable number of nodes for a Class A network is 2^{24} minus 2, which equals 16,777,214. Either way, that's a huge number of hosts on a single network segment!

Class A Valid Host IDs

Here's an example of how to figure out the valid host IDs in a Class A network address:

- All host bits off is the network address: 10.0.0.0.
- All host bits on is the broadcast address: 10.255.255.255.

The valid hosts are the numbers in between the network address and the broadcast address: 10.0.0.1 through 10.255.255.254. Notice that 0s and 255s can be valid host IDs. All you need to remember when trying to find valid host addresses is that the host bits can't all be turned off or on at the same time.

Class B Addresses

In a Class B network address, the first 2 bytes are assigned to the network address and the remaining 2 bytes are used for node addresses. The format is as follows:

```
network.network.node.node
```

For example, in the IP address 172.16.30.56, the network address is 172.16 and the node address is 30.56.

With a network address being 2 bytes (8 bits each), you get 2^{16} unique combinations. But the Internet designers decided that all Class B network addresses should start with the binary digit 1, then 0. This leaves 14 bit positions to manipulate, therefore 16,384, or 2^{14} unique Class B network addresses.

A Class B address uses 2 bytes for node addresses. This is 2^{16} minus the two reserved patterns of all 0s and all 1s for a total of 65,534 possible node addresses for each Class B network.

Class B Valid Host IDs

Here's an example of how to find the valid hosts in a Class B network:

- All host bits turned off is the network address: 172.16.0.0.

- All host bits turned on is the broadcast address: 172.16.255.255.

The valid hosts would be the numbers in between the network address and the broadcast address: 172.16.0.1 through 172.16.255.254.

Class C Addresses

The first 3 bytes of a Class C network address are dedicated to the network portion of the address, with only 1 measly byte remaining for the node address. Here's the format:

network.network.network.node

Using the example IP address 192.168.100.102, the network address is 192.168.100 and the node address is 102.

In a Class C network address, the first three bit positions are always the binary 110. The calculation is as follows: 3 bytes, or 24 bits, minus 3 reserved positions leaves 21 positions. Hence, there are 2^{21}, or 2,097,152, possible Class C networks.

Each unique Class C network has 1 byte to use for node addresses. This leads to 2^8, or 256, minus the two reserved patterns of all 0s and all 1s, for a total of 254 node addresses for each Class C network.

Class C Valid Host IDs

Here's an example of how to find a valid host ID in a Class C network:

- All host bits turned off is the network ID: 192.168.100.0.
- All host bits turned on is the broadcast address: 192.168.100.255.

The valid hosts would be the numbers in between the network address and the broadcast address: 192.168.100.1 through 192.168.100.254.

Private IP Addresses (RFC 1918)

The people who created the IP addressing scheme also created private IP addresses. These addresses can be used on a private network, but they're not routable through the Internet. This is designed for the purpose of creating a measure of well-needed security, but it also conveniently saves valuable IP address space.

If every host on every network was required to have real routable IP addresses, we would have run out of IP addresses to hand out years ago. But by using private IP addresses, ISPs, corporations, and home users only need a relatively tiny group of bona fide IP addresses to connect their networks to the Internet. This is economical because they can use private IP addresses on their inside networks and get along just fine.

To accomplish this task, the ISP and the corporation—the end user, no matter who they are—need to use something called *Network Address Translation (NAT)*, which basically takes a private IP address and converts it for use on the Internet. Doing things this way saves megatons of address space—good for us all!

The reserved private addresses are listed in Table 1.5.

Table 1.5 Reserved IP address space

Address Class	**Reserved Address Space**
Class A	10.0.0.0 through 10.255.255.255
Class B	172.16.0.0 through 172.31.255.255
Class C	192.168.0.0 through 192.168.255.255

NOTE

You must know your private address space to become Cisco certified!

So, What Private IP Address Should I Use?

That’s a really great question: Should you use Class A, Class B, or even Class C private addressing when setting up your network? Let’s take Acme Corporation in SF as an example. This company is moving into a new building and needs a whole new network. It has 14 departments, with about 70 users in each. You could probably squeeze one or two Class C addresses to use, or maybe you could use a Class B, or even a Class A just for fun.

The rule of thumb in the consulting world is, when you're setting up a corporate network—regardless of how small it is—you should use a Class A network address because it gives you the most flexibility and growth options. For example, if you used the 10.0.0.0 network address with a /24 mask, then you'd have 65,536 networks, each with 254 hosts. Lots of room for growth with that network!

But if you're setting up a home network, you'd opt for a Class C address because it is the easiest for people to understand and configure. Using the default Class C mask gives you one network with 254 hosts—plenty for a home network.

With the Acme Corporation, a nice 10.1.*x*.0 with a /24 mask (the *x* is the subnet for each department) makes this easy to design, install, and troubleshoot.

IPv4 Address Types

Most people use the term *broadcast* as a generic term, and most of the time, we understand what they mean—but not always! For example, you might say, “The host broadcasted through a router to a DHCP server,” but, well, it’s pretty unlikely that this would ever really happen. What you probably mean—using the correct technical jargon—is, “The DHCP client broadcasted for an IP address and a router then forwarded this as a unicast packet to the DHCP server.” Oh, and remember that with IPv4, broadcasts are pretty important, but with IPv6, there aren’t any broadcasts sent at all.

Okay, I’ve referred to IP addresses throughout the preceding chapters and now all throughout this chapter, and even showed you some examples. But I really haven’t gone into the different terms and uses associated with them yet, and it’s about time I did. So here are the address types that I’d like to define for you:

Loopback (localhost) Used to test the IP stack on the local computer. Can be any address from 127.0.0.1 through 127.255.255.254.

Layer 2 broadcasts These are sent to all nodes on a LAN.

Broadcasts (layer 3) These are sent to all nodes on the network.

Unicast This is an address for a single interface, and these are used to send packets to a single destination host.

Multicast These are packets sent from a single source and transmitted to many devices on different networks. Referred to as “one-to-many.”

Layer 2 Broadcasts

First, understand that layer 2 broadcasts are also known as hardware broadcasts—they only go out on a LAN, but they don’t go past the LAN boundary (router).

The typical hardware address is 6 bytes (48 bits) and looks something like 45:AC:24:E3:60:A5. The broadcast would be all 1s in binary, which would be all *F*s in hexadecimal, as in ff:ff:ff:ff:ff:ff and shown in Figure 1.21.

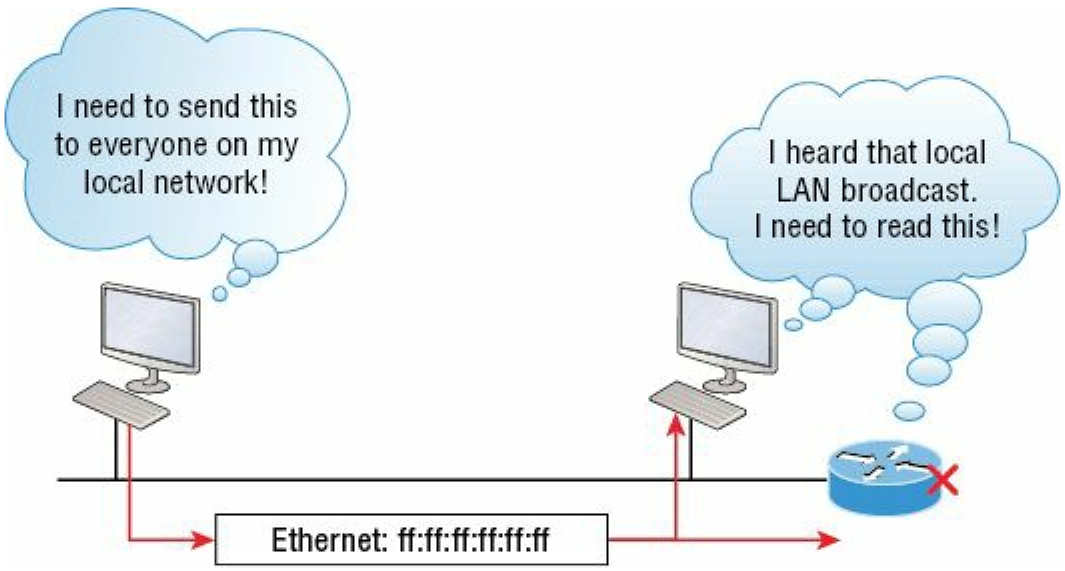

I need to send this to everyone on my local network!
I heard that local LAN broadcast. I need to read this!
Ethernet: ff:ff:ff:ff:ff:ff

Figure 1.21 Local layer 2 broadcasts

Every network interface card (NIC) will receive and read the frame, including the router, since this was a layer 2 broadcast, but the router would never, ever forward this!

Layer 3 Broadcasts

Then there are the plain old broadcast addresses at layer 3. Broadcast messages are meant to reach all hosts on a broadcast domain. These are the network broadcasts that have all host bits on.

Here's an example that you're already familiar with: The network address of 172.16.0.0 255.255.0.0 would have a broadcast address of 172.16.255.255—all host bits on. Broadcasts can also be "any network and all hosts," as indicated by 255.255.255.255, and shown in Figure 1.22.

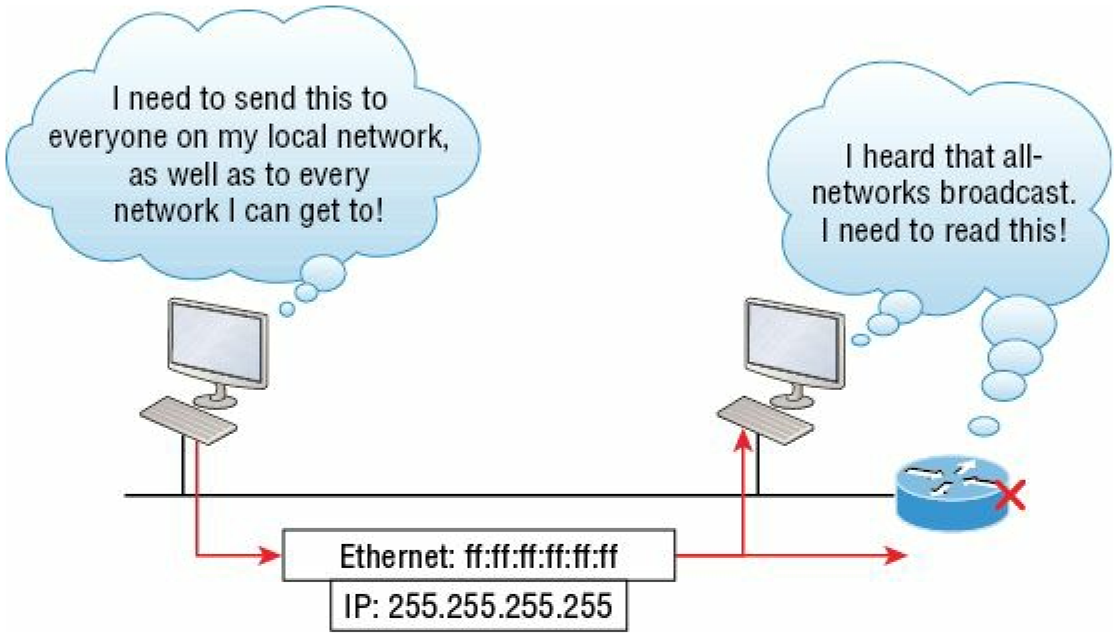

I need to send this to everyone on my local network, as well as to every network I can get to!
I heard that all-networks broadcast. I need to read this!
Ethernet: ff:ff:ff:ff:ff:ff
IP: 255.255.255.255

Figure 1.22 Layer 3 broadcasts

In Figure 1.22, all hosts on the LAN will get this broadcast on their NIC, including the router, but by default the router would never forward this packet.

Unicast Address

A unicast is defined as a single IP address that's assigned to a network interface card and is the destination IP address in a packet—in other words, it's used for directing packets to a specific host.

In Figure 1.23, both the MAC address and the destination IP address are for a single NIC on the network. All hosts on the broadcast domain would receive this frame and accept it. Only the destination NIC of 10.1.1.2 would accept the packet; the other NICs would discard the packet.

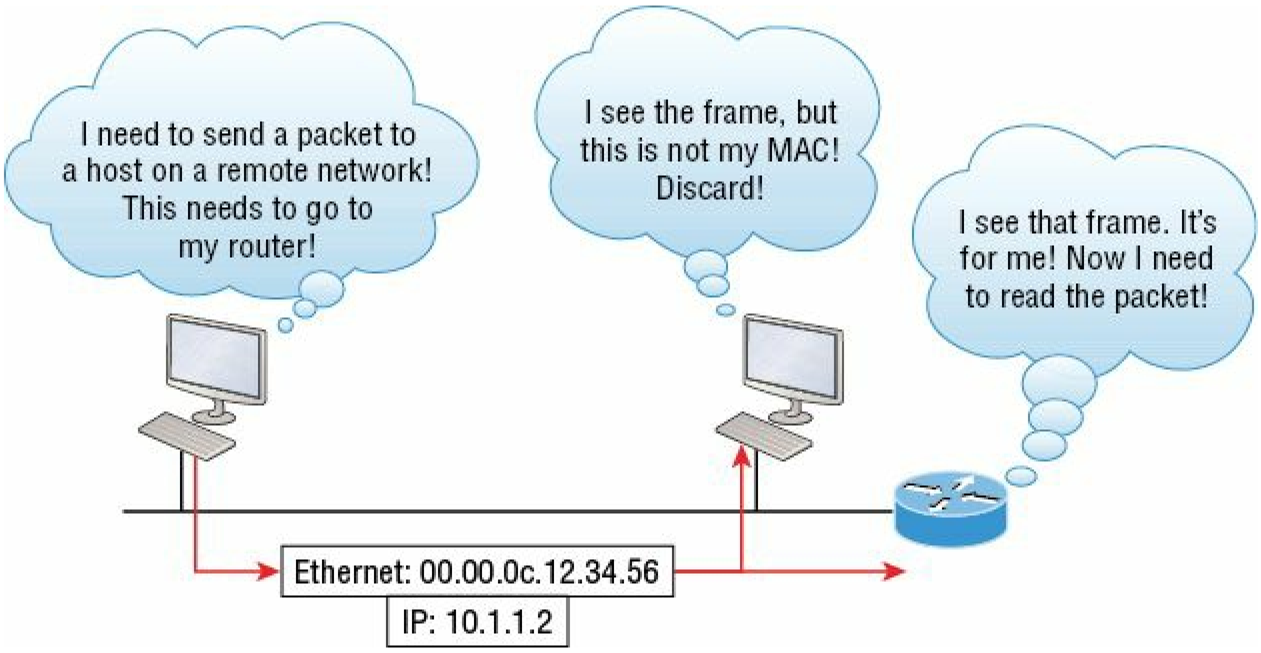

I need to send a packet to a host on a remote network! This needs to go to my router!
I see the frame, but this is not my MAC! Discard!
I see that frame. It's for me! Now I need to read the packet!
Ethernet: 00.00.0c.12.34.56
IP: 10.1.1.2

Figure 1.23 Unicast address

Multicast Address

Multicast is a different beast entirely. At first glance, it appears to be a hybrid of unicast and broadcast communication, but that isn't quite the case. Multicast does allow point-to-multipoint communication, which is similar to broadcasts, but it happens in a different manner. The crux of *multicast* is that it enables multiple recipients to receive messages without flooding the messages to all hosts on a broadcast domain. However, this is not the default behavior—it's what we *can* do with multicasting if it's configured correctly!

Multicast works by sending messages or data to IP *multicast group* addresses. Unlike with broadcasts, which aren't forwarded, routers then forward copies of the packet out to every interface that has hosts *subscribed* to that group address. This is where multicast differs from broadcast messages—with multicast communication, copies of packets, in theory, are sent only to subscribed hosts. For example, when I say in theory, I mean that the hosts will receive a multicast packet destined for 224.0.0.10. This is an EIGRP packet, and only a router running the EIGRP protocol will read these. All hosts on the broadcast LAN, and Ethernet is a broadcast multi-access LAN technology, will pick up the frame, read the destination address, then immediately discard the frame unless they're in the multicast group. This saves PC processing, not LAN bandwidth. Be warned though—multicasting can cause some serious LAN congestion if it's not implemented carefully! Figure 1.24 shows a Cisco router sending an EIGRP multicast packet on the local LAN and only the other Cisco router will accept and read this packet.

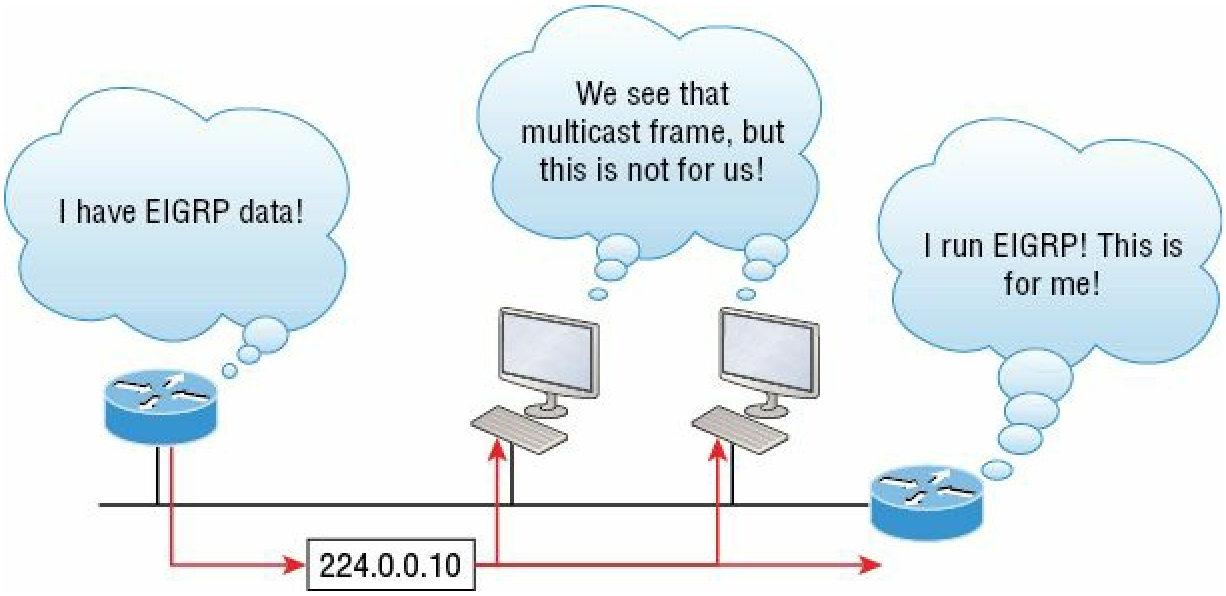

I have EIGRP data!
We see that multicast frame, but this is not for us!
I run EIGRP! This is for me!
224.0.0.10

Figure 1.24 EIGRP multicast example

There are several different groups that users or applications can subscribe to. The range of multicast addresses starts with 224.0.0.0 and goes through 239.255.255.255. As you can see, this range of addresses falls within IP Class D address space based on classful IP assignment.

Summary

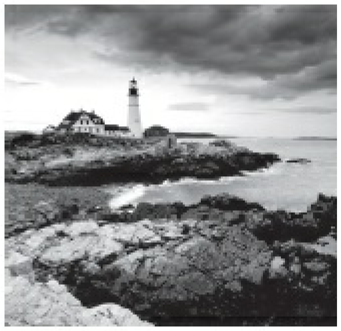
If you made it this far and understood everything the first time through, you should be extremely proud of yourself! We really covered a lot of ground in this chapter, but understand that the information in it is critical to being able to navigate well through the rest of this book.

If you didn't get a complete understanding the first time around, don't stress. It really wouldn't hurt you to read this chapter more than once. There is still a lot of ground to cover, so make sure you've got this material all nailed down. That way, you'll be ready for more, and just so you know, there's a lot more! What we're doing up to this point is building a solid foundation to build upon as you advance.

With that in mind, after you learned about the DoD model, the layers, and associated protocols, you learned about the oh-so-important topic of IP addressing. I discussed in detail the difference between each address class, how to find a network address and broadcast address, and what denotes a valid host address range.

Since you've already come this far, there's no reason to stop now and waste all those brainwaves and new neural connections. So don't stop—go through the written labs and review questions at the end of this chapter and make sure you understand each answer's explanation. The best is yet to come!

Chapter 2
Easy Subnetting



We'll pick up right where we left off in the last chapter and continue to explore the world of IP addressing. I'll open this chapter by telling you how to subnet an IP network—an indispensably crucial skill that's central to mastering networking in general! Forewarned is forearmed, so prepare yourself because being able to subnet quickly and accurately is pretty challenging and you'll need time to practice what you've learned to really nail it. So be patient and don't give up on this key aspect of networking until your skills are seriously sharp. I'm not kidding—this chapter is so important you should really just graft it into your brain!

So be ready because we're going to hit the ground running and thoroughly cover IP subnetting from the very start. And though I know this will sound weird to you, you'll be much better off if you just try to forget everything you've learned about subnetting before reading this chapter—especially if you've been to an official Cisco or Microsoft class! I think these forms of special torture often do more harm than good and sometimes even scare people away from networking completely. Those that survive and persevere usually at least question the sanity of continuing to study in this field. If this is you, relax, breathe, and know that you'll find that the way I tackle the issue of subnetting is relatively painless because I'm going to show you a whole new, much easier method to conquer this monster!

After working through this chapter, and I can't say this enough, after working through the extra study material at the end as well, you'll be able to tame the IP addressing/subnetting beast—just don't give up! I promise that you'll be really glad you didn't. It's one of those things that once you get it down, you'll wonder why you used to think it was so hard!

To find up-to-the minute updates for this chapter, please see www.lammle.com/ccna or the book's web page at www.sybex.com/go/ccna.

Subnetting Basics

In Chapter 1, "Introduction to TCP/IP," you learned how to define and find the valid host ranges used in a Class A, Class B, and Class C network address by turning the host bits all off and then all on. This is very good, but here's the catch: you were defining only one network, as shown in Figure 2.1.

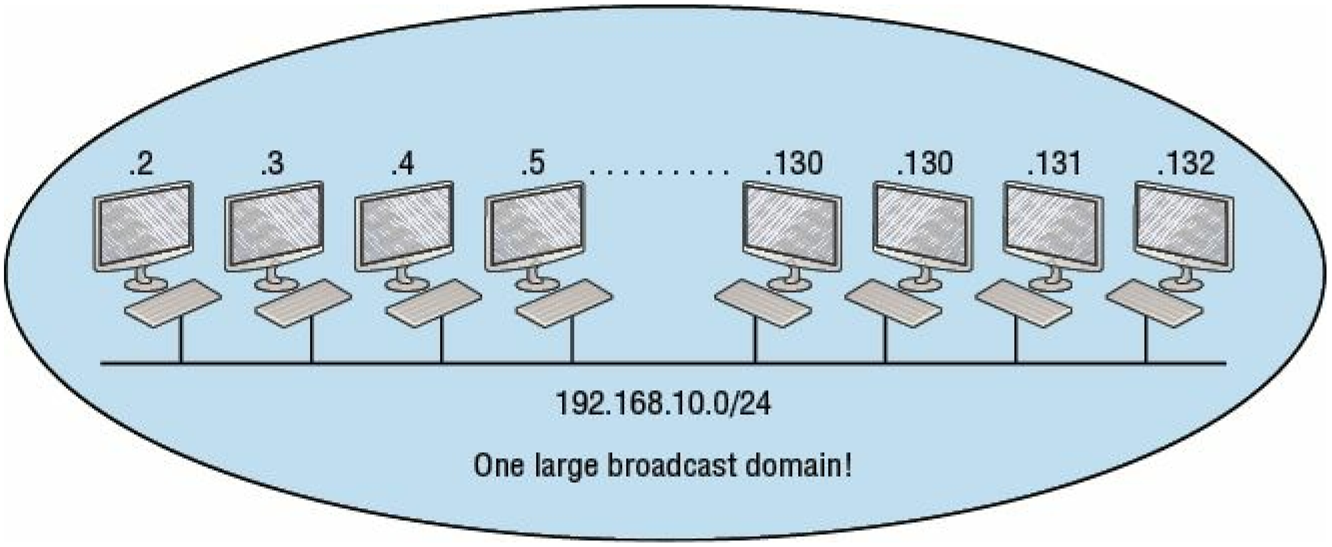


Figure 2.1 One network

By now you know that having one large network is not a good thing because the chapter you just read was veritably peppered with me incessantly telling you that! But how would you fix the out-of-control problem that Figure 2.1 illustrates? Wouldn't it be nice to be able to break up that one, huge network address and create four manageable networks from it? You betcha it would, but to make that happen, you would need to apply the infamous trick of *subnetting* because it's the best way to break up a giant network into a bunch of smaller ones. Take a look at Figure 2.2 and see how this might look.

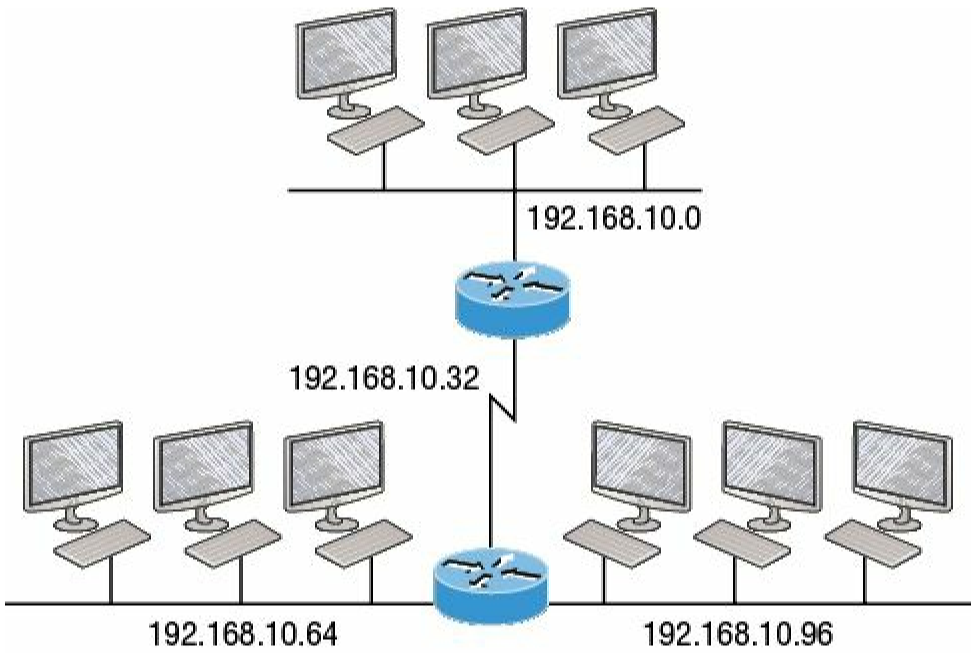


Figure 2.2 Multiple networks connected together

What are those 192.168.10.*x* addresses shown in the figure? Well that is what this chapter will explain—how to make one network into many networks!

Let's take off from where we left in Chapter 1 and start working in the host section (host bits) of a network address, where we can borrow bits to create subnets.

How to Create Subnets

Creating subnetworks is essentially the act of taking bits from the host portion of the address and reserving them to define the subnet address instead. Clearly this will result in fewer bits being available for defining your hosts, which is something you'll always want to keep in mind.

Later in this chapter, I'll guide you through the entire process of creating subnets starting with Class C addresses. As always in networking, before you actually implement anything, including subnetting, you must first determine your current requirements and make sure to plan for future conditions as well.

NOTE

In this first section, we'll be discussing classful routing, which refers to the fact that all hosts (nodes) in the network are using the exact same *subnet* mask. Later, when we move on to cover variable length subnet masks (VLSMs), I'll tell you all about classless routing, which is an environment wherein each network segment *can* use a different subnet mask.

To create a subnet, we'll start by fulfilling these three steps:

1. Determine the number of required network IDs:
   - One for each LAN subnet
   - One for each wide area network connection
2. Determine the number of required host IDs per subnet:
   - One for each TCP/IP host
   - One for each router interface
3. Based on the previous requirements, create the following:
   - A unique subnet mask for your entire network
   - A unique subnet ID for each physical segment
   - A range of host IDs for each subnet

Subnet Masks

For the subnet address scheme to work, every machine on the network must know which part of the host address will be used as the subnet address. This condition is met by *assigning* a *subnet mask* to each machine. A subnet mask is a 32-bit value that allows the device that's receiving IP packets to distinguish the network ID portion of the IP address from the host ID portion of the IP address. This 32-bit subnet mask is composed of 1s and 0s, where the 1s represent the positions that refer to the network subnet addresses.

Not all networks need subnets, and if not, it really means that they're using the default subnet mask, which is basically the same as saying that a network doesn't have a subnet address. Table 2.1shows the default subnet masks for Classes A, B, and C.

Table 2.1 Default subnet mask

Class	Format	Default Subnet Mask

A	*network.node.node.node*	255.0.0.0
B	*network.network.node.node*	255.255.0.0
C	*network.network.network.node*	255.255.255.0

Although you can use any mask in any way on an interface, typically it's not usually good to mess with the default masks. In other words, you don't want to make a Class B subnet mask read 255.0.0.0, and some hosts won't even let you type it in. But these days, most devices will. For a Class A network, you wouldn't change the first byte in a subnet mask because it should read 255.0.0.0 at a minimum. Similarly, you wouldn't assign 255.255.255.255 because this is all 1s, which is a broadcast address. A Class B address starts with 255.255.0.0, and a Class C starts with 255.255.255.0, and for the CCNA especially, there is no reason to change the defaults!

Understanding the Powers of 2

Powers of 2 are important to understand and memorize for use with IP subnetting. Reviewing powers of 2, remember that when you see a number noted with an exponent, it means you should multiply the number by itself as many times as the upper *number* specifies. For example, 2^3 is 2 x 2 x 2, which equals 8. Here's a list of powers of 2 to *commit* to memory:

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

$2^5 = 32$

$2^6 = 64$

$2^7 = 128$

$2^8 = 256$

$2^9 = 512$

$2^{10} = 1,024$

$2^{11} = 2,048$

$2^{12} = 4,096$

$2^{13} = 8,192$

$2^{14} = 16,384$

Memorizing these powers of 2 is a good idea, but it's not absolutely necessary. Just remember that since you're working with powers of 2, each successive power of 2 is double the previous one.

It works like this—all you have to do to remember the value of 2^9 is to first know that $2^8 = 256$. Why? Because when you double 2 to the eighth power (256), you get 2^9 (or 512). To determine the value of 2^{10}, simply start at $2^8 = 256$, and then double it twice.

You can go the other way as well. If you needed to know what 2^6 is, for example, you just cut 256 in half two times: once to reach 2^7 and then one more time to reach 2^6.

Classless Inter-Domain Routing (CIDR)

Another term you need to familiarize yourself with is *Classless Inter-Domain Routing (CIDR)*. It's basically the method that Internet service providers (ISPs) use to allocate a number of addresses to a company, a home—their customers. They provide addresses in a certain block size, something I'll talk about in greater detail soon.

When you receive a block of addresses from an ISP, what you get will look *something* like this: 192.168.10.32/28. This is telling you what your subnet mask is. The slash *notation* (/) means how many bits are turned on (1s). Obviously, the maximum could only be /32 because a byte is 8 bits and there are 4 bytes in an IP address: (4 × 8 = 32). But keep in mind that regardless of the class of address, the largest subnet mask available relevant to the Cisco exam objectives can only be a /30 because you've got to keep at least 2 bits for host bits.

Take, for example, a Class A default subnet mask, which is 255.0.0.0. This tells us that the first byte of the subnet mask is all ones (1s), or 11111111. When referring to a slash notation, you need to count all the 1 bits to figure out your mask. The 255.0.0.0 is *considered* a /8 because it has 8 bits that are 1s—that is, 8 bits that are turned on.

A Class B default mask would be 255.255.0.0, which is a /16 because 16 bits are ones (1s): 11111111.11111111.00000000.00000000.

Table 2.2 has a listing of every available subnet mask and its equivalent CIDR slash notation.

Table 2.2 CIDR values

Subnet Mask	**CIDR Value**
255.0.0.0	/8
255.128.0.0	/9
255.192.0.0	/10
255.224.0.0	/11
255.240.0.0	/12
255.248.0.0	/13
255.252.0.0	/14
255.254.0.0	/15
255.255.0.0	/16
255.255.128.0	/17
255.255.192.0	/18
255.255.224.0	/19

255.255.240.0	/20
255.255.248.0	/21
255.255.252.0	/22
255.255.254.0	/23
255.255.255.0	/24
255.255.255.128	/25
255.255.255.192	/26
255.255.255.224	/27
255.255.255.240	/28
255.255.255.248	/29
255.255.255.252	/30

The /8 through /15 can only be used with Class A network addresses. /16 through /23 can be used by Class A and B network addresses. /24 through /30 can be used by Class A, B, and C network addresses. This is a big reason why most companies use Class A network addresses. Since they can use all subnet masks, they get the maximum flexibility in network design.

No, you cannot configure a Cisco router using this slash format. But wouldn't that be nice? Nevertheless, it's *really* important for you to know subnet masks in the slash notation (CIDR).

IP Subnet-Zero

Even though `ip subnet-zero` is not a new command, Cisco courseware and Cisco exam objectives didn't used to cover it. Know that Cisco certainly covers it now! This command allows you to use the first and last subnet in your network design. For instance, the Class C mask of 255.255.255.192 provides subnets 64 and 128, another facet of subnetting that we'll discuss more thoroughly later in this chapter. But with the `ip subnet-zero` command, you now get to use subnets 0, 64, 128, and 192. It may not seem like a lot, but this provides two more subnets for every subnet mask we use.

Even though we don't discuss the command-line interface (CLI), it's important for you to be at least a little familiar with this command at this point:

```
Router#sh running-config

Building configuration...

Current configuration : 827 bytes

!

hostname Pod1R1

!

ip subnet-zero

!
```

This router output shows that the command `ip subnet-zero` is enabled on the router. Cisco has turned this command on by default starting with Cisco IOS version 12.*x* and now we're running 15.*x* code.

When taking your Cisco exams, make sure you read very carefully to see if Cisco is asking you not to use `ip subnet-zero`. There are actually instances where this may happen.

Subnetting Class C Addresses

There are many different ways to subnet a network. The right way is the way that works best for you. In a Class C address, only 8 bits are available for defining the hosts. Remember that subnet bits start at the left and move to the right, without skipping bits. This means that the only Class C subnet masks can be the following:

```
Binary     Decimal  CIDR
```

```
----------------------------------------------------------

00000000 = 255.255.255.0         /24

10000000 = 255.255.255.128       /25

11000000 = 255.255.255.192       /26

11100000 = 255.255.255.224       /27

11110000 = 255.255.255.240       /28

11111000 = 255.255.255.248       /29

11111100 = 255.255.255.252       /30
```

We can't use a /31 or /32 because, as I've said, we must have at least 2 host bits for assigning IP addresses to hosts. But this is only mostly true. Certainly we can never use a /32 because that would mean zero host bits available, yet Cisco has various forms of the IOS, as well as the new Cisco Nexus switches operating system, that support the /31 mask. The /31 is above the scope of the CCENT and CCNA objectives, so we won't be covering it in this book.

Coming up, I'm going to teach you that significantly less painful method of subnetting I promised you at the beginning of this chapter, which makes it ever so much easier to subnet larger numbers in a flash. Excited? Good! Because I'm not kidding when I tell you that you absolutely need to be able to subnet quickly and accurately to succeed in the networking real world and on the exam too!

Subnetting a Class C Address—The Fast Way!

When you've chosen a possible subnet mask for your network and need to determine the number of subnets, valid hosts, and the broadcast addresses of a subnet that mask will provide, all you need to do is answer five simple questions:

- How many subnets does the chosen subnet mask produce?
- How many valid hosts per subnet are available?
- What are the valid subnets?
- What's the broadcast address of each subnet?
- What are the valid hosts in each subnet?

This is where you'll be really glad you followed my advice and took the time to memorize your powers of 2. If you didn't, now would be a good time... Just refer back to the sidebar "Understanding the Powers of 2" earlier if you need to brush up. Here's how you arrive at the answers to those five big questions:

- *How many subnets?* 2x = number of subnets. *x* is the number of masked bits, or the 1s. For example, in 11000000, the number of 1s gives us 2^2 subnets. So in this example, there are 4 subnets.
- *How many hosts per subnet?* 2*y* – 2 = number of hosts per subnet. *y* is the number of unmasked bits, or the 0s. For example, in 11000000, the number of 0s gives us 2^6 – 2 hosts, or 62 hosts per subnet. You need to subtract 2 for the subnet address and the broadcast address, which are not valid hosts.
- *What are the valid subnets?* 256 – subnet mask = block size, or increment number. An example would be the 255.255.255.192 mask, where the interesting octet is the fourth octet (interesting because that is where our subnet numbers are). Just use this math: 256 – 192 = 64. The block size of a 192 mask is always 64. Start counting at zero in blocks of 64 until you reach the subnet mask value and these are your subnets in the fourth octet: 0, 64, 128, 192. Easy, huh?
- *What's the broadcast address for each subnet?* Now here's the really easy part. Since we counted our subnets in the last section as 0, 64, 128, and 192, the broadcast address is always the number right before the next subnet. For example, the 0 subnet has a broadcast address of 63 because the next subnet is 64. The 64 subnet has a broadcast address of 127 because the next subnet is 128, and so on. Remember, the broadcast address of the last subnet is always 255.
- *What are the valid hosts?* Valid hosts are the numbers between the subnets, omitting the all-0s and all-1s. For example, if 64 is the subnet number and 127 is the broadcast address, then 65–126 is the valid host range. Your valid range is *always* the group of numbers between the subnet address and the broadcast address.

If you're still confused, don't worry because it really isn't as hard as it seems to be at first—just hang in there! To help lift any mental fog, try a few of the practice examples next.

Subnetting Practice Examples: Class C Addresses

Here's your opportunity to practice subnetting Class C addresses using the method I just described. This is so cool. We're going to start with the first Class C subnet mask and work through every subnet that we can, using a

Class C address. When we're done, I'll show you how easy this is with Class A and B networks too!

Practice Example #1C: 255.255.255.128 (/25)

Since 128 is 10000000 in binary, there is only 1 bit for subnetting and 7 bits for hosts. We're going to subnet the Class C network address 192.168.10.0.

. 192.168.10.0 = Network address

. 255.255.255.128 = Subnet mask

Now, let's answer our big five:

- *How many subnets?* Since 128 is 1 bit on (**1**0000000), the answer would be $2^1 = 2$.
- *How many hosts per subnet?* We have 7 host bits off (**1**0000000), so the equation would be $2^7 – 2 = 126$ hosts. Once you figure out the block size of a mask, the amount of hosts is always the block size minus 2. No need to do extra math if you don't need to!
- *What are the valid subnets?* 256 – 128 = 128. Remember, we'll start at zero and count in our block size, so our subnets are 0, 128. By just counting your subnets when counting in your block size, you really don't need to do steps 1 and 2. We can see we have two subnets, and in the step before this one, just remember that the amount of hosts is always the block size minus 2, and in this example, that gives us 2 subnets, each with 126 hosts.
- *What's the broadcast address for each subnet?* The number right before the value of the next subnet is all host bits turned on and equals the broadcast address. For the zero subnet, the next subnet is 128, so the broadcast of the 0 subnet is 127.
- *What are the valid hosts?* These are the numbers between the subnet and broadcast address. The easiest way to find the hosts is to write out the subnet address and the broadcast address, which makes valid hosts completely obvious. The following table shows the 0 and 128 subnets, the valid host ranges of each, and the broadcast address of both subnets:

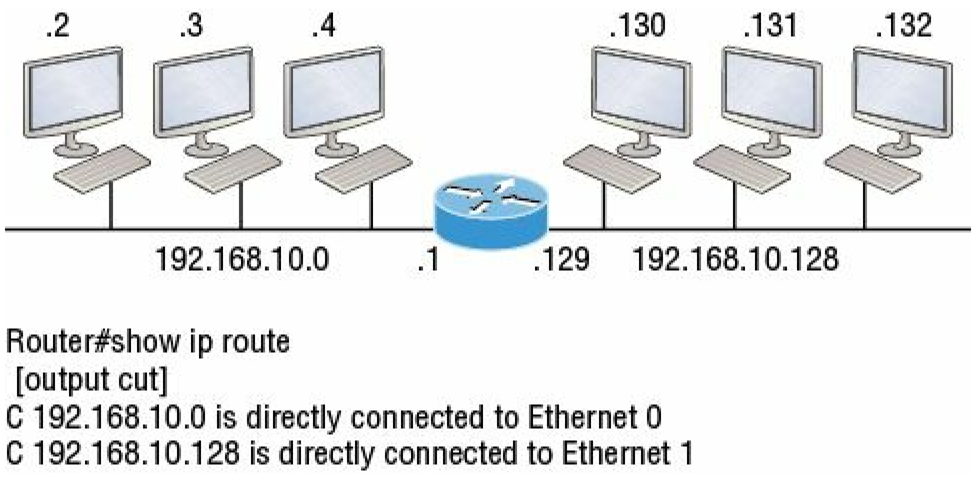
Subnet	**0**	**128**

First host	1	129
Last host	126	254
Broadcast	**127**	**255**

Looking at a Class C /25, it's pretty clear that there are two subnets. But so what—why is this significant? Well actually, it's not because that's not the right question. What you really want to know is what you would do with this information!

I know this isn't exactly everyone's favorite pastime, but what we're about to do is really important, so bear with me; we're going to talk about subnetting —period. The key to understanding subnetting is to understand the very reason you need to do it, and I'm going to demonstrate this by going through the process of building a physical network.

Okay—because we added that router shown in Figure 2.3, in order for the hosts on our internetwork to communicate, they must now have a logical network addressing scheme. We could use IPv6, but IPv4 is still the most popular for now. It's also what we're studying at the moment, so that's what we're going with.



```
Router#show ip route
 [output cut]
C 192.168.10.0 is directly connected to Ethernet 0
C 192.168.10.128 is directly connected to Ethernet 1
```

Figure 2.3 Implementing a Class C /25 logical network

Looking at Figure 2.3, you can see that there are two physical networks, so we're going to implement a logical addressing scheme that allows for two logical networks. As always, it's a really good idea to look ahead and consider likely short- and long-term growth scenarios, but for this example in this book, a /25 gets it done.

Figure 2.3 shows us that both subnets have been assigned to a router interface, which creates our broadcast domains and assigns our subnets. Use the command `show ip route` to see the routing table on a router. Notice that instead of one large broadcast domain, there are now two smaller broadcast domains, providing for up to 126 hosts in each. The `C` in the router output translates to "directly connected network," and we can see we have two of those with two broadcast domains and that we created and implemented them. So congratulations—you did it! You have successfully subnetted a network and applied it to a network design. Nice! Let's do it again.

Practice Example #2C: 255.255.255.192 (/26)

This time, we're going to subnet the network address 192.168.10.0 using the subnet mask 255.255.255.192.

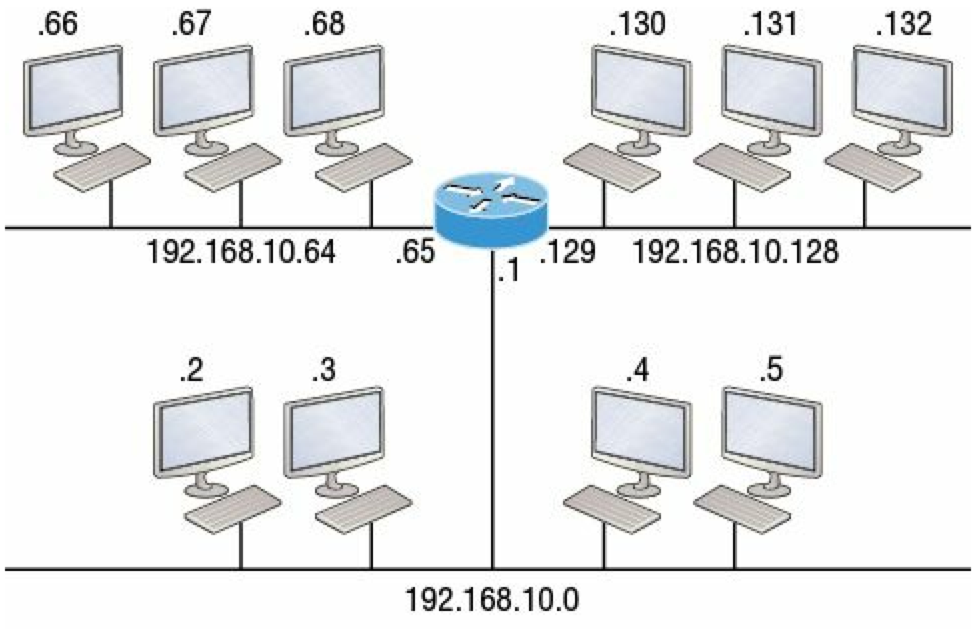
- 192.168.10.0 = Network address
- 255.255.255.192 = Subnet mask

Now, let's answer the big five:

- *How many subnets?* Since 192 is 2 bits on (**1**1000000), the answer would be 2^2 = 4 subnets.
- *How many hosts per subnet?* We have 6 host bits off (11000000), giving us $2^6 - 2$ = 62 hosts. The amount of hosts is always the block size minus 2.
- *What are the valid subnets?* 256 – 192 = 64. Remember to start at zero and count in our block size. This means our subnets are 0, 64, 128, and 192. We can see we have a block size of 64, so we have 4 subnets, each with 62 hosts.
- *What's the broadcast address for each subnet?* The number right before the value of the next subnet is all host bits turned on and equals the broadcast address. For the zero subnet, the next subnet is 64, so the broadcast address for the zero subnet is 63.
- *What are the valid hosts?* These are the numbers between the subnet and broadcast address. As I said, the easiest way to find the hosts is to write out the subnet address and the broadcast address, which clearly delimits our valid hosts. The following table shows the 0, 64, 128, and 192 subnets, the valid host ranges of each, and the broadcast address of each subnet:

The subnets (do this first)	**0**	**64**	**128**	**192**
Our first host (perform host addressing last)	1	65	129	193
Our last host	62	126	190	254
The broadcast address (do this second)	63	127	191	255

Again, before getting into the next example, you can see that we can now subnet a /26 as long as we can count in increments of 64. And what are you going to do with this fascinating information? Implement it! We'll use Figure 2.4 to practice a /26 network implementation.



```
Router#show ip route
 [output cut]
C 192.168.10.0 is directly connected to Ethernet 0
C 192.168.10.64 is directly connected to Ethernet 1
C 192.168.10.128 is directly connected to Ethernet 2
```

Figure 2.4 Implementing a class C /26 (with three networks)

The /26 mask provides four subnetworks, and we need a subnet for each router interface. With this mask, in this example, we actually have room with a spare subnet to add to another router interface in the future. Always plan for growth if possible!

Practice Example #3C: 255.255.255.224 (/27)

This time, we'll subnet the network address 192.168.10.0 and subnet mask 255.255.255.224.

. 192.168.10.0 = Network address

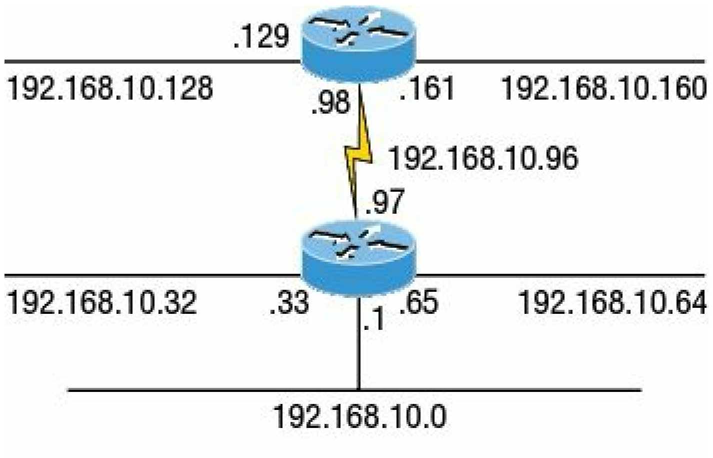
. 255.255.255.224 = Subnet mask

- *How many subnets?* 224 is 11100000, so our equation would be 2^3 = 8.
- *How many hosts?* 2^5 – 2 = 30.
- *What are the valid subnets?* 256 – 224 = 32. We just start at zero and count to the subnet mask value in blocks (increments) of 32: 0, 32, 64, 96, 128, 160, 192, and 224.
- *What's the broadcast address for each subnet (always the number right before the next subnet)?*
- *What are the valid hosts (the numbers between the subnet number and the broadcast address)?*

To answer the last two questions, first just write out the subnets, then write out the broadcast addresses—the number right before the next subnet. Last, fill in the host addresses. The following table gives you all the subnets for the 255.255.255.224 Class C subnet mask:

The subnet address	**0**	**32**	**64**	**96**	**128**	**160**	**192**	**224**
The first valid host	1	33	65	97	129	161	193	225
The last valid host	30	62	94	126	158	190	222	254
The broadcast address	31	63	95	127	159	191	223	255

In practice example #3C, we're using a 255.255.255.224 (/27) network, which provides eight subnets as shown previously. We can take these subnets and implement them as shown in Figure 2.5 using any of the subnets available.



```
Router#show ip route
 [output cut]
C 192.168.10.0 is directly connected to Ethernet 0
C 192.168.10.32 is directly connected to Ethernet 1
C 192.168.10.64 is directly connected to Ethernet 2
C 192.168.10.96 is directly connected to Serial 0
```

Figure 2.5 Implementing a Class C /27 logical network

Notice that used six of the eight subnets available for my network design. The lightning bolt symbol in the figure represents a wide area network (WAN) such as a T1 or other serial connection through an ISP or telco. In other words, something you don't own, but it's still a subnet just like any LAN connection on a router. As usual, I used the first valid host in each subnet as the router's interface address. This is just a rule of thumb; you can use any address in the valid host range as long as you remember what address you configured so you can set the default gateways on your hosts to the router address.

Practice Example #4C: 255.255.255.240 (/28)

Let's practice another one:

192.168.10.0 = Network address

255.255.255.240 = Subnet mask

- *Subnets?* 240 is 11110000 in binary. 2^4 = 16.
- *Hosts?* 4 host bits, or $2^4 - 2 = 14$.
- *Valid subnets?* 256 – 240 = 16. Start at 0: 0 + 16 = 16. 16 + 16 = 32. 32 + 16 = 48. 48 + 16 = 64. 64 + 16 = 80. 80 + 16 = 96. 96 + 16 = 112. 112 + 16 = 128. 128 + 16 = 144. 144 + 16 = 160. 160 + 16 = 176. 176 + 16 = 192. 192 + 16 = 208. 208 + 16 = 224. 224 + 16 = 240.

- *Broadcast address for each subnet?*
- *Valid hosts?*

To answer the last two questions, check out the following table. It gives you the subnets, valid hosts, and broadcast addresses for each subnet. First, find the address of each subnet using the block size (increment). Second, find the broadcast address of each subnet increment, which is always the number right before the next valid subnet, and then just fill in the host addresses. The following table shows the available subnets, hosts, and broadcast addresses provided from a Class C 255.255.255.240 mask.

Subnet	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
First host	1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241
Last host	14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254
Broadcast	15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255

TIP

Cisco has figured out that most people cannot count in 16s and therefore have a hard time finding valid subnets, hosts, and broadcast addresses with the Class C 255.255.255.240 mask. You'd be wise to study this mask.

Practice Example #5C: 255.255.255.248 (/29)

Let's keep practicing:

. 192.168.10.0 = Network address

. 255.255.255.248 = Subnet mask

- *Subnets?* 248 in binary = 11111000. 2^5 = 32.
- *Hosts?* 2^3 – 2 = 6.
- *Valid subnets?* 256 – 248 = 0, 8, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96, 104, 112, 120, 128, 136, 144, 152, 160, 168, 176, 184, 192, 200, 208, 216, 224, 232, 240, and 248.
- *Broadcast address for each subnet?*
- *Valid hosts?*

Take a look at the following table. It shows some of the subnets (first four and last four only), valid hosts, and broadcast addresses for the Class C 255.255.255.248 mask:

Subnet	**0**	**8**	**16**	**24**	**…**	**224**	**232**	**240**	**248**
First host	1	9	17	25	…	225	233	241	249
Last host	6	14	22	30	…	230	238	246	254
Broadcast	7	15	23	31	…	231	239	247	255

TIP

If you try to configure a router interface with the address 192.168.10.6 255.255.255.248 and receive the following error, It means that `ip subnet-zero` is not enabled:

```
Bad mask /29 for address 192.168.10.6
```

You must be able to subnet to see that the address used in this example is in the zero subnet!

Practice Example #6C: 255.255.255.252 (/30)

Okay—just one more:

, 192.168.10.0 = Network address

, 255.255.255.252 = Subnet mask

- *Subnets?* 64.
- *Hosts?* 2.
- *Valid subnets?* 0, 4, 8, 12, etc., all the way to 252.
- *Broadcast address for each subnet? (Always the number right before the next subnet.)*
- *Valid hosts?* (The numbers between the subnet number and the broadcast

address.)

The following table shows you the subnet, valid host, and broadcast address of the first four and last four subnets in the 255.255.255.252 Class C subnet:

Subnet	**0**	**4**	**8**	**12**	**…**	**240**	**244**	**248**	**252**
First host	1	5	9	13	…	241	245	249	253
Last host	2	6	10	14	…	242	246	250	254
Broadcast	3	7	11	15	…	243	247	251	255

Real World Scenario

Should We Really Use This Mask That Provides Only Two Hosts?

You are the network administrator for Acme Corporation in San Francisco, with dozens of WAN links connecting to your corporate office. Right now your network is a classful network, which means that the same subnet mask is on each host and router interface. You've read about classless routing, where you can have different sized masks, but don't know what to use on your point-to-point WAN links. Is the 255.255.255.252 (/30) a helpful mask in this situation?

Yes, this is a very helpful mask in wide area networks and of course with any type of point-to-point link!

If you were to use the 255.255.255.0 mask in this situation, then each network would have 254 hosts. But you use only 2 addresses with a WAN or point-to-point link, which is a waste of 252 hosts per subnet! If you use the 255.255.255.252 mask, then each subnet has only 2 hosts, and you don't want to waste precious addresses. This is a really important subject, one that we'll address in a lot more detail in the section on VLSM network design in the next chapter!

Subnetting in Your Head: Class C Addresses

It really is possible to subnet in your head? Yes, and it's not all that hard either—take the following example:

- 192.168.10.50 = Node address
- 255.255.255.224 = Subnet mask

First, determine the subnet and broadcast address of the network in which the previous IP address resides. You can do this by answering question 3 of the big 5 questions: 256 – 224 = 32. 0, 32, 64, and so on. The address of 50 falls between the two subnets of 32 and 64 and must be part of the 192.168.10.32 subnet. The next subnet is 64, so the broadcast address of the 32 subnet is 63. Don't forget that the broadcast address of a subnet is always the number right before the next subnet. The valid host range equals the numbers between the subnet and broadcast address, or 33–62. This is too easy!

Let's try another one. We'll subnet another Class C address:

- 192.168.10.50 = Node address
- 255.255.255.240 = Subnet mask

What is the subnet and broadcast address of the network of which the previous IP address is a member? 256 – 240 = 16. Now just count by our increments of 16 until we pass the host address: 0, 16, 32, 48, 64. Bingo—the host address is between the 48 and 64 subnets. The subnet is 192.168.10.48, and the broadcast address is 63 because the next subnet is 64. The valid host range equals the numbers between the subnet number and the broadcast address, or 49–62.

Let's do a couple more to make sure you have this down.

You have a node address of 192.168.10.174 with a mask of 255.255.255.240. What is the valid host range?

The mask is 240, so we'd do a 256 – 240 = 16. This is our block size. Just keep adding 16 until we pass the host address of 174, starting at zero, of course: 0, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176. The host address of 174 is between 160 and 176, so the subnet is 160. The broadcast address is 175; the valid host range is 161–174. That was a tough one!

One more—just for fun. This one is the easiest of all Class C subnetting:

- 192.168.10.17 = Node address
- 255.255.255.252 = Subnet mask

What is the subnet and broadcast address of the subnet in which the previous IP address resides? 256 – 252 = 0 (always start at zero unless told otherwise).

0, 4, 8, 12, 16, 20, etc. You've got it! The host address is between the 16 and 20 subnets. The subnet is 192.168.10.16, and the broadcast address is 19. The valid host range is 17–18.

Now that you're all over Class C subnetting, let's move on to Class B subnetting. But before we do, let's go through a quick review.

What Do We Know?

Okay—here's where you can really apply what you've learned so far and begin committing it all to memory. This is a very cool section that I've been using in my classes for years. It will really help you nail down subnetting for good!

When you see a subnet mask or slash notation (CIDR), you should know the following:

/25 What do we know about a /25?

- 128 mask
- 1 bit on and 7 bits off (10000000)
- Block size of 128
- Subnets 0 and 128
- 2 subnets, each with 126 hosts

/26 What do we know about a /26?

- 192 mask
- 2 bits on and 6 bits off (11000000)
- Block size of 64
- Subnets 0, 64, 128, 192
- subnets, each with 62 hosts

/27 What do we know about a /27?

- 224 mask
- 3 bits on and 5 bits off (11100000)
- Block size of 32
- Subnets 0, 32, 64, 96, 128, 160, 192, 224
- 8 subnets, each with 30 hosts

/28 What do we know about a /28?

- 240 mask
- 4 bits on and 4 bits off
- Block size of 16
- Subnets 0, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240
- 16 subnets, each with 14 hosts

/29 What do we know about a /29?

- 248 mask
- 5 bits on and 3 bits off
- Block size of 8
- Subnets 0, 8, 16, 24, 32, 40, 48, etc.
- 32 subnets, each with 6 hosts

/30 What do we know about a /30?

- 252 mask
- 6 bits on and 2 bits off
- Block size of 4
- Subnets 0, 4, 8, 12, 16, 20, 24, etc.
- 64 subnets, each with 2 hosts

Table 2.3 puts all of the previous information into one compact little table. You should practice writing this table out on scratch paper, and if you can do it, write it down before you start your exam!

Table 2.3 What do you know?

CIDR Notation	Mask	Bits	Block Size	Subnets	Hosts
/25	128	1 bit on and 7 bits off	128	0 and 128	2 subnets, each with 126 hosts
/26	192	2 bits on	64	0, 64, 128, 192	4 subnets,

		and 6 bits off			each with 62 hosts
/27	224	3 bits on and 5 bits off	32	0, 32, 64, 96, 128, 160, 192, 224	8 subnets, each with 30 hosts
/28	240	4 bits on and 4 bits off	16	0, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240	16 subnets, each with 14 hosts
/29	248	5 bits on and 3 bits off	8	0, 8, 16, 24, 32, 40, 48, etc.	32 subnets, each with 6 hosts
/30	252	6 bits on and 2 bits off	4	0, 4, 8, 12, 16, 20, 24, etc.	64 subnets, each with 2 hosts

Regardless of whether you have a Class A, Class B, or Class C address, the /30 mask will provide you with only two hosts, ever. As suggested by Cisco, this mask is suited almost exclusively for use on point-to-point links.

If you can memorize this “What Do We Know?” section, you’ll be much better off in your day-to-day job and in your studies. Try saying it out loud, which helps you memorize things—yes, your significant other and/or coworkers will think you’ve lost it, but they probably already do if you’re in the networking field anyway. And if you’re not yet in the networking field but are studying all this to break into it, get used to it!

It’s also helpful to write these on some type of flashcards and have people test your skill. You’d be amazed at how fast you can get subnetting down if you memorize block sizes as well as this “What Do We Know?” section.

Subnetting Class B Addresses

Before we dive into this, let’s look at all the possible Class B subnet masks first. Notice that we have a lot more possible subnet masks than we do with a Class C network address:

```
255.255.0.0    (/16)
```

```
255.255.128.0  (/17)      255.255.255.0    (/24)
255.255.192.0  (/18)      255.255.255.128  (/25)
255.255.224.0  (/19)      255.255.255.192  (/26)
255.255.240.0  (/20)      255.255.255.224  (/27)
255.255.248.0  (/21)      255.255.255.240  (/28)
255.255.252.0  (/22)      255.255.255.248  (/29)
255.255.254.0  (/23)      255.255.255.252  (/30)
```

We know the Class B network address has 16 bits available for host addressing. This means we can use up to 14 bits for subnetting because we need to leave at least 2 bits for host addressing. Using a /16 means you are not subnetting with Class B, but it *is* a mask you can use!

NOTE

By the way, do you notice anything interesting about that list of subnet values—a pattern, maybe? Ah ha! Since subnet mask bits start on the left and move to the right and bits can't be skipped, the numbers are always the same regardless of the class of address. If you haven't already, memorize this pattern!

The process of subnetting a Class B network is pretty much the same as it is for a Class C, except that you have more host bits and you start in the third octet.

Use the same subnet numbers for the third octet with Class B that you used for the fourth octet with Class C, but add a zero to the network portion and a 255 to the broadcast section in the fourth octet. The following table shows you an example host range of two subnets used in a Class B 240 (/20) subnet mask:

Subnet address	**16.0**	**32.0**
Broadcast address	31.255	47.255

Just add the valid hosts between the numbers and you're set!

NOTE

The preceding example is true only until you get up to /24. After that, it's numerically exactly like Class C.

Subnetting Practice Examples: Class B Addresses

The following sections will give you an opportunity to practice subnetting Class B addresses. Again, I have to mention that this is the same as subnetting with Class C, except we start in the third octet—with the exact same numbers!

Practice Example #1B: 255.255.128.0 (/17)

. 172.16.0.0 = Network address

. 255.255.128.0 = Subnet mask

- *Subnets?* 2^1 = 2 (same amount as Class C).
- *Hosts?* 2^{15} – 2 = 32,766 (7 bits in the third octet, and 8 in the fourth).
- *Valid subnets?* 256 – 128 = 128. 0, 128. Remember that subnetting is performed in the third octet, so the subnet numbers are really 0.0 and 128.0, as shown in the next table. These are the exact numbers we used with Class C; we use them in the third octet and add a 0 in the fourth octet for the network address.
- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows the two subnets available, the valid host range, and the broadcast address of each:

Subnet	**0.0**	**128.0**
First host	0.1	128.1

Last host	127.254	255.254
Broadcast	127.255	255.255

Okay, notice that we just added the fourth octet's lowest and highest values and came up with the answers. And again, it's done exactly the same way as for a Class C subnet. We just used the same numbers in the third octet and added 0 and 255 in the fourth octet—pretty simple, huh? I really can't say this enough: it's just not that hard. The numbers never change; we just use them in different octets!

Question: Using the previous subnet mask, do you think 172.16.10.0 is a valid host address? What about 172.16.10.255? Can 0 and 255 in the fourth octet ever be a valid host address? The answer is absolutely, yes, those are valid hosts! Any number between the subnet number and the broadcast address is always a valid host.

Practice Example #2B: 255.255.192.0 (/18)

. 172.16.0.0 = Network address

. 255.255.192.0 = Subnet mask

- *Subnets?* 2^2 = 4.
- *Hosts?* 2^{14} – 2 = 16,382 (6 bits in the third octet, and 8 in the fourth).
- *Valid subnets?* 256 – 192 = 64. 0, 64, 128, 192. Remember that the subnetting is performed in the third octet, so the subnet numbers are really 0.0, 64.0, 128.0, and 192.0, as shown in the next table.
- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows the four subnets available, the valid host range, and the broadcast address of each:

Subnet	**0.0**	**64.0**	**128.0**	**192.0**
First host	0.1	64.1	128.1	192.1
Last host	63.254	127.254	191.254	255.254

Broadcast	63.255	127.255	191.255	255.255

Again, it's pretty much the same as it is for a Class C subnet—we just added 0 and 255 in the fourth octet for each subnet in the third octet.

Practice Example #3B: 255.255.240.0 (/20)

172.16.0.0 = Network address

255.255.240.0 = Subnet mask

- *Subnets?* 2^4 = 16.
- *Hosts?* 2^{12} – 2 = 4094.
- *Valid subnets?* 256 – 240 = 0, 16, 32, 48, etc., up to 240. Notice that these are the same numbers as a Class C 240 mask—we just put them in the third octet and add a 0 and 255 in the fourth octet.
- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows the first four subnets, valid hosts, and broadcast addresses in a Class B 255.255.240.0 mask:

Subnet	**0.0**	**16.0**	**32.0**	**48.0**
First host	0.1	16.1	32.1	48.1
Last host	15.254	31.254	47.254	63.254
Broadcast	15.255	31.255	47.255	63.255

Practice Example #4B: 255.255.248.0 (/21)

172.16.0.0 = Network address

255.255.248.0 = Subnet mask

- *Subnets?* 2^5 = 32.
- *Hosts?* 2^{11} – 2 = 2046.
- *Valid subnets?* 256 – 248 = 0, 8, 16, 24, 32, etc., up to 248.

- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows the first five subnets, valid hosts, and broadcast addresses in a Class B 255.255.248.0 mask:

Subnet	**0.0**	**8.0**	**16.0**	**24.0**	**32.0**
First host	0.1	8.1	16.1	24.1	32.1
Last host	7.254	15.254	23.254	31.254	39.254
Broadcast	7.255	15.255	23.255	31.255	39.255

Practice Example #5B: 255.255.252.0 (/22)

. 172.16.0.0 = Network address

. 255.255.252.0 = Subnet mask

- *Subnets?* 2^6 = 64.
- *Hosts?* 2^{10} – 2 = 1022.
- *Valid subnets?* 256 – 252 = 0, 4, 8, 12, 16, etc., up to 252.
- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows the first five subnets, valid hosts, and broadcast addresses in a Class B 255.255.252.0 mask:

Subnet	**0.0**	**4.0**	**8.0**	**12.0**	**16.0**
First host	0.1	4.1	8.1	12.1	16.1
Last host	3.254	7.254	11.254	15.254	19.254
Broadcast	3.255	7.255	11.255	15.255	19.255

Practice Example #6B: 255.255.254.0 (/23)

. 172.16.0.0 = Network address

. 255.255.254.0 = Subnet mask

- *Subnets?* 2^7 = 128.
- *Hosts?* 2^9 – 2 = 510.
- *Valid subnets?* 256 – 254 = 0, 2, 4, 6, 8, etc., up to 254.
- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows the first five subnets, valid hosts, and broadcast addresses in a Class B 255.255.254.0 mask:

Subnet	**0.0**	**2.0**	**4.0**	**6.0**	**8.0**
First host	0.1	2.1	4.1	6.1	8.1
Last host	1.254	3.254	5.254	7.254	9.254
Broadcast	1.255	3.255	5.255	7.255	9.255

Practice Example #7B: 255.255.255.0 (/24)

Contrary to popular belief, 255.255.255.0 used with a Class B network address is not called a Class B network with a Class C subnet mask. It's amazing how many people see this mask used in a Class B network and think it's a Class C subnet mask. This is a Class B subnet mask with 8 bits of subnetting—it's logically different from a Class C mask. Subnetting this address is fairly simple:

. 172.16.0.0 = Network address

. 255.255.255.0 = Subnet mask

- *Subnets?* 2^8 = 256.
- *Hosts?* 2^8 – 2 = 254.
- *Valid subnets?* 256 – 255 = 1. 0, 1, 2, 3, etc., all the way to 255.
- *Broadcast address for each subnet?*

- *Valid hosts?*

The following table shows the first four and last two subnets, the valid hosts, and the broadcast addresses in a Class B 255.255.255.0 mask:

Subnet	**0.0**	**1.0**	**2.0**	**3.0**	**...**	**254.0**	**255.0**
First host	0.1	1.1	2.1	3.1	...	254.1	255.1
Last host	0.254	1.254	2.254	3.254	...	254.254	255.254
Broadcast	0.255	1.255	2.255	3.255	...	254.255	255.255

Practice Example #8B: 255.255.255.128 (/25)

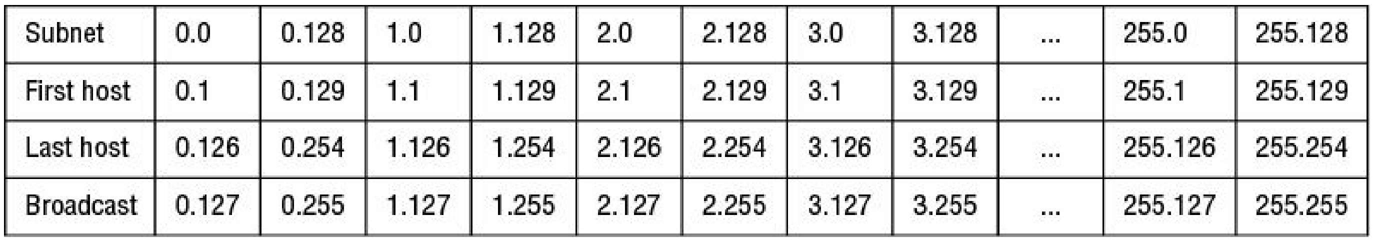
This is actually one of the hardest subnet masks you can play with. And worse, it actually is a really good subnet to use in production because it creates over 500 subnets with 126 hosts for each subnet—a nice mixture. So, don’t skip over it!

172.16.0.0 = Network address

255.255.255.128 = Subnet mask

- *Subnets?* 2^9 = 512.
- *Hosts?* 2^7 – 2 = 126.
- *Valid subnets?* Now for the tricky part. 256 – 255 = 1. 0, 1, 2, 3, etc., for the third octet. But you can’t forget the one subnet bit used in the fourth octet. Remember when I showed you how to figure one subnet bit with a Class C mask? You figure this the same way. You actually get two subnets for each third octet value, hence the 512 subnets. For example, if the third octet is showing subnet 3, the two subnets would actually be 3.0 and 3.128.
- *Broadcast address for each subnet?* The numbers right before the next subnet.
- *Valid hosts?* The numbers between the subnet numbers and the broadcast address.

The following graphic shows how you can create subnets, valid hosts, and broadcast addresses using the Class B 255.255.255.128 subnet mask. The first eight subnets are shown, followed by the last two subnets:



Subnet	0.0	0.128	1.0	1.128	2.0	2.128	3.0	3.128	...	255.0	255.128
First host	0.1	0.129	1.1	1.129	2.1	2.129	3.1	3.129	...	255.1	255.129
Last host	0.126	0.254	1.126	1.254	2.126	2.254	3.126	3.254	...	255.126	255.254
Broadcast	0.127	0.255	1.127	1.255	2.127	2.255	3.127	3.255	...	255.127	255.255

Practice Example #9B: 255.255.255.192 (/26)

Now, this is where Class B subnetting gets easy. Since the third octet has a 255 in the mask section, whatever number is listed in the third octet is a subnet number. And now that we have a subnet number in the fourth octet, we can subnet this octet just as we did with Class C subnetting. Let's try it out:

172.16.0.0 = Network address

255.255.255.192 = Subnet mask

- *Subnets?* 2^{10} = 1024.
- *Hosts?* 2^{6} – 2 = 62.
- *Valid subnets?* 256 – 192 = 64. The subnets are shown in the following table. Do these numbers look familiar?
- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows the first eight subnet ranges, valid hosts, and broadcast addresses:

Subnet	**0.0**	**0.64**	**0.128**	**0.192**	**1.0**	**1.64**	**1.128**	**1.192**
First host	0.1	0.65	0.129	0.193	1.1	1.65	1.129	1.193
Last host	0.62	0.126	0.190	0.254	1.62	1.126	1.190	1.254
Broadcast	0.63	0.127	0.191	0.255	1.63	1.127	1.191	1.255

Notice that for each subnet value in the third octet, you get subnets 0, 64, 128, and 192 in the fourth octet.

Practice Example #10B: 255.255.255.224 (/27)

This one is done the same way as the preceding subnet mask, except that we just have more subnets and fewer hosts per subnet available.

- 172.16.0.0 = Network address
- 255.255.255.224 = Subnet mask

- *Subnets?* 2^{11} = 2048.
- *Hosts?* 2^5 – 2 = 30.
- *Valid subnets?* 256 – 224 = 32. 0, 32, 64, 96, 128, 160, 192, 224.
- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows the first eight subnets:

Subnet	**0.0**	**0.32**	**0.64**	**0.96**	**0.128**	**0.160**	**0.192**	**0.224**
First host	0.1	0.33	0.65	0.97	0.129	0.161	0.193	0.225
Last host	0.30	0.62	0.94	0.126	0.158	0.190	0.222	0.254
Broadcast	0.31	0.63	0.95	0.127	0.159	0.191	0.223	0.255

This next table shows the last eight subnets:

Subnet	**255.0**	**255.32**	**255.64**	**255.96**	**255.128**	**255.160**	**255.192**	**25**
First host	255.1	255.33	255.65	255.97	255.129	255.161	255.193	25
Last host	255.30	255.62	255.94	255.126	255.158	255.190	255.222	25
Broadcast	255.31	255.63	255.95	255.127	255.159	255.191	255.223	25

Subnetting in Your Head: Class B Addresses

Are you nuts? Subnet Class B addresses in our heads? It's actually easier than writing it out—I'm not kidding! Let me show you how:

Question: What is the subnet and broadcast address of the subnet in which 172.16.10.33 /27 resides?

Answer: The interesting octet is the fourth one. 256 – 224 = 32. 32 + 32 = 64. You've got it: 33 is between 32 and 64. But remember that the third octet is considered part of the subnet, so the answer would be the 10.32 subnet. The broadcast is 10.63, since 10.64 is the next subnet. That was a pretty easy one.

Question: What subnet and broadcast address is the IP address 172.16.66.10 255.255.192.0 (/18) a member of?

Answer: The interesting octet here is the third octet instead of the fourth one. 256 – 192 = 64. 0, 64, 128. The subnet is 172.16.64.0. The broadcast must be 172.16.127.255 since 128.0 is the next subnet.

Question: What subnet and broadcast address is the IP address 172.16.50.10 255.255.224.0 (/19) a member of?

Answer: 256 – 224 = 0, 32, 64 (remember, we always start counting at 0). The subnet is 172.16.32.0, and the broadcast must be 172.16.63.255 since 64.0 is the next subnet.

Question: What subnet and broadcast address is the IP address 172.16.46.255 255.255.240.0 (/20) a member of?

Answer: 256 – 240 = 16. The third octet is important here: 0, 16, 32, 48. This subnet address must be in the 172.16.32.0 subnet, and the broadcast must be 172.16.47.255 since 48.0 is the next subnet. So, yes, 172.16.46.255 is a valid host.

Question: What subnet and broadcast address is the IP address 172.16.45.14 255.255.255.252 (/30) a member of?

Answer: Where is our interesting octet? 256 – 252 = 0, 4, 8, 12, 16—the fourth. The subnet is 172.16.45.12, with a broadcast of 172.16.45.15 because the next subnet is 172.16.45.16.

Question: What is the subnet and broadcast address of the host 172.16.88.255/20?

Answer: What is a /20 written out in dotted decimal? If you can't answer this, you can't answer this question, can you? A /20 is 255.255.240.0, gives us a block size of 16 in the third octet, and since no subnet bits are on in the fourth octet, the answer is always 0 and 255 in the fourth octet: 0, 16, 32, 48, 64, 80, 96. Because 88 is between 80 and 96, the subnet is 80.0 and the broadcast address is 95.255.

Question: A router receives a packet on an interface with a destination address of 172.16.46.191/26. What will the router do with this packet?

Answer: Discard it. Do you know why? 172.16.46.191/26 is a 255.255.255.192 mask, which gives us a block size of 64. Our subnets are then 0, 64, 128 and 192. 191 is the broadcast address of the 128 subnet, and by default, a router will discard any *broadcast* packets.

Subnetting Class A Addresses

You don't go about Class A subnetting any differently than Classes B and C, but there are 24 bits to play with instead of the 16 in a Class B address and the 8 in a Class C address.

Let's start by listing all the Class A masks:

```
255.0.0.0    (/8)

255.128.0.0  (/9)            255.255.240.0  (/20)

255.192.0.0  (/10)           255.255.248.0  (/21)

255.224.0.0  (/11)           255.255.252.0  (/22)

255.240.0.0  (/12)           255.255.254.0  (/23)

255.248.0.0  (/13)           255.255.255.0  (/24)

255.252.0.0  (/14)           255.255.255.128  (/25)

255.254.0.0  (/15)           255.255.255.192  (/26)

255.255.0.0  (/16)           255.255.255.224  (/27)

255.255.128.0  (/17)         255.255.255.240  (/28)

255.255.192.0  (/18)         255.255.255.248  (/29)

255.255.224.0  (/19)         255.255.255.252  (/30)
```

That's it. You must leave at least 2 bits for defining hosts. I hope you can see the pattern by now. Remember, we're going to do this the same way as a Class B or C subnet. It's just that, again, we simply have more host bits and we just use the same subnet numbers we used with Class B and C, but we start using these numbers in the second octet. However, the reason Class A addresses are so popular to implement is because they give the most flexibility. You can subnet in the second, third or fourth octet. I'll show you this in the next examples.

Subnetting Practice Examples: Class A Addresses

When you look at an IP address and a subnet mask, you must be able to distinguish the bits used for subnets from the bits used for determining hosts. This is imperative.

Practice Example #1A: 255.255.0.0 (/16)

Class A addresses use a default mask of 255.0.0.0, which leaves 22 bits for subnetting because you must leave 2 bits for host addressing. The 255.255.0.0 mask with a Class A address is using 8 subnet bits:

- *Subnets?* 2^8 = 256.
- *Hosts?* 2^{16} – 2 = 65,534.
- *Valid subnets?* What is the interesting octet? 256 – 255 = 1. 0, 1, 2, 3, etc. (all in the second octet). The subnets would be 10.0.0.0, 10.1.0.0, 10.2.0.0, 10.3.0.0, etc., up to 10.255.0.0.
- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows the first two and the last two subnets, the valid host range and the broadcast addresses for the private Class A 10.0.0.0 network:

Subnet	**10.0.0.0**	**10.1.0.0**	**...**	**10.254.0.0**	**10.255.0.0**
First host	10.0.0.1	10.1.0.1	...	10.254.0.1	10.255.0.1
Last host	10.0.255.254	10.1.255.254	...	10.254.255.254	10.255.255.254
Broadcast	10.0.255.255	10.1.255.255	...	10.254.255.255	10.255.255.255

Practice Example #2A: 255.255.240.0 (/20)

. 255.255.240.0 gives us 12 bits of subnetting and leaves us 12 bits for host addressing.

- *Subnets?* 2^{12} = 4096.
- *Hosts?* 2^{12} – 2 = 4094.
- *Valid subnets?* What is your interesting octet? 256 – 240 = 16. The subnets in the second octet are a block size of 1 and the subnets in the third octet are 0, 16, 32, etc.
- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows some examples of the host ranges—the first three subnets and the last subnet:

Subnet	**10.0.0.0**	**10.0.16.0**	**10.0.32.0**	**…**	**10.255.240.0**
First host	10.0.0.1	10.0.16.1	10.0.32.1	…	10.255.240.1
Last host	10.0.15.254	10.0.31.254	10.0.47.254	…	10.255.255.254
Broadcast	10.0.15.255	10.0.31.255	10.0.47.255	…	10.255.255.255

Practice Example #3A: 255.255.255.192 (/26)

Let's do one more example using the second, third, and fourth octets for subnetting:

- *Subnets?* 2^{18} = 262,144.
- *Hosts?* 2^{6} – 2 = 62.
- *Valid subnets?* In the second and third octet, the block size is 1, and in the fourth octet, the block size is 64.
- *Broadcast address for each subnet?*
- *Valid hosts?*

The following table shows the first four subnets and their valid hosts and broadcast addresses in the Class A 255.255.255.192 mask:

Subnet	**10.0.0.0**	**10.0.0.64**	**10.0.0.128**	**10.0.0.192**

First host	10.0.0.1	10.0.0.65	10.0.0.129	10.0.0.193
Last host	10.0.0.62	10.0.0.126	10.0.0.190	10.0.0.254
Broadcast	10.0.0.63	10.0.0.127	10.0.0.191	10.0.0.255

This table shows the last four subnets and their valid hosts and broadcast addresses:

Subnet	**10.255.255.0**	**10.255.255.64**	**10.255.255.128**	**10.255.255.19**
First host	10.255.255.1	10.255.255.65	10.255.255.129	10.255.255.193
Last host	10.255.255.62	10.255.255.126	10.255.255.190	10.255.255.254
Broadcast	10.255.255.63	10.255.255.127	10.255.255.191	10.255.255.255

Subnetting in Your Head: Class A Addresses

Again, I know this sounds hard, but as with Class C and Class B, the numbers are the same; we just start in the second octet. What makes this easy? You only need to worry about the octet that has the largest block size, which is typically called the interesting octet, and one that is something other than 0 or 255, such as, for example, 255.255.240.0 (/20) with a Class A network. The second octet has a block size of 1, so any number listed in that octet is a subnet. The third octet is a 240 mask, which means we have a block size of 16 in the third octet. If your host ID is 10.20.80.30, what is your subnet, broadcast address, and valid host range?

The subnet in the second octet is 20 with a block size of 1, but the third octet is in block sizes of 16, so we'll just count them out: 0, 16, 32, 48, 64, 80, 96… voilà! By the way, you can count by 16s by now, right? Good! This makes our subnet 10.20.80.0, with a broadcast address of 10.20.95.255 because the next subnet is 10.20.96.0. The valid host range is 10.20.80.1 through 10.20.95.254. And yes, no lie! You really can do this in your head if you just get your block sizes nailed!

Let's practice on one more, just for fun!

Host IP: 10.1.3.65/23

First, you can’t answer this question if you don’t know what a /23 is. It’s 255.255.254.0. The interesting octet here is the third one: 256 – 254 = 2. Our subnets in the third octet are 0, 2, 4, 6, etc. The host in this question is in subnet 2.0, and the next subnet is 4.0, so that makes the broadcast address 3.255. And any address between 10.1.2.1 and 10.1.3.254 is considered a valid host.

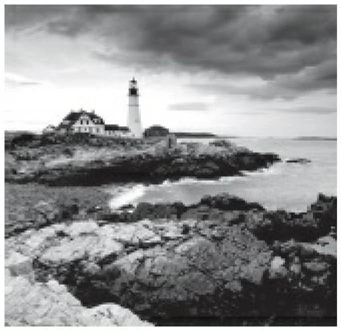
Summary

You probably really did get lost a couple of times. No worries because as I told you, that's what usually happens. Don't waste time feeling bad if you have to read each chapter more than once, or even 10 times, before you're truly good to go. If you do have to read the chapters more than once, you'll be seriously better off in the long run even if you were pretty comfortable the first time through!

This chapter provided you with an important understanding of IP subnetting—the painless way! And when you've got the key material presented in this chapter really nailed down, you should be able to subnet IP addresses in your head.

This chapter is extremely essential to your Cisco certification process, so if you just skimmed it, please go back, read it thoroughly, and don't forget to do all the written labs too!

Chapter 3
VLSMs, Summarization, and Troubleshooting TCP/IP



Now that IP addressing and subnetting have been thoroughly covered in the last two chapters, you're fully prepared and ready to learn all about variable length subnet masks (VLSMs). I'll also show you how to design and implement a network using VLSM in this chapter. After ensuring you've mastered VLSM design and implementation, I'll demonstrate how to summarize classful boundaries.

We'll wrap up the chapter by going over IP address troubleshooting, focusing on the steps Cisco recommends to follow when troubleshooting an IP network.

So get psyched because this chapter will give you powerful tools to hone your knowledge of IP addressing and networking and seriously refine the important skills you've gained so far. So stay with me—I guarantee that your hard work will pay off! Ready? Let's go!

NOTE

To find up-to-the minute updates for this chapter, please see www.lammle.com/ccna or the book's web page at www.sybex.com/go/ccna.

Variable Length Subnet Masks (VLSMs)

Teaching you a simple way to create many networks from a large single network using subnet masks of different lengths in various kinds of network designs is what my primary focus will be in this chapter. Doing this is called VLSM networking, and it brings up another important subject I mentioned in the previous chapter, “Easy Subnetting,” classful and classless networking.

Older routing protocols like Routing Information Protocol version 1 (RIPv1) do not have a field for subnet information, so the subnet information gets dropped. This means that if a router running RIP has a subnet mask of a certain value, it assumes that *all* interfaces within the classful address space have the same subnet mask. This is called classful routing, and RIP is considered a classful routing protocol. We’ll cover RIP and the difference between classful and classless networks later on in Chapter 9, “IP Routing,” but for now, just remember that if you try to mix and match subnet mask lengths in a network that’s running an old routing protocol, such as RIP, it just won’t work!

However, classless routing protocols do support the advertisement of subnet information, which means you can use VLSM with routing protocols such as RIPv2, Enhanced Interior Gateway Protocol (EIGRP), and Open Shortest Path First (OSPF). The benefit of this type of network is that it saves a bunch of IP address space.

As the name suggests, VLSMs can use subnet masks with different lengths for different router interfaces. Check out Figure 3.1 to see an example of why classful network designs are inefficient.

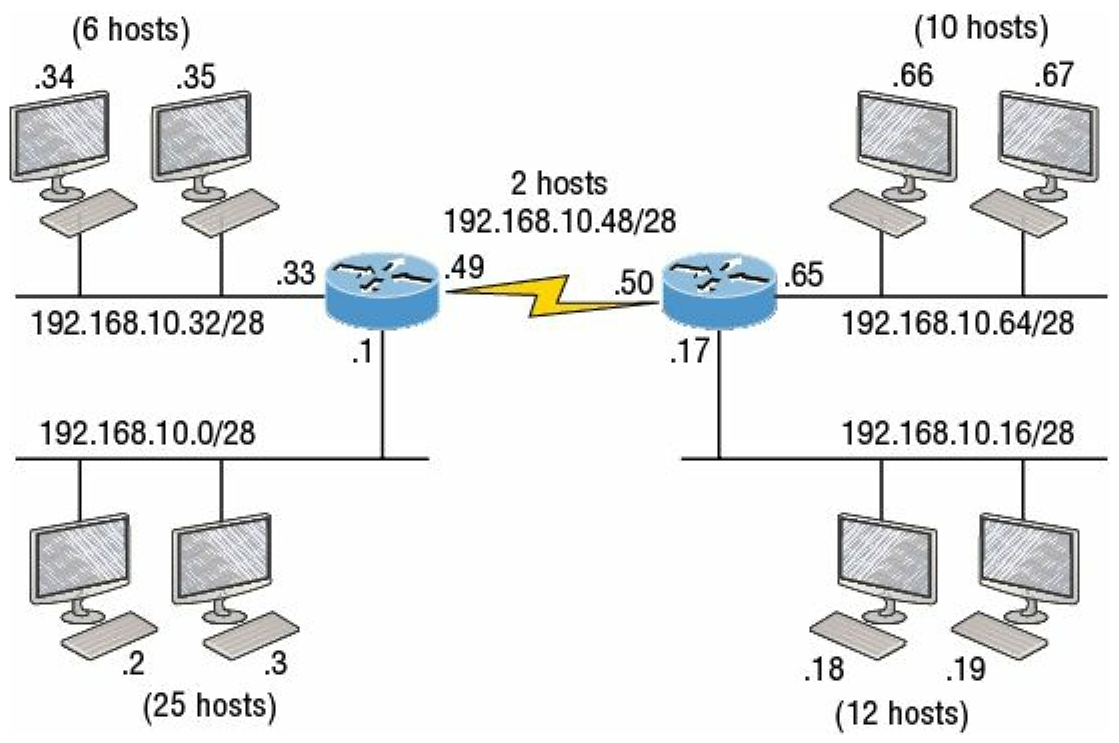


Figure 3.1 Typical classful network

Looking at Figure 3.1, you can see that there are two routers, each with two LANs and connected together with a WAN serial link. In a typical classful network design that's running RIP, you could subnet a network like this:

- 192.168.10.0 = Network
- 255.255.255.240 (/28) = Mask

Our subnets would be—you know this part, right?— 0, 16, 32, 48, 64, 80, etc., which allows us to assign 16 subnets to our internetwork. But how many hosts would be available on each network? Well, as you know by now, each subnet provides only 14 hosts, so each LAN has only 14 valid hosts available (don't forget that the router interface needs an address too and is included in the amount of needed valid hosts). This means that one LAN doesn't even have enough addresses needed for all the hosts, and this network as it is shown would not work as addressed in the figure! Since the point-to-point WAN link also has 14 valid hosts, it would be great to be able to nick a few valid hosts from that WAN link to give to our LANs!

All hosts and router interfaces have the same subnet mask—again, known as classful routing—and if we want this network to be efficient, we would definitely need to add different masks to each router interface.

But that's not our only problem—the link between the two routers will never use more than two valid hosts! This wastes valuable IP address space, and it's the big reason you need to learn about VLSM network design.

VLSM Design

Let’s take Figure 3.1 and use a classless design instead, which will become the new network shown in Figure 3.2. In the previous example, we wasted address space—one LAN didn’t have enough addresses because every router interface and host used the same subnet mask. Not so good. A better solution would be to provide for only the needed number of hosts on each router interface, and we’re going to use VLSMs to achieve that goal.

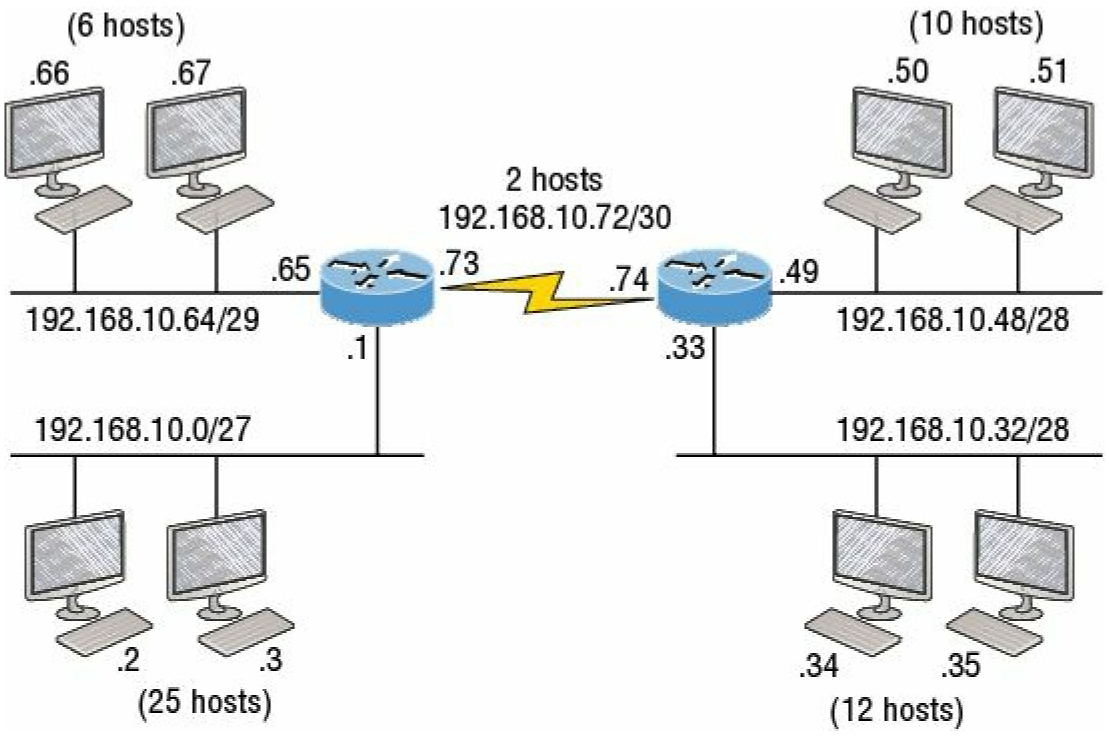


Figure 3.2 Classless network design

Now remember that we can use different size masks on each router interface. If we use a /30 on our WAN links and a /27, /28, and /29 on our LANs, we’ll get 2 hosts per WAN interface and 30, 14, and 6 hosts per LAN interface—nice (remember to count your router interface as a host)! This makes a huge difference—not only can we get just the right amount of hosts on each LAN, we still have room to add more WANs and LANs using this same network!

NOTE

To implement a VLSM design on your network, you need to have a routing protocol that sends subnet mask information with the route updates. The protocols that do that are RIPv2, EIGRP, and OSPF. Remember, RIPv1 will not work in classless networks, so it's considered a classful routing protocol.

Implementing VLSM Networks

To create VLSMs quickly and efficiently, you need to understand how block sizes and charts work together to create the VLSM masks. Table 3.1shows you the block sizes used when creating VLSMs with Class C networks. For example, if you need 25 hosts, then you'll need a block size of 32. If you need 11 hosts, you'll use a block size of 16. Need 40 hosts? Then you'll need a block of 64. You cannot just make up block sizes—they've got to be the block sizes shown in Table 3.1. So memorize the block sizes in this table—it's easy. They're the same numbers we used with subnetting!

Table 3.1 Block sizes

Prefix	**Mask**	**Hosts**	**Block Size**
/25	128	126	128
/26	192	.62	.64
/27	224	.30	.32
/28	240	.14	.16
/29	248	. 6	. 8
/30	252	. 2	. 4

The next step is to create a VLSM table. Figure 3.3 shows you the table used in creating a VLSM network. The reason we use this table is so we don't accidentally overlap networks.

Subnet	Mask	Subnets	Hosts	Block
/25	128	2	126	128
/26	192	4	62	64
/27	224	8	30	32
/28	240	16	14	16
/29	248	32	6	8
/30	252	64	2	4

Network	Hosts	Block	Subnet	Mask
A				
B				
C				
D				
E				
F				
G				
H				
I				
J				
K				
L				

0	
4	
8	
12	
16	
20	
24	
28	
32	
36	
40	
44	
48	
52	
56	
60	
64	
68	
72	
76	
80	
84	
88	
92	
96	
100	
104	
108	
112	
116	
120	
124	
128	
132	
136	
140	
144	
148	
152	
156	
160	
164	
168	
172	
176	
180	
184	
188	
192	
196	
200	
204	
208	
212	
216	
220	
224	
228	
232	
236	
240	
244	
248	
252	
256	

Figure 3.3 The VLSM table

You'll find the sheet shown in Figure 3.3 very valuable because it lists every block size you can use for a network address. Notice that the block sizes start at 4 and advance all the way up to a block size of 128. If you have two networks with block sizes of 128, you can have only 2 networks. With a block size of 64, you can have only 4, and so on, all the way to 64 networks using a block size of 4. Of course, this is assuming you're using the `ip subnet-zero` command in your network design.

So now all you need to do is fill in the chart in the lower-left corner, then add the subnets to the worksheet and you're good to go!

Based on what you've learned so far about block sizes and the VLSM table, let's create aVLSM network using a Class C network address 192.168.10.0 for the network in Figure 3.4, then fill out the VLSM table, as shown in Figure 3.5.

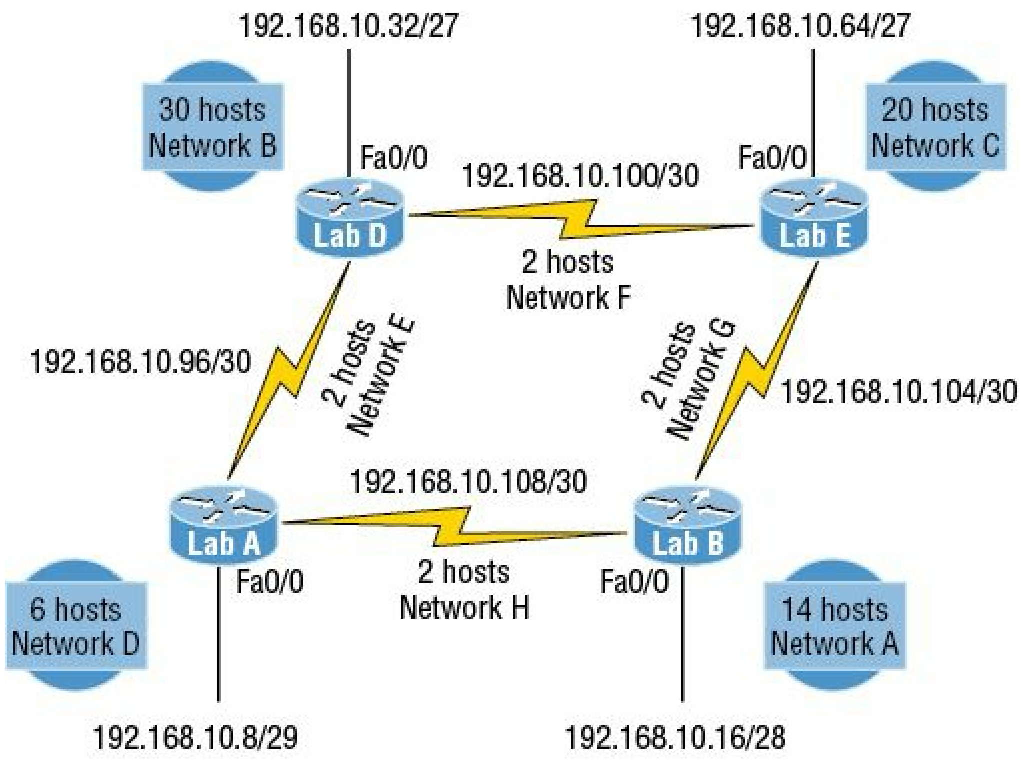
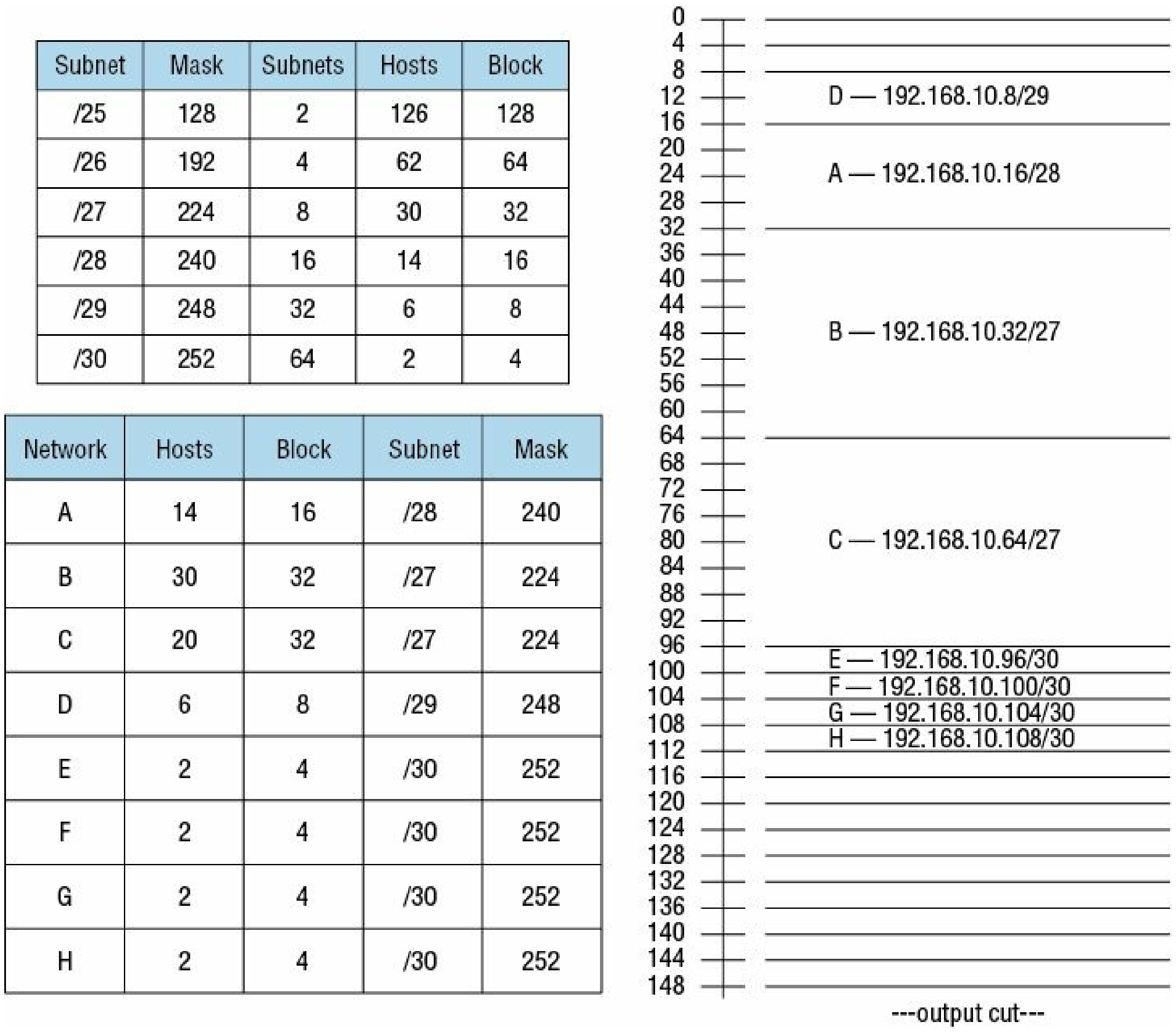


Figure 3.4 VLSM network example 1



Subnet	Mask	Subnets	Hosts	Block
/25	128	2	126	128
/26	192	4	62	64
/27	224	8	30	32
/28	240	16	14	16
/29	248	32	6	8
/30	252	64	2	4

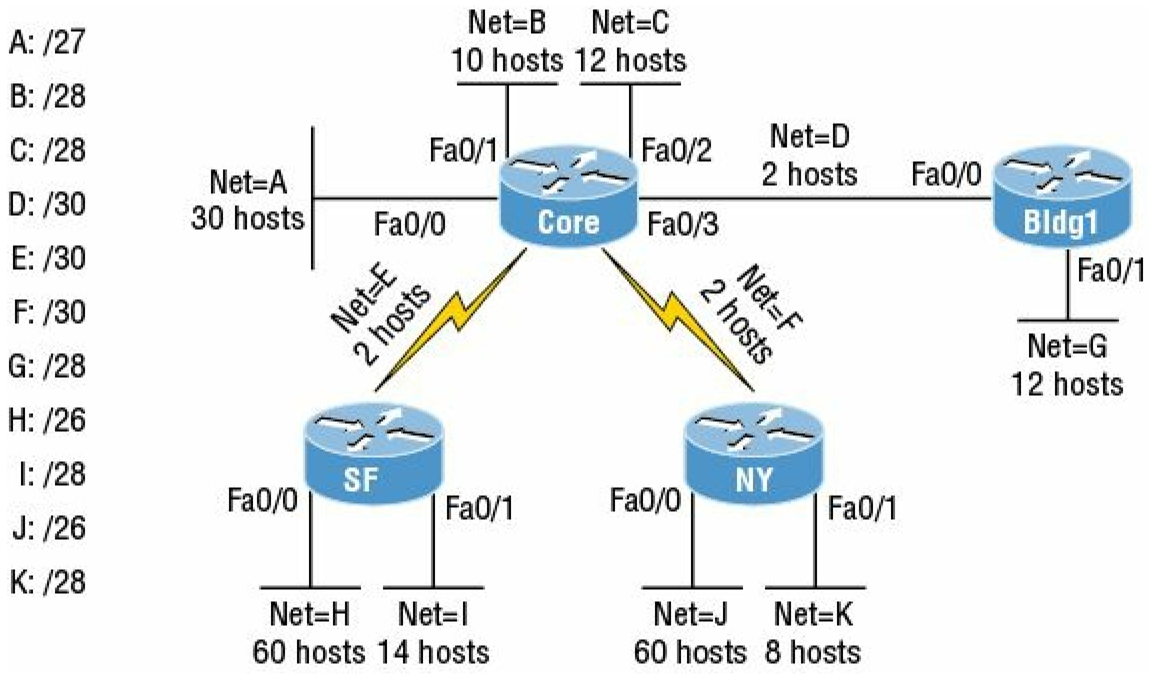
Network	Hosts	Block	Subnet	Mask
A	14	16	/28	240
B	30	32	/27	224
C	20	32	/27	224
D	6	8	/29	248
E	2	4	/30	252
F	2	4	/30	252
G	2	4	/30	252
H	2	4	/30	252

Figure 3.5 VLSM table example 1

In Figure 3.4, we have four WAN links and four LANs connected together, so we need to create a VLSM network that will save address space. Looks like we have two block sizes of 32, a block size of 16, and a block size of 8, and our WANs each have a block size of 4. Take a look and see how I filled out our VLSM chart in Figure 3.5.

There are two important things to note here. The first is that we still have plenty of room for growth with this VLSM network design. The second point is that we could never achieve this goal with one subnet mask using classful routing.

Let's do another one. Figure 3.6 shows a network with 11 networks, two block sizes of 64, one of 32, five of 16, and three of 4.



[**Figure 3.6**]() VLSM network example 2

First, create your VLSM table and use your block size chart to fill in the table with the subnets you need. [Figure 3.7]() shows a possible solution.

Subnet	Mask	Subnets	Hosts	Block
/25	128	2	126	128
/26	192	4	62	64
/27	224	8	30	32
/28	240	16	14	16
/29	248	32	6	8
/30	252	64	2	4

Network	Hosts	Block	Subnet	Mask
A				
B				
C				
D				
E				
F				
G				
H				
I				
J				
K				

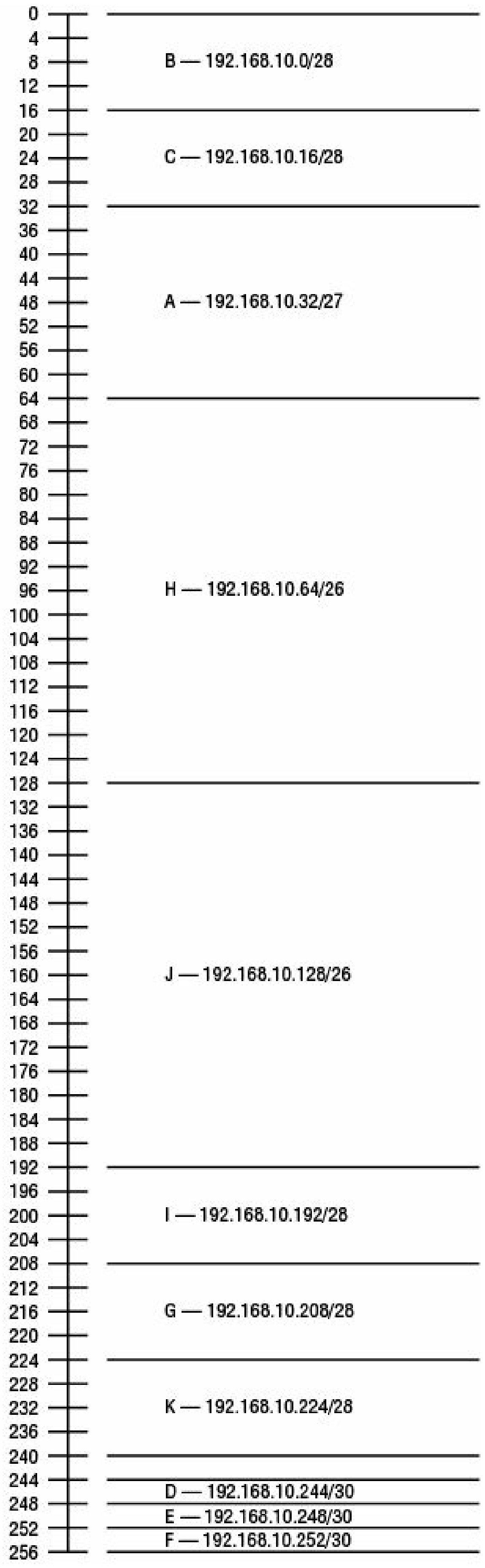


Figure 3.7 VLSM table example 2

Notice that I filled in this entire chart and only have room for one more block size of 4. You can only gain that amount of address space savings with a VLSM network!

Keep in mind that it doesn't matter where you start your block sizes as long as you always begin counting from zero. For example, if you had a block size of 16, you must start at 0 and incrementally progress from there—0, 16, 32, 48, and so on. You can't start with a block size of 16 or some value like 40, and you can't progress using anything but increments of 16.

Here's another example. If you had block sizes of 32, start at zero like this: 0, 32, 64, 96, etc. Again, you don't get to start wherever you want; you must always start counting from zero. In the example in Figure 3.7, I started at 64 and 128, with my two block sizes of 64. I didn't have much choice because my options are 0, 64, 128, and 192. However, I added the block size of 32, 16, 8, and 4 elsewhere, but they were always in the correct increments required of the specific block size. Remember that if you always start with the largest blocks first, then make your way to the smaller blocks sizes, you will automatically fall on an increment boundary. It also guarantees that you are using your address space in the most effective way.

Okay—you have three locations you need to address, and the IP network you have received is 192.168.55.0 to use as the addressing for the entire network. You'll use `ip subnet-zero` and RIPv2 as the routing protocol because RIPv2 supports VLSM networks but RIPv1 does not. Figure 3.8 shows the network diagram and the IP address of the RouterA S0/0 interface.

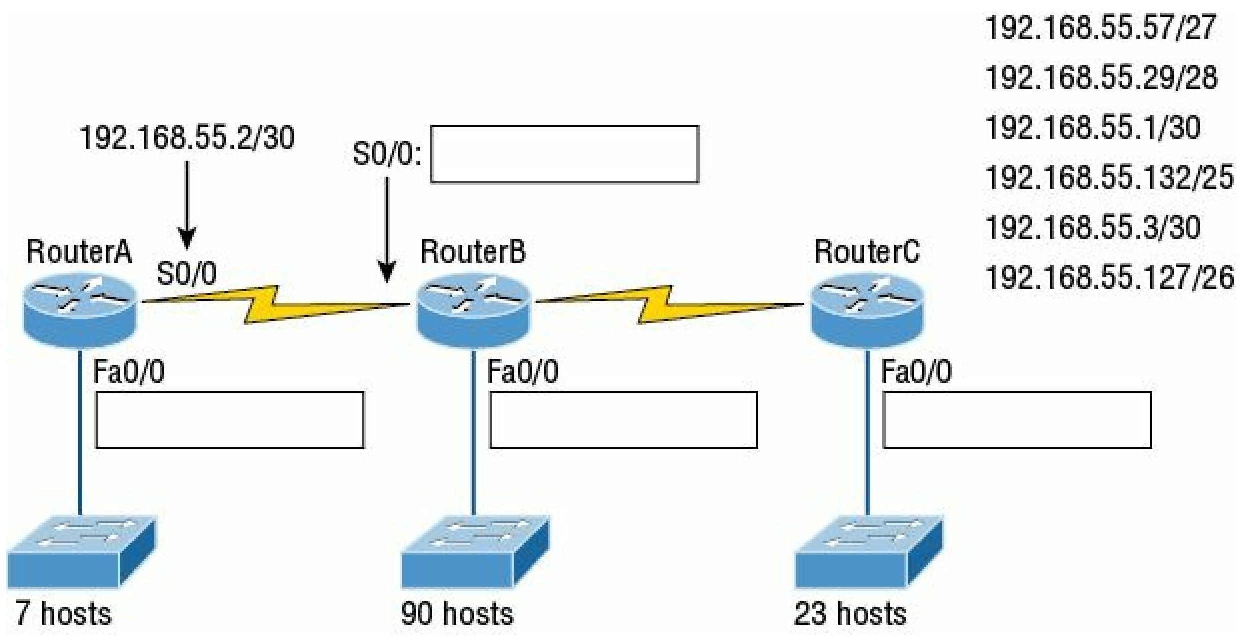


Figure 3.8 VLSM design example 1

From the list of IP addresses on the right of the figure, which IP address do you think will be placed in each router's FastEthernet 0/0 interface and serial 0/0 of RouterB?

To answer this, look for clues in Figure 3.8. The first is that interface S0/0 on RouterA has IP address 192.168.55.2/30 assigned, which makes for an easy answer because A /30 is 255.255.255.252, which gives you a block size of 4. Your subnets are 0, 4, 8, etc. Since the known host has an IP address of 2, the only other valid host in the zero subnet is 1, so the third answer down is the right one for the S0/0 interface of RouterB.

The next clues are the listed number of hosts for each of the LANs. RouterA needs 7 hosts—a block size of 16 (/28). RouterB needs 90 hosts—a block size of 128 (/25). And RouterC needs 23 hosts—a block size of 32 (/27).

Figure 3.9 illustrates this solution.

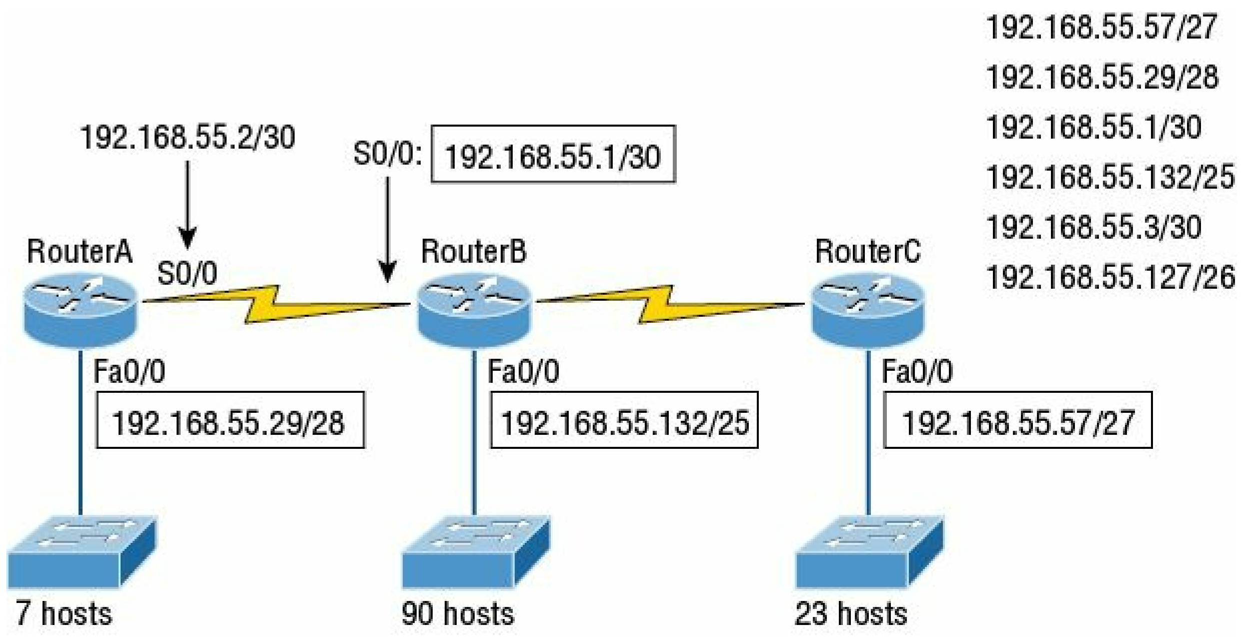


Figure 3.9 Solution to VLSM design example 1

This is actually pretty simple because once you've figured out the block size needed for each LAN, all you need to get to the right solution is to identify proper clues and, of course, know your block sizes well!

One last example of VLSM design before we move on to summarization. Figure 3.10 shows three routers, all running RIPv2. Which Class C addressing scheme would you use to maintain the needs of this network while saving as much address space as possible?

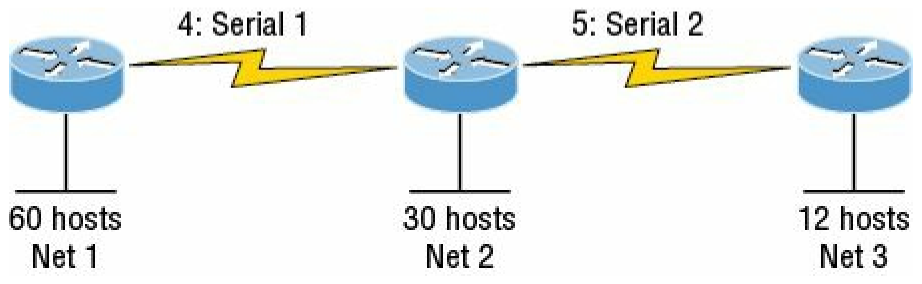


Figure 3.10 VLSM design example 2

This is actually a pretty clean network design that's just waiting for you to fill out the chart. There are block sizes of 64, 32, and 16 and two block sizes of 4. Coming up with the right solution should be a slam dunk! Take a look at my answer in Figure 3.11.

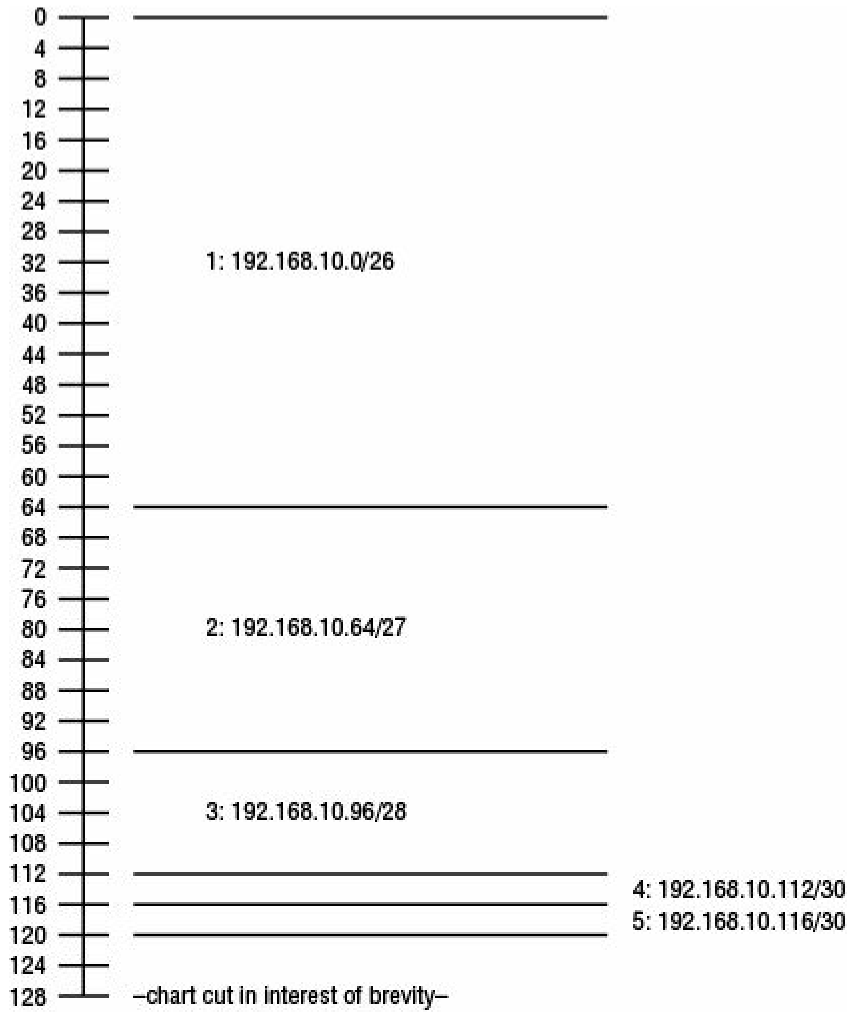


Figure 3.11 Solution to VLSM design example 2

My solution began at subnet 0, and I used the block size of 64. Clearly, I didn't have to go with a block size of 64 because I could've chosen a block size of 4 instead. But I didn't because I usually like to start with the largest block size and move to the smallest. With that done, I added the block sizes of 32 and 16 as well as the two block sizes of 4. This solution is optimal because it still leaves lots of room to add subnets to this network!

Summarization

Summarization, also called route aggregation, allows routing protocols to advertise many networks as one address. The purpose of this is to reduce the size of routing tables on routers to save memory, which also shortens the amount of time IP requires to parse the routing table when determining the best path to a remote network.

Why Bother with VLSM Design?

You have just been hired by a new company and need to add on to their existing network. There are no restrictions to prevent you from starting over with a completely new IP address scheme. Should you use a VLSM classless network or opt for a classful network?

Let's say you happen to have plenty of address space because you're using the Class A 10.0.0.0 private network address, so you really can't imagine that you'd ever run out of IP addresses. So why would you want to bother with the VLSM design process in this environment?

Good question! Here's your answer…

By creating contiguous blocks of addresses to specific areas of your network, you can then easily summarize the network and keep route updates with a routing protocol to a minimum. Why would anyone want to advertise hundreds of networks between buildings when you can just send one summary route between buildings and achieve the same result? This approach will optimize the network's performance dramatically!

To make sure this is clear, let me take a second to explain summary routes. Summarization, also called supernetting, provides route updates in the most efficient way possible by advertising many routes in one advertisement instead of individually. This saves a ton of bandwidth and minimizes router processing. As always, you need to use blocks of addresses to configure your summary routes and watch your network's performance hum along efficiently! And remember, block sizes are used in all sorts of networks anyway.

Still, it's important to understand that summarization works only if you design your network properly. If you carelessly hand out IP subnets to any location on the network, you'll quickly notice that you no longer have any summary boundaries. And you won't get very far creating summary routes without those, so watch your step!

Figure 3.12 shows how a summary address would be used in an internetwork.

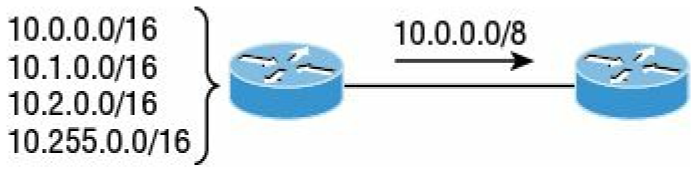


Figure 3.12 Summary address used in an internetwork

Summarization is pretty straightforward because all you really need to have down is a solid understanding of the block sizes we've been using for subnetting and VLSM design. For example, if you wanted to summarize the following networks into one network advertisement, you just have to find the block size first, which will make it easy to find your answer:

. 192.168.16.0 through network 192.168.31.0

Okay—so what's the block size? Well, there are exactly 16 Class C networks, which fit neatly into a block size of 16.

Now that we've determined the block size, we just need to find the network address and mask used to summarize these networks into one advertisement. The network address used to advertise the summary address is always the first network address in the block—in this example, 192.168.16.0. To figure out a summary mask, we just need to figure out which mask will get us a block size of 16. If you came up with 240, you got it right! 240 would be placed in the third octet, which is exactly the octet where we're summarizing, so the mask would be 255.255.240.0.

Here's another example:

. Networks 172.16.32.0 through 172.16.50.0

This isn't as clean as the previous example because there are two possible answers. Here's why: Since you're starting at network 32, your options for block sizes are 4, 8, 16, 32, 64, etc., and block sizes of 16 and 32 could work as this summary address. Let's explore your two options:

- If you went with a block size of 16, then the network address would be 172.16.32.0 with a mask of 255.255.240.0 (240 provides a block of 16). The problem is that this only summarizes from 32 to 47, which means that networks 48 through 50 would be advertised as single networks. Even so, this could still be a good solution depending on your network design.

- If you decided to go with a block size of 32 instead, then your summary address would still be 172.16.32.0, but the mask would be 255.255.224.0 (224 provides a block of 32). The possible problem with this answer is that it will summarize networks 32 through 63 and we only have networks 32 to 50. No worries if you're planning on adding networks 51 to 63 later into the same network, but you could have serious problems in your internetwork if somehow networks 51 to 63 were to show up and be advertised from somewhere else in your network! So even though this option does allow for growth, it's a lot safer to go with option #1.

Let's take a look at another example: Your summary address is 192.168.144.0/20, so what's the range of host addresses that would be forwarded according to this summary? The /20 provides a summary address of 192.168.144.0 and mask of 255.255.240.0.

The third octet has a block size of 16, and starting at summary address 144, the next block of 16 is 160, so your network summary range is 144 to 159 in the third octet. This is why it comes in handy to be able to count in 16s!

A router with this summary address in the routing table will forward any packet having destination IP addresses of 192.168.144.1 through 192.168.159.254.

Only two more summarization examples, then we'll move on to troubleshooting.

In summarization example 4, Figure 3.13, the Ethernet networks connected to router R1 are being summarized to R2 as 192.168.144.0/20. Which range of IP addresses will R2 forward to R1 according to this summary?

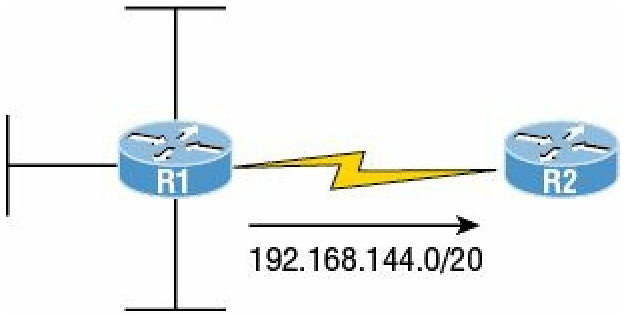


Figure 3.13 Summarization example 4

No worries—solving this is easier than it looks initially. The question actually has the summary address listed in it: 192.168.144.0/20. You already know that /20 is 255.255.240.0, which means you've got a block size of 16 in the third octet. Starting at 144, which is also right there in the question, makes the next block size of 16 equal 160. You can't go above 159 in the third octet, so the IP addresses that will be forwarded are 192.168.144.1 through 192.168.159.254.

Okay, last one. In Figure 3.14, there are five networks connected to router R1. What's the best summary address to R2?

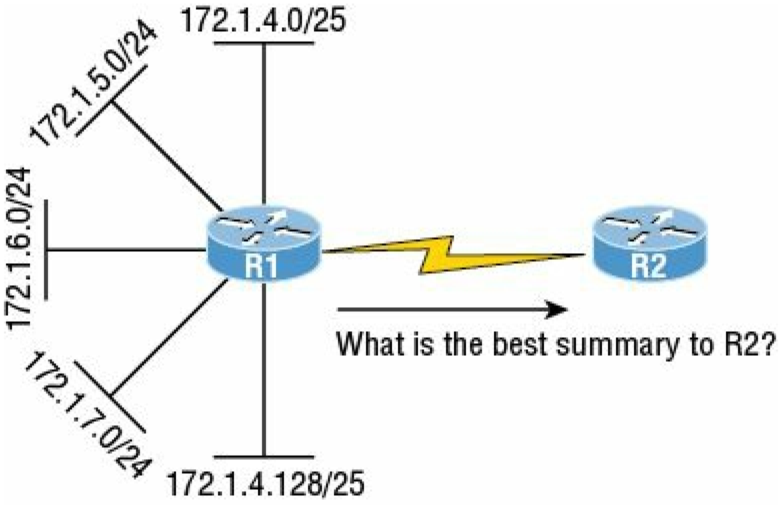


Figure 3.14 Summarization example 5

I'll be honest with you—this is a much harder question than the one in Figure 3.13, so you're going to have to look carefully to see the answer. A good approach here would be to write down all the networks and see if you can find anything in common with all of them:

- 172.1.4.128/25
- 172.1.7.0/24
- 172.1.6.0/24
- 172.1.5.0/24
- 172.1.4.0/25

Do you see an octet that looks interesting to you? I do. It's the third octet. 4, 5, 6, 7, and yes, it's a block size of 4. So you can summarize 172.1.4.0 using a mask of 255.255.252.0, meaning you would use a block size of 4 in the third octet. The IP addresses forwarded with this summary would be 172.1.4.1 through 172.1.7.254.

To summarize the summarization section, if you've nailed down your block sizes, then finding and applying summary addresses and masks is a relatively straightforward task. But you're going to get bogged down pretty quickly if you don’t know what a /20 is or if you can’t count by 16s!

Troubleshooting IP Addressing

Because running into trouble now and then in networking is a given, being able to troubleshoot IP addressing is clearly a vital skill. I'm not being negative here—just realistic. The positive side to this is that if you're the one equipped with the tools to diagnose and clear up the inevitable trouble, you get to be the hero when you save the day! Even better? You can usually fix an IP network regardless of whether you're on site or at home!

So this is where I'm going to show you the "Cisco way" of troubleshooting IP addressing. Let's use Figure 3.15 as an example of your basic IP trouble—poor Sally can't log in to the Windows server. Do you deal with this by calling the Microsoft team to tell them their server is a pile of junk and causing all your problems? Though tempting, a better approach is to first double-check and verify your network instead.

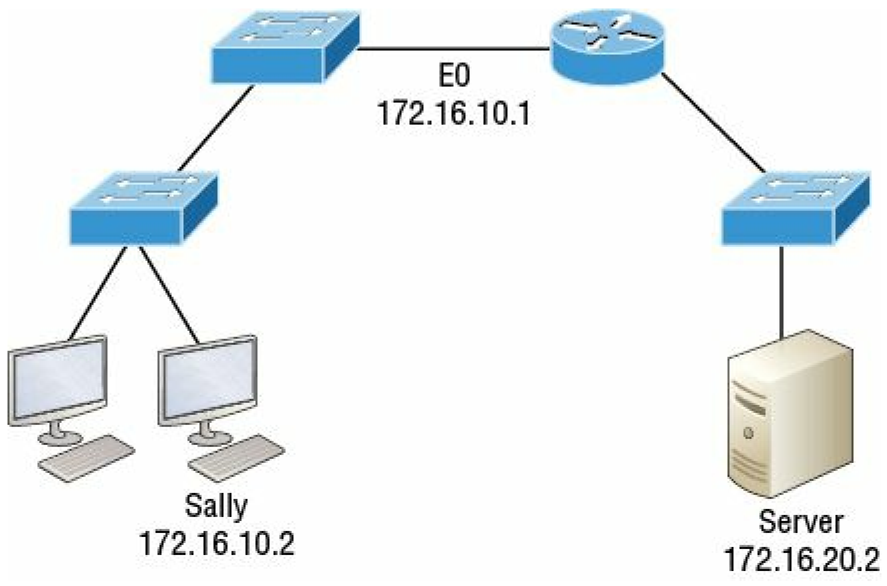


Figure 3.15 Basic IP troubleshooting

Okay, let's get started by going through the troubleshooting steps that Cisco recommends. They're pretty simple, but important nonetheless. Pretend you're at a customer host and they're complaining that they can't communicate to a server that just happens to be on a remote network. Here are the four troubleshooting steps Cisco recommends:

1. Open a Command window and ping 127.0.0.1. This is the diagnostic, or loopback, address, and if you get a successful ping, your IP stack is considered initialized. If it fails, then you have an IP stack failure and need to reinstall TCP/IP on the host.

```
C:\>ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:

Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:

     Packets: Sent &#x0003D; 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

     Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

2. From the Command window, ping the IP address of the local host (we'll assume correct configuration here, but always check the IP configuration too!). If that's successful, your network interface card (NIC) is functioning. If it fails, there is a problem with the NIC. Success here doesn't just mean that a cable is plugged into the NIC, only that the IP protocol stack on the host can communicate to the NIC via the LAN driver.

```
C:\>ping 172.16.10.2

Pinging 172.16.10.2 with 32 bytes of data:

Reply from 172.16.10.2: bytes=32 time<1ms TTL=128

Reply from 172.16.10.2: bytes=32 time<1ms TTL=128

Reply from 172.16.10.2: bytes=32 time<1ms TTL=128

Reply from 172.16.10.2: bytes=32 time<1ms TTL=128

Ping statistics for 172.16.10.2:

     Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

     Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

3. From the Command window, ping the default gateway (router). If the ping works, it means that the NIC is plugged into the network and can

communicate on the local network. If it fails, you have a local physical network problem that could be anywhere from the NIC to the router.

```
C:\>ping 172.16.10.1

Pinging 172.16.10.1 with 32 bytes of data:

Reply from 172.16.10.1: bytes=32 time<1ms TTL=128

Reply from 172.16.10.1: bytes=32 time<1ms TTL=128

Reply from 172.16.10.1: bytes=32 time<1ms TTL=128

Reply from 172.16.10.1: bytes=32 time<1ms TTL=128

Ping statistics for 172.16.10.1:

     Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

     Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

4. If steps 1 through 3 were successful, try to ping the remote server. If that works, then you know that you have IP communication between the local host and the remote server. You also know that the remote physical network is working.

```
C:\>ping 172.16.20.2

Pinging 172.16.20.2 with 32 bytes of data:

Reply from 172.16.20.2: bytes=32 time<1ms TTL=128

Reply from 172.16.20.2: bytes=32 time<1ms TTL=128

Reply from 172.16.20.2: bytes=32 time<1ms TTL=128

Reply from 172.16.20.2: bytes=32 time<1ms TTL=128

Ping statistics for 172.16.20.2:

     Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

     Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

If the user still can't communicate with the server after steps 1 through 4 have been completed successfully, you probably have some type of name resolution problem and need to check your Domain Name System (DNS) settings. But if the ping to the remote server fails, then you know you have some type of remote physical network problem and need to go to the server and work through steps 1 through 3 until you find the snag.

Before we move on to determining IP address problems and how to fix them, I just want to mention some basic commands that you can use to help troubleshoot your network from both a PC and a Cisco router. Keep in mind that though these commands may do the same thing, they're implemented differently.

`ping` Uses ICMP echo request and replies to test if a node IP stack is initialized and alive on the network.

`traceroute` Displays the list of routers on a path to a network destination by using TTL time-outs and ICMP error messages. This command will not work from a command prompt.

`tracert` Same function as `traceroute`, but it's a Microsoft Windows command and will not work on a Cisco router.

`arp -a` Displays IP-to-MAC-address mappings on a Windows PC.

`show ip arp` Same function as `arp -a`, but displays the ARP table on a Cisco router. Like the commands `traceroute` and `tracert`, `arp -a` and `show ip arp` are not interchangeable through DOS and Cisco.

`ipconfig /all` Used only from a Windows command prompt; shows you the PC network configuration.

Once you've gone through all these steps and, if necessary, used the appropriate commands, what do you do when you find a problem? How do you go about fixing an IP address configuration error? Time to cover the next step—determining and fixing the issue at hand!

Determining IP Address Problems

It's common for a host, router, or other network device to be configured with the wrong IP address, subnet mask, or default gateway. Because this happens way too often, you must know how to find and fix IP address configuration errors.

A good way to start is to draw out the network and IP addressing scheme. If that's already been done, consider yourself lucky because though sensible, it's rarely done. Even if it is, it's usually outdated or inaccurate anyway. So either way, it's a good idea to bite the bullet and start from scratch.

Once you have your network accurately drawn out, including the IP addressing scheme, you need to verify each host's IP address, mask, and default gateway address to establish the problem. Of course, this is assuming that you don't have a physical layer problem, or if you did, that you've already fixed it.

Let's check out the example illustrated in Figure 3.16.

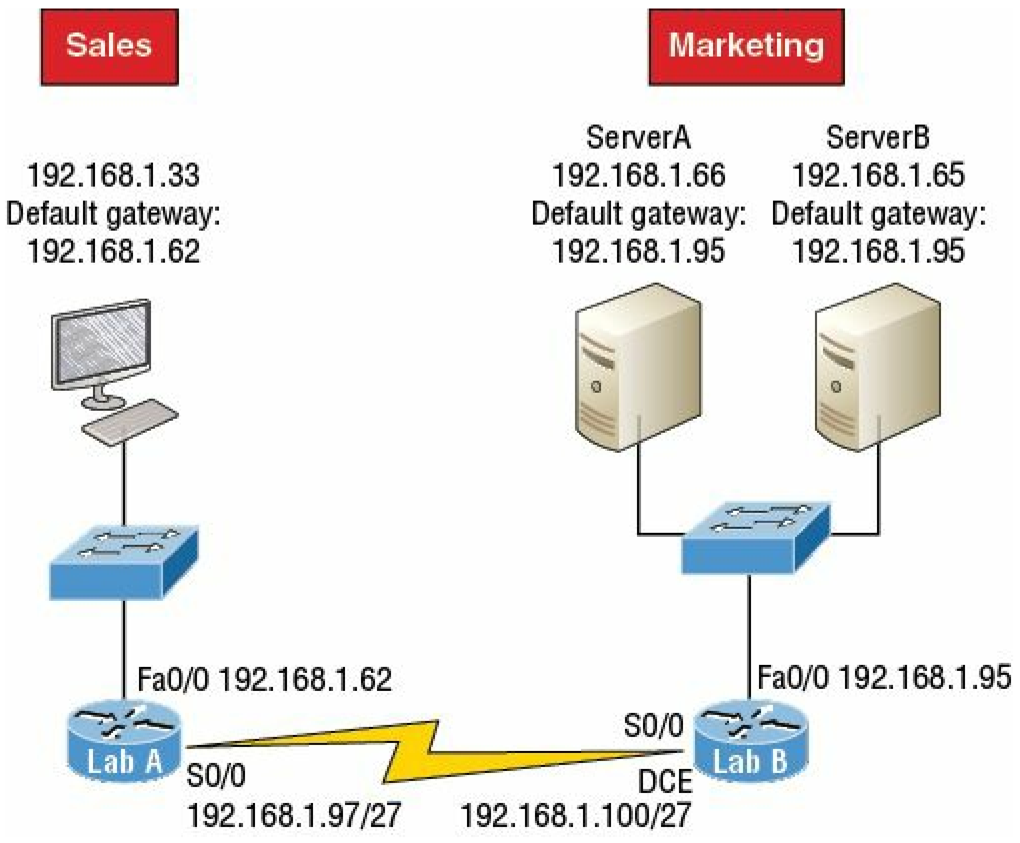


Figure 3.16 IP address problem 1

A user in the sales department calls and tells you that she can't get to ServerA in the marketing department. You ask her if she can get to ServerB in the marketing department, but she doesn't know because she doesn't have rights to log on to that server. What do you do?

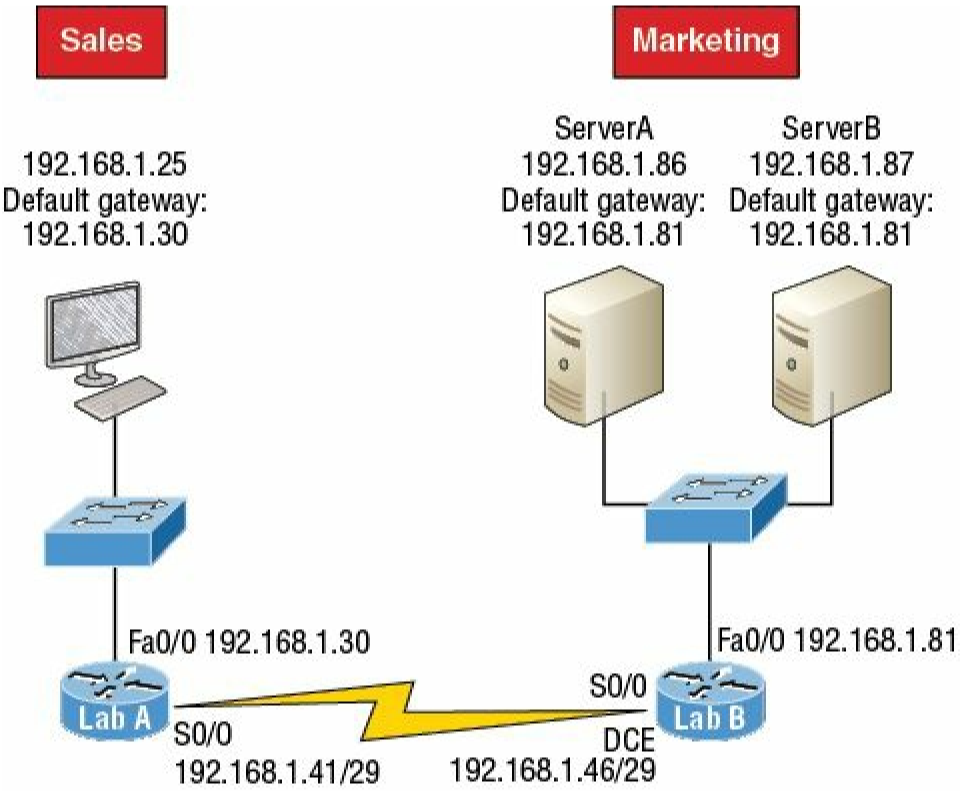
First, guide your user through the four troubleshooting steps you learned in the preceding section. Okay—let's say steps 1 through 3 work but step 4 fails. By looking at the figure, can you determine the problem? Look for clues in the network drawing. First, the WAN link between the Lab A router and the Lab B router shows the mask as a /27. You should already know that this mask is 255.255.255.224 and determine that all networks are using this mask. The network address is 192.168.1.0. What are our valid subnets and hosts? 256 – 224 = 32, so this makes our subnets 0, 32, 64, 96, 128, etc. So, by looking at the figure, you can see that subnet 32 is being used by the sales department. The WAN link is using subnet 96, and the marketing department is using subnet 64.

Now you've got to establish what the valid host ranges are for each subnet. From what you learned at the beginning of this chapter, you should now be able to easily determine the subnet address, broadcast addresses, and valid host ranges. The valid hosts for the Sales LAN are 33 through 62, and the broadcast address is 63 because the next subnet is 64, right? For the Marketing LAN, the valid hosts are 65 through 94 (broadcast 95), and for the WAN link, 97 through 126 (broadcast 127). By closely examining the figure, you can determine that the default gateway on the Lab B router is incorrect. That address is the broadcast address for subnet 64, so there's no way it could be a valid host!

TIP

If you tried to configure that address on the Lab B router interface, you'd receive a *bad mask* error. Cisco routers don't let you type in subnet and broadcast addresses as valid hosts!

Did you get all that? Let's try another one to make sure. Figure 3.17 shows a network problem.



[Figure 3.17] IP address problem 2

A user in the Sales LAN can't get to ServerB. You have the user run through the four basic troubleshooting steps and find that the host can communicate to the local network but not to the remote network. Find and define the IP addressing problem.

If you went through the same steps used to solve the last problem, you can see that first, the WAN link again provides the subnet mask to use— /29, or 255.255.255.248. Assuming classful addressing, you need to determine what the valid subnets, broadcast addresses, and valid host ranges are to solve this problem.

The 248 mask is a block size of 8 (256 – 248 = 8, as discussed in Chapter 4), so the subnets both start and increment in multiples of 8. By looking at the figure, you see that the Sales LAN is in the 24 subnet, the WAN is in the 40 subnet, and the Marketing LAN is in the 80 subnet. Can you see the problem yet? The valid host range for the Sales LAN is 25–30, and the configuration appears correct. The valid host range for the WAN link is 41–46, and this also appears correct. The valid host range for the 80 subnet is 81–86, with a broadcast address of 87 because the next subnet is 88. ServerB has been configured with the broadcast address of the subnet.

Okay, now that you can figure out misconfigured IP addresses on hosts, what do you do if a host doesn't have an IP address and you need to assign one? What you need to do is scrutinize the other hosts on the LAN and figure out the network, mask, and default gateway. Let's take a look at a couple of examples of how to find and apply valid IP addresses to hosts.

You need to assign a server and router IP addresses on a LAN. The subnet assigned on that segment is 192.168.20.24/29. The router needs to be assigned the first usable address and the server needs the last valid host ID. What is the IP address, mask, and default gateway assigned to the server?

To answer this, you must know that a /29 is a 255.255.255.248 mask, which provides a block size of 8. The subnet is known as 24, the next subnet in a block of 8 is 32, so the broadcast address of the 24 subnet is 31 and the valid host range is 25–30.

- Server IP address: 192.168.20.30
- Server mask: 255.255.255.248
- Default gateway: 192.168.20.25 (router's IP address)

Take a look at Figure 3.18 and solve this problem.

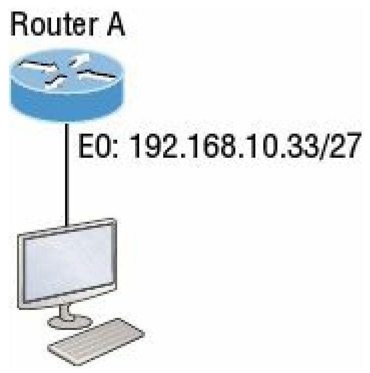


Figure 3.18 Find the valid host #1

Look at the router's IP address on Ethernet0. What IP address, subnet mask, and valid host range could be assigned to the host?

The IP address of the router's Ethernet0 is 192.168.10.33/27. As you already know, a /27 is a 224 mask with a block size of 32. The router's interface is in the 32 subnet. The next subnet is 64, so that makes the broadcast address of the 32 subnet 63 and the valid host range 33–62.

- Host IP address: 192.168.10.34–62 (any address in the range except for 33, which is assigned to the router)
- Mask: 255.255.255.224

- Default gateway: 192.168.10.33

Figure 3.19 shows two routers with Ethernet configurations already assigned. What are the host addresses and subnet masks of HostA and HostB?

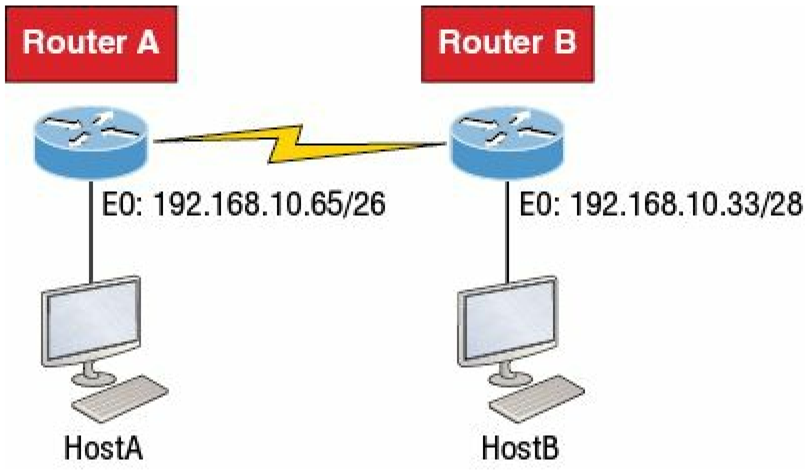


Figure 3.19 Find the valid host #2

Router A has an IP address of 192.168.10.65/26 and Router B has an IP address of 192.168.10.33/28. What are the host configurations? Router A Ethernet0 is in the 192.168.10.64 subnet and Router B Ethernet0 is in the 192.168.10.32 network.

- Host A IP address: 192.168.10.66–126
- Host A mask: 255.255.255.192
- Host A default gateway: 192.168.10.65
- Host B IP address: 192.168.10.34–46
- Host B mask: 255.255.255.240
- Host B default gateway: 192.168.10.33

Just a couple more examples before you can put this chapter behind you—hang in there!

Figure 3.20 shows two routers. You need to configure the S0/0 interface on RouterA. The IP address assigned to the serial link is 172.16.17.0/22. What IP address can be assigned?

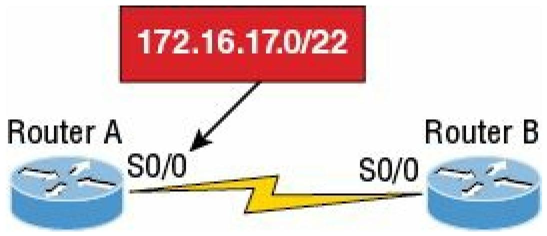


Figure 3.20 Find the valid host address #3

First, know that a /22 CIDR is 255.255.252.0, which makes a block size of 4 in the third octet. Since 17 is listed, the available range is 16.1 through 19.254, so in this example, the IP address S0/0 could be 172.16.18.255 since that's within the range.

Okay, last one! You need to find a classful network address that has one Class C network ID and you need to provide one usable subnet per city while allowing enough usable host addresses for each city specified in Figure 3.21. What is your mask?

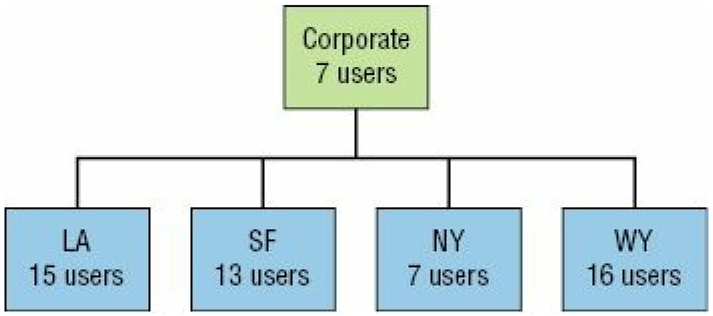


Figure 3.21 Find the valid subnet mask

Actually, this is probably the easiest thing you've done all day! I count 5 subnets needed, and the Wyoming office needs 16 users—always look for the network that needs the most hosts! What block size is needed for the Wyoming office? Your answer is 32. You can't use a block size of 16 because you always have to subtract 2. What mask provides you with a block size of 32? 224 is your answer because this provides 8 subnets, each with 30 hosts.

You're done—the diva has sung and the chicken has safely crossed the road… whew! Time to take a break, but skip the shot and the beer if that's what you had in mind because you need to have your head straight to go through the written lab and review questions next!

Summary

Again, if you got to this point without getting lost along the way a few times, you're awesome, but if you did get lost, don't stress because most people do! Just be patient with yourself and go back over the material that tripped you up until it's all crystal clear. You'll get there!

This chapter provided you with keys to understanding the oh-so-very-important topic of variable length subnet masks. You should also know how to design and implement simple VLSM networks and be clear on summarization as well.

And make sure you understand and memorize Cisco's troubleshooting methods. You must remember the four steps that Cisco recommends to take when trying to narrow down exactly where a network and/or IP addressing problem is and then know how to proceed systematically to fix it. In addition, you should be able to find valid IP addresses and subnet masks by looking at a network diagram.