Exam Topics in This Chapter

20 Describe the different classes of IP addresses (and subnetting).

21 Identify the functions of the TCP/IP network-layer protocols.

22 Identify the functions performed by ICMP.

23 Configure IP addresses.

24 Verify IP addresses.

26 Define flow control and describe the three basic methods used in networking.

CHAPTER 6

TCP/IP and IP Routing

The TCP/IP suite includes the most important protocols covered on the CCNA exam and the protocols used most often in networks today. This chapter covers the TCP/IP protocols, including IP addressing and subnetting. Cisco expects CCNAs not just to know IP addressing and routing, but also to know the concepts behind many other TCP/IP protocols. In addition, CCNAs should be able to easily recall the commands used to examine the details of IP processing in a router. Of course, Cisco also requires you to continually prove your understanding of IP subnetting on the CCNA exam and on almost all other Cisco exams.

This chapter begins by describing TCP and UDP, the two main options for OSI Layer 4 protocols in TCP/IP. Routers care about TCP and UDP because they can examine TCP and UDP headers when making filtering decisions with access lists. Also, a deep understanding of TCP helps with many other parts of networking, including topics that you will see on the CCNP exam. After TCP and UDP, a couple of other short topics, ARP and ICMP, are covered. The TCP/IP protocols require ARP and ICMP to work.

You need solid skills with IP addressing and subnetting to succeed as a network engineer or to do well on the CCNA exam. For the exam, you need to be able to answer subnetting questions quickly and confidently. In real life, you need to apply these same concepts in a variety of ways. The second section of this chapter details IP addressing and subnetting, including some tricks that make the math required to answer test questions a bit easier.

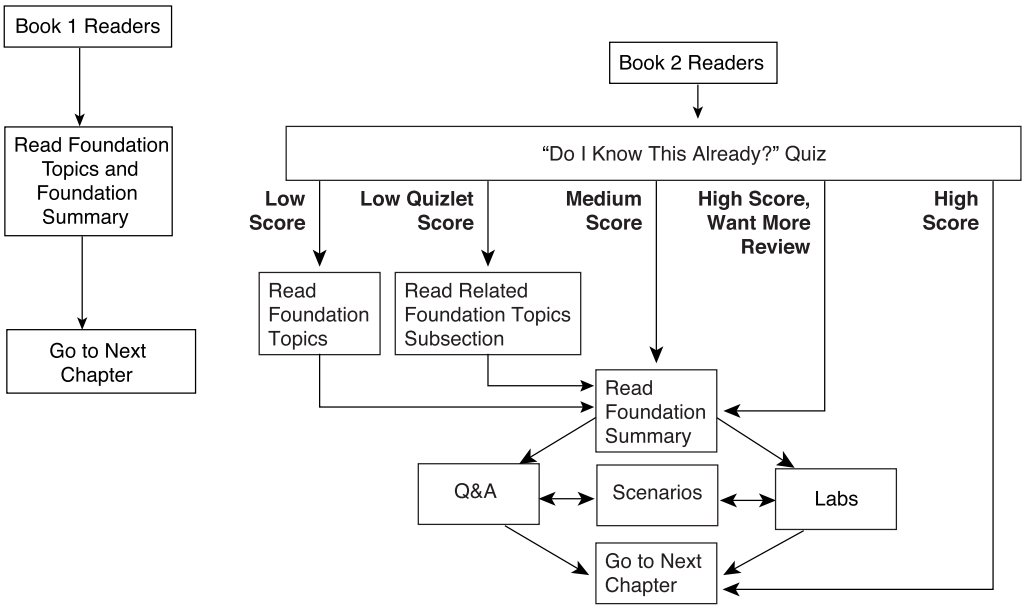
Finally, you need to be able to configure TCP/IP on a Cisco router. Actually, that part of the chapter is a bit anticlimatic—configuring IP is pretty easy. Included in that section are some additional features about how to troubleshoot and manage an IP network.

How to Best Use This Chapter

By taking the following steps, you can make better use of your study time:

- Keep your notes and the answers for all your work with this book in one place, for easy reference.
- Take the "Do I Know This Already?" quiz, and write down your answers. Studies show that retention is significantly increased through writing down facts and concepts, even if you never look at the information again.
- Use the diagram in Figure 6-1 to guide you to the next step.

Figure 6-1 *How to Use This Chapter*



"Do I Know This Already?" Quiz

The purpose of the "Do I Know This Already?" quiz is to help you decide what parts of this chapter to use. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

This 12-question quiz helps you determine how to spend your limited study time. The quiz is sectioned into three smaller four-question "quizlets," which correspond to the three major headings in the chapter. Figure 6-1 outlines suggestions on how to spend your time in this chapter. Use Table 6-1 to record your score.

Table 6-1 *Scoresheet for Quiz and Quizlets*

Quizlet Number	Foundation Topics Section Covering These Questions	Questions	Score
1	TCP/IP Protocols	1 to 4	
2	IP Addressing and Subnetting	5 to 8	
3	IP Configuration	9 to 12	
All questions	N/A	1 to 12	

1 What do *TCP*, *UDP*, *IP*, and *ICMP* stand for? Which protocol is considered to be Layer 3–equivalent when comparing TCP/IP to the OSI protocols?

2 Describe how to view the IP ARP cache in a Cisco router. Also describe the three key elements of each entry.

3 Does FTP or TFTP perform error recovery? If so, describe the basics of how error recovery is performed.

4 How many TCP segments are exchanged to establish a TCP connection? How many are required to terminate a TCP connection?

5 Given the IP address 134.141.7.11 and the mask 255.255.255.0, what is the subnet number?

6 Given the IP address 134.141.7.11 and the mask 255.255.255.0, what is the subnet broadcast address?

7 Given the IP address 200.1.1.130 and the mask 255.255.255.224, what are the assignable IP addresses in this subnet?

_____

_____

_____

8 Given the IP address 220.8.7.100 and the mask 255.255.255.240, what are all the subnet numbers if the same (static) mask is used for all subnets in this network?

_____

_____

_____

9 Create a minimal configuration enabling IP on each interface on a 2501 router (two serial, one Ethernet). The NIC assigned you network 8.0.0.0. Your boss says that you need, at most, 200 hosts per subnet. You decide against using VLSM. Your boss says to plan your subnets so that you can have as many subnets as possible rather than allowing for larger subnets later. When choosing the actual IP address values and subnet numbers, you decide to start with the lowest numerical values. Assume that point-to-point serial links will be attached to this router and that RIP is the routing protocol.

_____

_____

_____

10 Describe the question and possible responses in setup mode when a router wants to know the mask used on an interface. How can the router derive the correct mask from the information supplied by the user?

_____

_____

_____

11 Define the purpose of the **trace** command. What type of messages does it send, and what type of messages does it receive?

_____

_____

_____

12 What causes the output from an IOS **ping** command to display "UUUUU"?

_____

_____

_____

The answers to the quiz are found in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **8 or less overall score**—Read the entire chapter. This includes the "Foundation Topics" and "Foundation Summary" sections and the Q&A section at the end of the chapter.
- **2 or less on any quizlet**—Review the subsection(s) of the "Foundation Topics" part of this chapter, based on Table 6-1. Then move into the "Foundation Summary" section and the Q&A section at the end of the chapter.
- **9 to 10 overall score**—Begin with the "Foundation Summary" section, and then go to the Q&A section and the scenarios at the end of the chapter.
- **11 or more overall score**—If you want more review on these topics, skip to the "Foundation Summary" section and then go to the Q&A section at the end of the chapter. Otherwise, move to the next chapter.

Foundation Topics

TCP/IP Protocols

21 Identify the functions of the TCP/IP network-layer protocols.

22 Identify the functions performed by ICMP.

26 Define flow control and describe the three basic methods used in networking.

CCNAs work with multiple protocols on a daily basis; none of these is more important than TCP/IP. This section examines the TCP, UDP, ICMP, and ARP protocols in detail. TCP and UDP are the two transport layer (Layer 4) protocols most often used by applications in a TCP/IP network. ICMP and ARP are actually parts of the network layer (Layer 3) of TCP/IP and are used in conjunction with IP. As you'll see on the exam, IP addressing is something that all CCNAs must master to confidently pass the exam. Because of the importance of IP, IP addressing will be covered in great detail in the next section of this chapter.

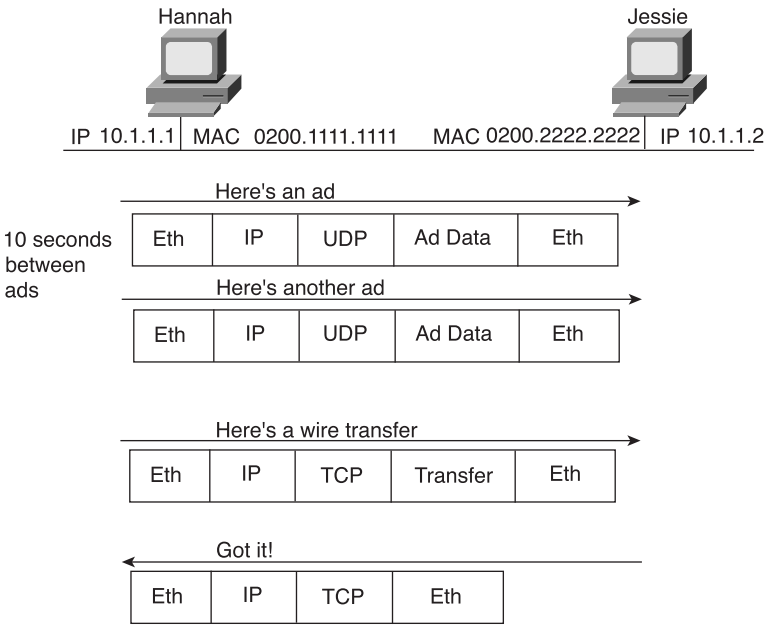
Overview of a Sample TCP/IP Network

TCP/IP encompasses a lot of smaller protocols—in fact, the name itself is a combination of two of the most popular of these many protocols, the Transmission Control Protocol and the Internet Protocol. The best way to get a sense of how some of these varied TCP/IP protocols work together is to examine a simple TCP/IP network with some simple applications. After that, we will look at each protocol more closely.

The sample network consists of two PCs, labeled Hannah and Jessie. Hannah uses an application that she wrote to send advertisements that display on Jessie's screen. The application sends a new ad to Jessie every 10 seconds. Hannah also uses a wire-transfer application to send Jessie some money. Finally, Hannah uses a web browser to access the web server that runs on Jessie's PC. The ad application and wire-transfer application are imaginary, just for this example. The web application works just like it would in real life.

We begin with the ad and wire-transfer applications behaving normally. Figure 6-2 shows the network, with flows for both applications.

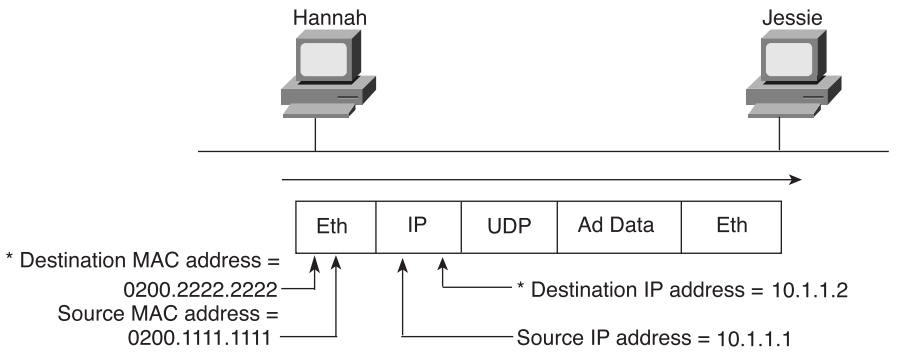
Figure 6-2 *Sample Network, Ad, and Wire Applications Working*



Hannah designed the ad application to not use acknowledgments—because another ad will be sent in 10 seconds anyway, there is little value to knowing whether the last ad got to the other user. For the wire transfer, Hannah included the capability to acknowledge the data. The ad application uses the User Datagram Protocol (UDP), and the wire application uses TCP because UDP does not use acknowledgments and TCP does.

However, these two applications cannot work until some overhead occurs. For example, the Ethernet frames going from Hannah to Jessie have information in them that Hannah did not know in advance—namely the destination IP and Ethernet addresses, as seen in Figure 6-3.

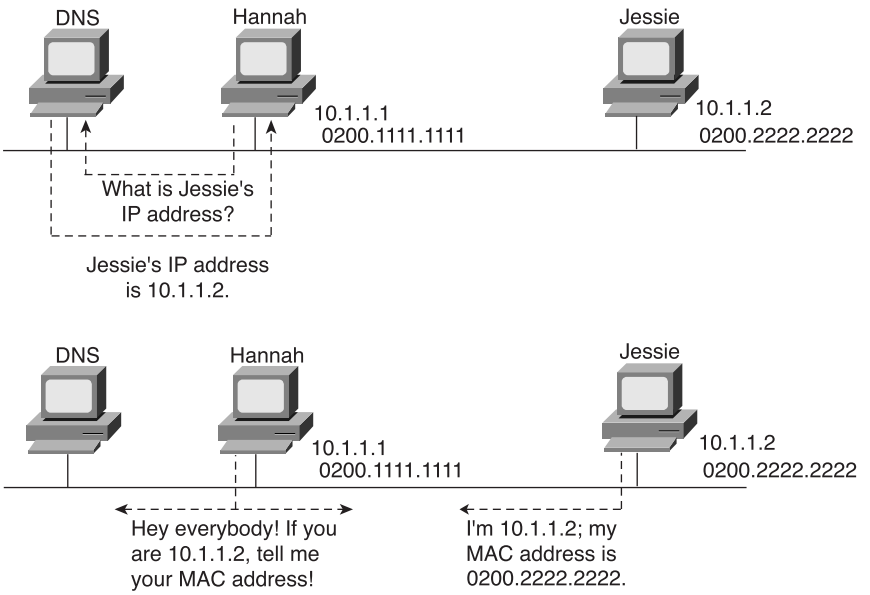
Figure 6-3 *Sample Network, with Addresses Shown in Headers*



Hannah knows her own name, IP address, and MAC address because those things are configured in advance. Before the applications can work, somehow Hannah needs to know Jessie's host name. (They are first cousins in real life, so knowing the names shouldn't be too hard!) *What Hannah does not know are Jessie's IP and MAC addresses.*

To find the two missing facts, Hannah uses the Domain Name System (DNS) and the Address Resolution Protocol (ARP). Hannah knows the IP address of the DNS because the address was preconfigured on Hannah's machine. Hannah now sends a *DNS request* to the DNS, asking for Jessie's IP address. The DNS replies with the address, 10.1.1.2. Hannah still needs to know the Ethernet MAC address used by 10.1.1.2, so Hannah issues something called an *ARP broadcast*. An ARP broadcast is sent to a broadcast Ethernet address, so everyone on the LAN receives it. Because Jessie is on the LAN, Jessie receives the ARP broadcast. Because Jessie's IP address is 10.1.1.2 and the ARP broadcast is looking for the MAC address associated with 10.1.1.2, Jessie replies with her own MAC address. Figure 6-4 outlines the process.

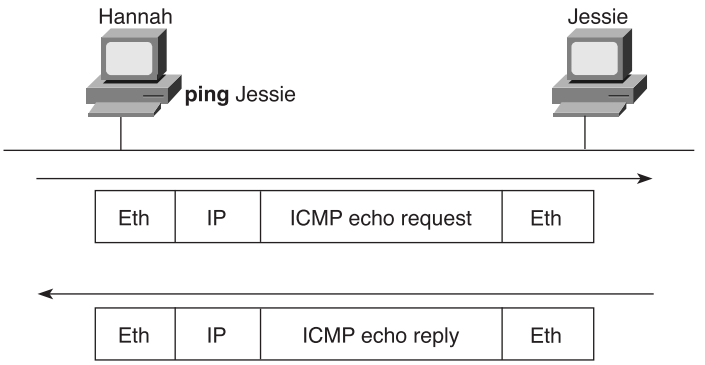
Figure 6-4 *Sample Network, DNS, and ARP Process*



Now Hannah knows the destination IP and Ethernet addresses that she should use when sending frames to Jessie, and the messages in Figure 6-2 can be sent successfully.

But what if all this overhead happens and the applications still do not work? Well, as any network engineer will tell you, it's probably the application! But if you ask the application support personnel, the problem is in the network! Hannah, being a great network troubleshooter (in spite of being my one-year-old daughter), decides to simply test basic network connectivity using the **ping** command. Ping (Packet INternet Groper) uses the *Internet Control Message Protocol (ICMP*) protocol, sending a message called an *ICMP echo request* to another IP address. The computer with that IP address should reply with an *ICMP echo reply*. If that works, you have successfully tested the IP network, and if it works, the problem is more likely related to the application. ICMP does not rely on any application, so it really just tests basic IP connectivity—Layers 1, 2, and 3 of the OSI model. Figure 6-5 outlines the basic process.

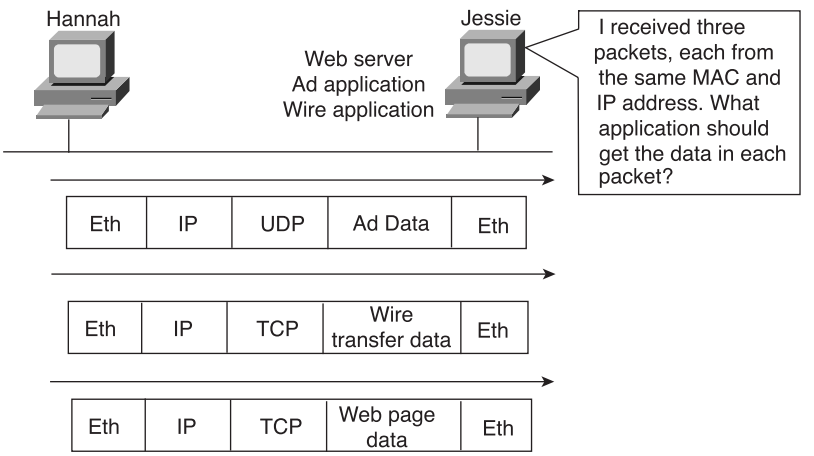
Figure 6-5 *Sample Network,* **ping** *Command*



ICMP contains many features, which will be discussed in detail in an upcoming section.

Understanding one more key concept in this overview will be useful. In this case, all three applications are being used—ad, wire transfer, and web browsing. Jessie receives packets for all three applications from Hannah, as shown in Figure 6-6.

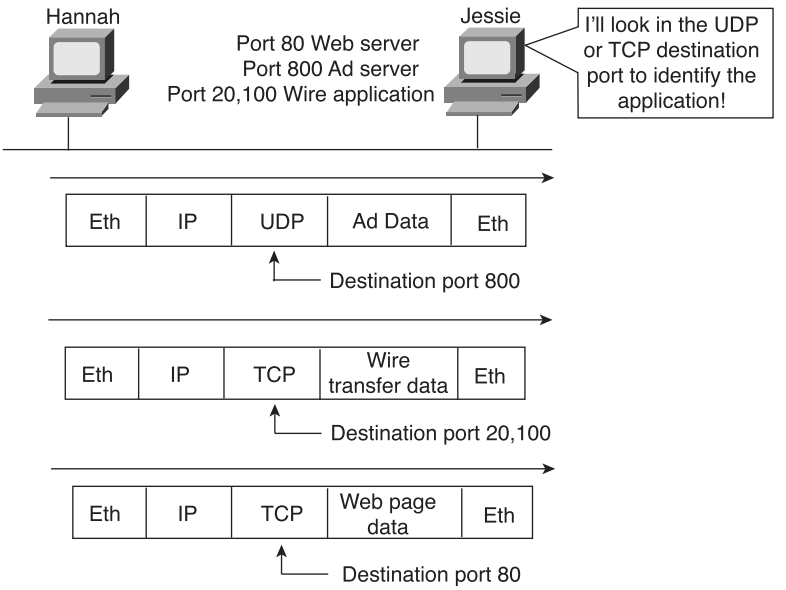
Figure 6-6 *Hannah Sending Packets to Jessie, with Three Applications*



Jessie needs to know which application to give the data to, but all three packets are from the same Ethernet and IP address. You might think that Jessie could look at whether the packet contains a UDP or a TCP header, but, as you see in the figure, two applications (wire transfer and web) both are using TCP. Fortunately, UDP and TCP designers purposefully included a field in the TCP and UDP headers to allow multiplexing, called the port number. *Multiplexing*

is the term used to generally describe the capability to determine which application gets the data for each packet. Each of Hannah's applications uses a different port number, so Jessie knows which application to give the data to.

Figure 6-7 *Hannah Sending Packets to Jessie, with Three Applications Using Port Numbers to Multiplex*



This overview provides a basic understanding of how a TCP/IP network behaves. More details will be covered in the next several sections of the book.

Transmission Control Protocol

Routers and hosts discard packets for a variety of reasons. A router might not have a route telling it where to forward a packet. A router or host might examine the FCS field in the frame trailer and discover that there were bit errors in transmission. Regardless, packets can be lost.

TCP provides a variety of useful features, including error recovery. In fact, TCP is best known for its error-recovery feature—but it does more. In fact, the CCNA exam covers the basics of five different features of TCP. Will every test taker see questions on all of these topics? Probably not. But TCP functions are within the scope of the test, so it pays to be ready.

The Transmission Control Protocol (TCP), defined in RFC 793, performs the following functions:

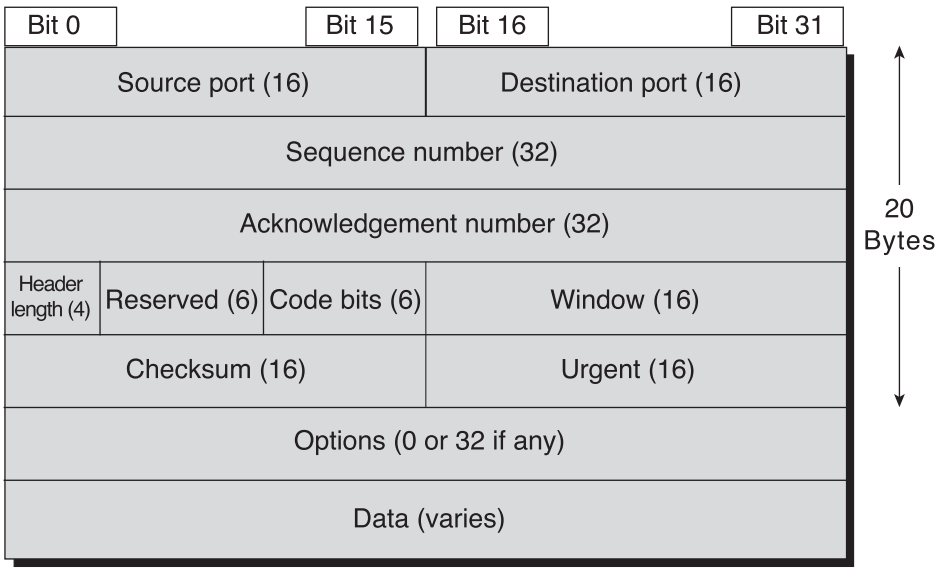
- Multiplexing
- Error recovery (reliability)

- Flow control using windowing
- Connection establishment and termination
- Data transfer

TCP accomplishes these goals through mechanisms at the endpoint computers. TCP relies on IP for end-to-end delivery of the data, including routing issues. In other words, TCP performs only part of the functions necessary to deliver the data between applications, and the role that it plays is directed toward providing services for the applications that sit at the endpoint computers.

Figure 6-8 shows the fields in the TCP header. Not all the fields will be described in this text, but several fields will be referred to in this section. The Internetworking Technologies Multimedia (ITM) CD, which is a suggested prerequisite for the exam, lists the fields along with brief explanations, as does the Cisco Press book on which ITM is based: *Internetworking Technologies Handbook,* Third Edition.

Figure 6-8 *TCP Header Fields*



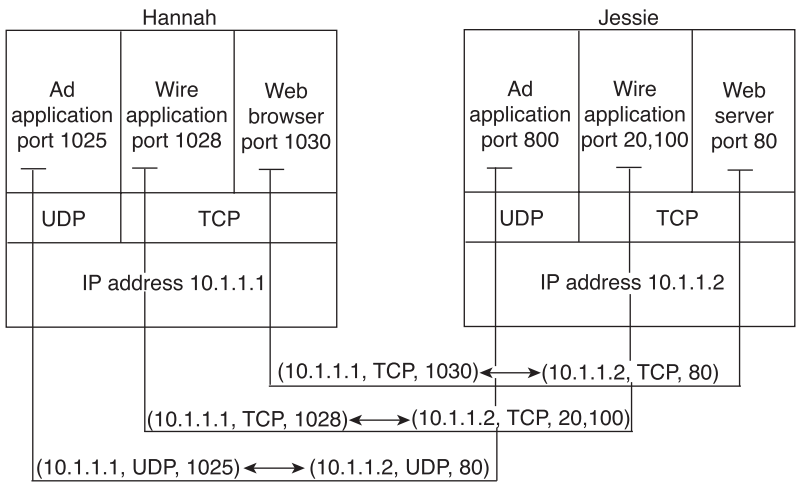
Multiplexing

In this context, multiplexing defines the process by which a host decides, among all its applications, which one should be given the incoming data. In the overview in the previous section, Jessie needed to decide whether to give the incoming data to the ad application, the wire-transfer application, or the web server application—that process is what we call multiplexing.

Multiplexing relies on the use of a concept called a *socket*. A socket consists of three things: an IP address, a transport protocol, and a port number. So for a web server application on Jessie, the socket would be (10.1.1.2, TCP, port 80) because, by default, web servers use the well-known port 80. When Hannah's web browser connected to the web server, Hannah used a socket as well—possibly one like this: (10.1.1.1, TCP, 1030). Why 1030? Well, Hannah just needs a port number that is unique on Hannah, so Hannah saw that port 1030 was available and used it. In fact, hosts typically allocate dynamic port numbers starting at 1024 because the ports below 1024 are reserved for well-known applications, such as web services.

In the overview section, Hannah and Jessie used three applications at the same time—hence, there were three socket connections open. Because a socket on a single computer should be unique, a connection between two sockets should identify a unique connection between two computers. The fact that each connection between two sockets is unique allows you each to use multiple applications at the same time, talking to applications running on the same or different computers; multiplexing, based on sockets, ensures that the data is delivered to the correct applications. Figure 6-9 shows the three socket connections between Hannah and Jessie.

Figure 6-9 *Connections Between Sockets*



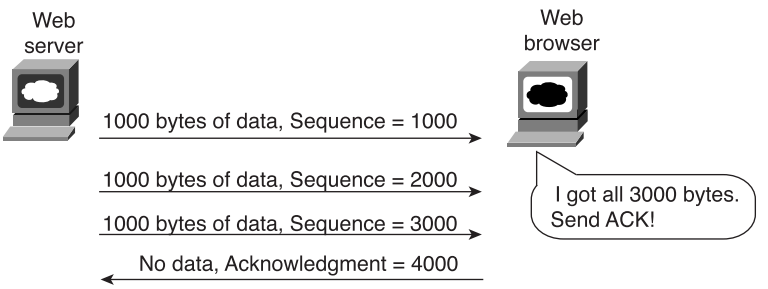
Port numbers are a vital part of the socket concept. Well-known port numbers are used by servers; other port numbers are used by clients. Applications that provide a service, such as FTP, Telnet, and web servers, open a socket using a well-known port and listen for connection requests. Because these connection requests from clients are required to include both the source and the destination port numbers, the port numbers used by the servers must be well known. Therefore, each server has a hard-coded, well-known port number, as defined in the well-known numbers RFC. On client machines, where the requests originate, any unused port number can be allocated. The result is that each client on the same host uses a different port number, but a

server uses the same port number for all connections. For example, 100 Telnet clients on the same host would each use a different port number, but the Telnet server with 100 clients connected to it would have only one socket and, therefore, only one port number. The combination of source and destination sockets allows all participating hosts to distinguish the source and destination of the data. (Look to www.rfc-editor.org to find RFCs such as the well-known numbers RFC 1700.)

Error Recovery (Reliability)

TCP provides for reliable data transfer, which is also called reliability or error recovery, depending on what document you read. To accomplish reliability, TCP numbers data bytes using the sequence and acknowledgment fields in the TCP header. TCP achieves reliability in both directions, using the Sequence Number field of one direction combined with the Acknowledgment Field in the opposite direction. If you remember error recovery from Chapter 3, "OSI Reference Model and Layered Communication," TCP performs it the same way. Figure 6-10 shows the basic operation.

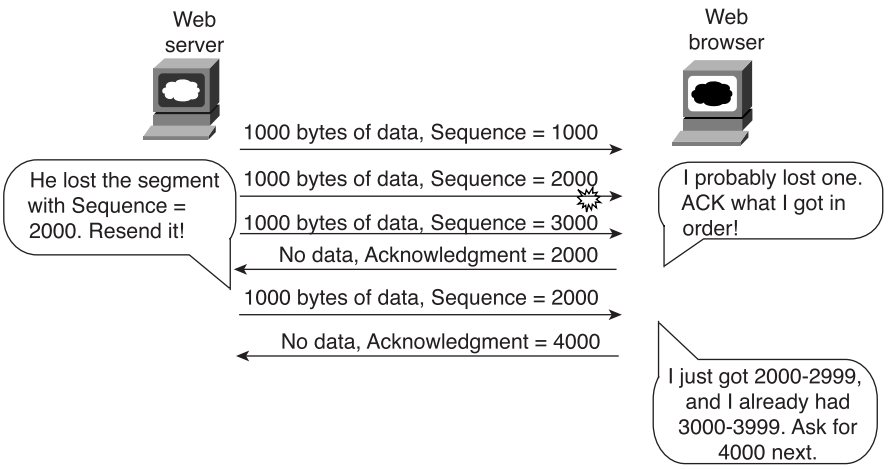
Figure 6-10 *TCP Acknowledgment Without Errors*



In Figure 6-10, the acknowledgment field in the TCP header sent by the web client (4000) implies the next byte to be received; this is called *forward acknowledgment*. The sequence number reflects the number of the first byte in the segment. In this case, each TCP segment is 1000 bytes in length; the sequence and acknowledgment fields count the number of bytes.

Figure 6-11 depicts the same scenario, but the second TCP segment was lost or was in error. The web client's reply has an ACK field equal to 2000, implying that the web client is expecting byte number 2000 next. The TCP function at the web server then could recover lost data by resending the second TCP segment. The TCP protocol allows for resending just that segment and then waiting, hoping that the web client will reply with an acknowledgment that equals 4000.

Figure 6-11 *TCP Acknowledgment with Errors*

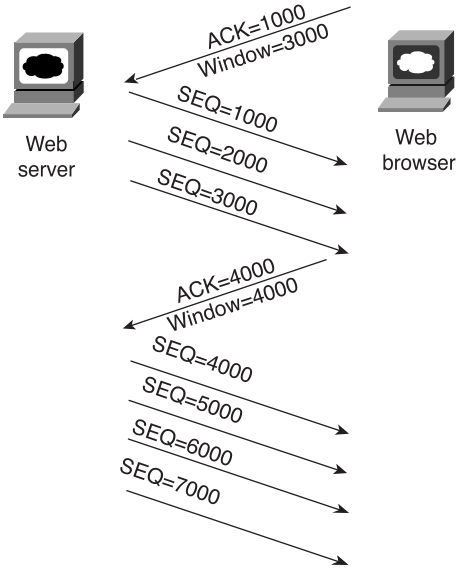


Flow Control Using Windowing

TCP implements flow control by taking advantage of the sequence and acknowledgment fields in the TCP header, along with another field called the *Window* field. This Window field implies the maximum number of unacknowledged bytes allowed outstanding at any instant in time. The window starts small and then grows until errors occur. The window then "slides" up and down based on network performance, so it is sometimes called a *sliding window*. When the window is full, the sender will not send, which controls the flow of data. Figure 6-12 shows windowing with a current window size of 3000. Each TCP segment has 1000 bytes of data.

Notice that the web server must wait after sending the third segment because the window is exhausted. When the acknowledgment has been received, another window can be sent. Because there have been no errors, the web client grants a larger window to the server, so now 4000 bytes can be sent before an acknowledgment is received by the server. In other words, the Window field is used by the receiver to tell the sender how much data it can send before it must stop and wait for the next acknowledgment. As with other TCP features, windowing is symmetrical—both sides send and receive, and, in each case, the receiver grants a window to the sender using the Window field.

Figure 6-12 *TCP Windowing*

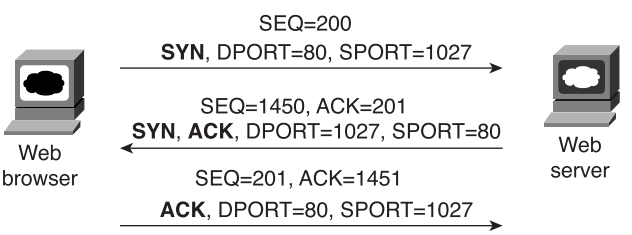


Windowing does not require that the sender stop sending in all cases. If an acknowledgment is received before the window is exhausted, a new window begins and the sender continues to send data until the current window is exhausted.

Connection Establishment and Termination

TCP connection establishment occurs before any of the other TCP features can begin their work. Connection establishment refers to the process of initializing sequence and acknowledgment fields and agreeing to the port numbers used. Figure 6-13 shows an example of connection establishment flow.

Figure 6-13 *TCP Connection Establishment*



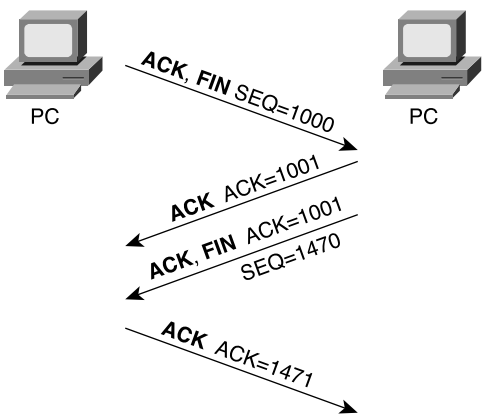
This three-way connection-establishment flow must complete before data transfer can begin. The connection exists between the two sockets, although there is no single socket field in the TCP header. Of the three parts of a socket, the IP addresses are implied based on the source and destination IP addresses in the IP header. TCP is implied because a TCP header is in use, as specified by the protocol field value in the IP header. Therefore, the only parts of the socket that need to be encoded in the TCP header are the port numbers.

TCP signals connection establishment using two bits inside the flag fields of the TCP header. Called the SYN and ACK flags, these bits have a particularly interesting meaning. SYN means "synchronize the sequence numbers," which is one necessary component in initialization for TCP. The ACK field means "the acknowledgment field is valid in this header." Until the sequence numbers are initialized, the acknowledgment field cannot be very useful. Also notice that in the initial TCP segment in Figure 6-13, no acknowledgment number is shown; this is because that number is not valid yet. Because the ACK field must be present in all the ensuing segments, the ACK bit will continue to be set until the connection is terminated.

TCP initializes the Sequence Number and Acknowledgment Number fields to any number that fits into the four-byte fields; the actual values shown in Figure 6-13 are simply example values. The initialization flows are each considered to have a single byte of data, as reflected in the Acknowledgment Number fields in the example.

Figure 6-14 shows TCP connection termination.

Figure 6-14 *TCP Connection Termination*



This four-way termination sequence is straightforward and uses an additional flag, called the FIN bit. (FIN is short for "finished," as you might guess.) *One interesting note:* Before the device receiving the first FIN segment sends the third TCP segment in the sequence, TCP notifies the application that the connection is coming down. TCP waits on an acknowledgment

from the application before sending the third segment in the figure. That's why the second segment is required: to acknowledge the first segment so that the side taking down the connection doesn't start resending the first TCP segment.

Ordered Data Transfer

TCP not only causes retransmissions when segments are lost, but it also reorders segments that arrive out of sequence. The process is not hard to imagine: If segments arrive with sequence numbers 1000, 3000, and 2000, each with 1000 bytes of data, the receiver can reorder them, and no retransmissions are required.

TCP Function Summary

Table 6-2 summarizes TCP functions.

Table 6-2 *TCP Function Summary*

Function	Description
Multiplexing	Function that allows receiving hosts to decide the correct application for which the data is destined, based on the port number
Error recovery (reliability)	Process of numbering and acknowledging data with sequence and acknowledgment header fields
Flow control using windowing	Process that uses window sizes to protect buffer space and routing devices
Connection establishment and termination	Process used to initialize port numbers and sequence and acknowledgement fields
Ordered data transfer	Continuous stream of bytes from upper-layer process that is "segmented" for transmission

User Datagram Protocol

The CCNA exam requires that you be able to compare and contrast the User Datagram Protocol (UDP) with TCP. UDP provides a service for applications to exchange messages. Unlike TCP, UDP is connectionless and provides no reliability, no windowing, and no function to ensure that the data is received in the order in which it was sent. However, UDP provides some functions of TCP, such as data transfer and multiplexing, and it does so with fewer bytes of overhead in the UDP header.

UDP multiplexes using port numbers in an identical fashion to TCP. The only difference in UDP (compared to TCP) sockets is that, instead of designating TCP as the transport protocol, the transport protocol is UDP. An application could open identical port numbers on the same host but use TCP in one case and UDP in the other—that is not typical, but it certainly is allowed. If a particular service supports both TCP and UDP transport, it uses the same value for the TCP and UDP port number, as shown in the assigned numbers RFC (currently RFC 1700—see www.isi.edu/in-notes/rfc1700.txt).

UDP data transfer differs from TCP data transfer in that no reordering or recovery is accomplished. Applications using UDP are tolerant of the lost data, or they have some application mechanism to recover lost data. For example, DNS requests use UDP because the user will retry an operation if the DNS resolution fails. The Network File System (NFS) performs recovery with application layer code, so UDP features are acceptable to NFS.

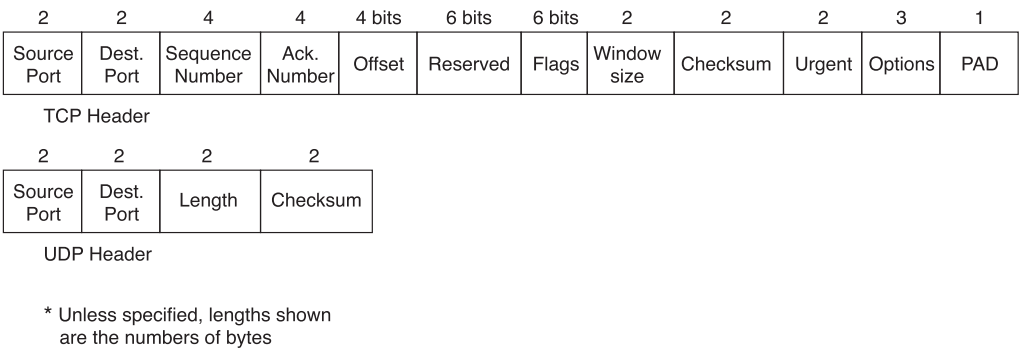
Table 6-3 contrasts typical transport layer functions as performed (or not performed) by UDP or TCP.

Table 6-3 *TCP and UDP Functional Comparison*

Function	Description (TCP)	Description (UDP)
Data transfer	This involves a continuous stream of ordered data.	This involves message (datagram) delivery.
Multiplexing	Receiving hosts decide the correct application for which the data is destined, based on the port number.	Receiving hosts decide the correct application for which the data is destined, based on the port number.
Reliable transfer	Acknowledgment of data uses the sequence and acknowledgment fields in the TCP header.	This is not a feature of UDP.
Flow control	This process is used to protect buffer space and routing devices.	This is not a feature of UDP.
Connections	This process is used to initialize port numbers and other TCP header fields.	UDP is connectionless.

Figure 6-15 shows TCP and UDP header formats. Note the existence of both Source Port and Destination Port fields in the TCP and UDP headers but the absence of Sequence Number and Acknowledgment Number fields in the UDP header. UDP does not need these fields because it makes no attempt to number the data for acknowledgments or resequencing.

Figure 6-15 *TCP and UDP Headers*

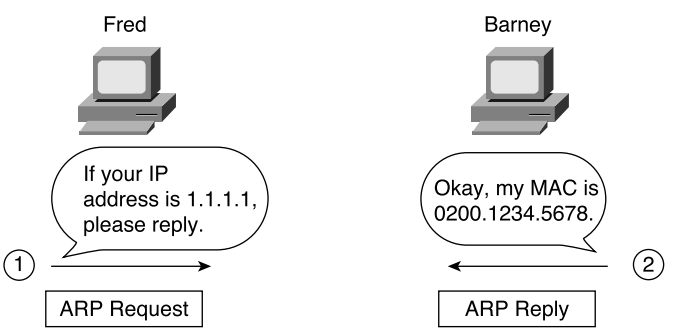


UDP gains some advantages over TCP by not using the sequence and acknowledgment fields. The most obvious advantage of UDP over TCP is that there are fewer bytes of overhead. Not as obvious is the fact that UDP does not require waiting on acknowledgments or holding the data in memory until it is acknowledged. This means that UDP applications are not artificially slowed by the acknowledgment process, and memory is freed more quickly.

Address Resolution Protocol

The Address Resolution Protocol (ARP) answers this question: Given an IP address of another device on the same LAN, what is its MAC address? ARP is needed because, to send an IP packet across some LANs, the data-link header and trailer (which encapsulate the packet) must first be created. The source MAC address in this new header is known because it is the MAC address of the sender. However, the destination MAC is not known in advance; ARP is the method that IP uses to discover the destination MAC address. Figure 6-16 shows an example of the ARP process.

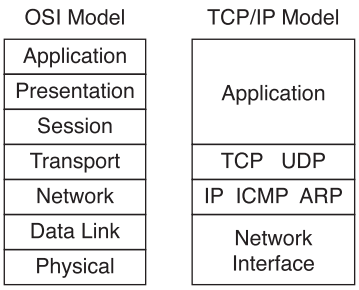
Figure 6-16 *The ARP Process*



The ARP reply includes Barney's MAC address, in this example. An ARP cache holds the ARP entries (IP address and MAC address in each entry) for each interface. After a period of disuse for an entry, the entry in the table is removed. After the ARP entry times out, another ARP exchange is required.

From an architectural perspective, ARP is a Layer 3 function and is defined in RFC 826. In fact, ARP does not use IP—ARP is carried inside a frame, just like IP. Note the location of ARP in the architectural model in Figure 6-17.

Figure 6-17 *TCP/IP Architectural Model*



Internet Control Message Protocol

You should know both the general concepts and several specifics about the Internet Control Message Protocol (ICMP) for the CCNA exam. *Control Message* is the most descriptive part of the name— ICMP helps control and manage the work of IP and, therefore, is considered to be part of TCP/IP's network layer. RFC 792 defines ICMP and includes the following excerpt, which describes the protocol well:

> Occasionally a gateway or destination host will communicate with a source host, for example, to report an error in datagram processing. For such purposes, this protocol, the Internet Control Message Protocol (ICMP), is used. ICMP uses the basic support of IP as if it were a higher level protocol; however, ICMP is actually an integral part of IP and must be implemented by every IP module.

ICMP uses messages to accomplish its tasks. Many of these messages are used in even the smallest IP network. Table 6-4 lists several ICMP messages, with the ones most likely to be on the exam noted with an asterisk. Not surprisingly, these are the same messages used most often. The Destination Unreachable, Time Exceeded, and Redirect messages will be described in more detail following Table 6-4.

Table 6-4 *ICMP Message Types*

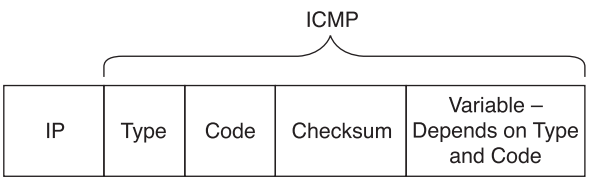
Message	Purpose
*Destination Unreachable	This tells the source host that there is a problem delivering a packet.
*Time Exceeded	The time that it takes a packet to be delivered has become too long; the packet has been discarded.
Source Quench	The source is sending data faster than it can be forwarded; this message requests that the sender slow down.
*Redirect	The router sending this message has received some packet for which another router would have had a better route; the message tells the sender to use the better route.
*Echo	This is used by the **ping** command to verify connectivity.
Parameter Problem	This is used to identify a parameter that is incorrect.
Timestamp	This is used to measure round-trip time to particular hosts.
Address Mask Request/Reply	This is used to inquire about and learn the correct subnet mask to be used.
Router Advertisement and Selection	This is used to allow hosts to dynamically learn the IP addresses of the routers attached to the subnet.

*Most likely to be on the CCNA exam

Start Extra Credit

Each ICMP message contains a Type field and a Code field, as shown in Figure 6-18. The Type field implies the message types from Table 6-4. The Code field implies a subtype; several subtypes will be shown in the following examples. I do not expect any ICMP header format questions on the exam; it's just good for perspective with the upcoming explanations.

Figure 6-18 *ICMP Header Formats*



End Extra Credit

ICMP Echo Request and Echo Reply

The ICMP Echo Request and Echo Reply messages are sent and received by the **ping** command. In fact, when people say that they sent a ping packet, they really mean that they sent an ICMP Echo Request. These two messages are very self-explanatory. The Echo Request simply means that the host to which it is addressed should reply to the packet. The Echo Reply is the ICMP message type that should be used in the reply. The Echo Request includes some data that can be specified by the **ping** command; whatever data is sent in the Echo Request is sent back in the Echo Reply.

The **ping** command itself supplies many creative ways to use Echo Requests and Replies. For instance, the **ping** command enables you to specify the length as well as the source and destination addresses, and it also enables you to set other fields in the IP header. Example 6-6, later in this chapter, shows a good example of the capabilities of the **ping** command.

Destination Unreachable ICMP Message

The ICMP unreachable message is sent when a message cannot be delivered completely to the application at the destination host. However, packet delivery could fail for many reasons, so there are five separate unreachable functions (codes) using this single ICMP unreachable message. All five code types pertain directly to some IP, TCP, or UDP feature and are better described by using Figure 6-19 as an example network.

Assume that Fred is trying to connect to the web server, called Web. (Web uses TCP as the transport layer protocol.) Three of the ICMP unreachable codes would possibly be used by Routers A and B. The other two codes would be used by the web server. These ICMP codes would be sent to Fred as a result of the packet originally sent by Fred.

Figure 6-19 *Sample Network for Discussing ICMP Unreachable Codes*

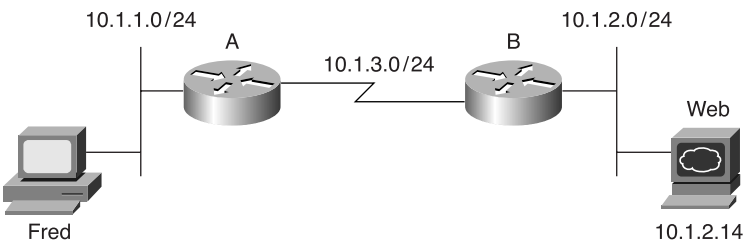


Table 6-5 summarizes the more common ICMP unreachable codes.

Table 6-5 *ICMP Unreachable Codes*

Unreachable Code	When Used	Typically Sent By
Network unreachable	There is no match in a routing table for the destination of the packet.	Router
Host unreachable	The packet can be routed to a router connected to the destination subnet, but the host is not responding.	Router
Can't fragment	The packet has the Don't Fragment bit set, and a router must fragment to forward the packet.	Router
Protocol unreachable	The packet is delivered to the destination host, but the transport layer protocol is not available on that host.	Endpoint host
Port unreachable	The packet is delivered to the destination host, but the destination port has not been opened by an application.	Endpoint host

The list that follows explains each code in Table 6-5 in greater detail:

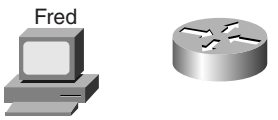
- **Network unreachable**—A code meaning "network unreachable" will be used by Router A if Router A does not have a route telling it where to forward the packet. In this case, Router A needs a route to subnet 10.1.2.0. The ICMP unreachable message, with the code "network unreachable," is sent by Router A to Fred, in response to Fred's packet destined for 10.1.2.14.
- **Host unreachable**—This code implies that the single destination host is unavailable. If Router A has a route to 10.1.2.0, the packet will be delivered to Router B. However, if the web server is down, Router B will not get an ARP reply from web. The ICMP unreachable message, with code "host unreachable," will be sent by Router B to Fred, in response to Fred's packet destined for 10.1.2.14.
- **Can't fragment**—This code is the last of the three ICMP unreachable codes that might be sent by a router. Fragmentation defines a process in which a router needs to forward a packet, but the outgoing interface allows only packets that are smaller than the forwarded packet. The router can break the packet into pieces. However, if Router A or Router B needs to fragment the packet but the Do Not Fragment bit is set in the IP header, the router will discard the packet. The ICMP unreachable message, with code "can't fragment," will be sent by Router A or B to Fred, in response to Fred's packet destined for 10.1.2.14.
- **Protocol unreachable**—If the packet successfully arrives at the web server, two other unreachable codes are possible. One implies that the protocol above IP, typically TCP or UDP, is not running on that host. This is highly unlikely because most operating systems that use TCP/IP use a single software package that provides IP, TCP and UDP functions. But if the host receives the IP packet and TCP or UDP is not available, the ICMP unreachable message, with code "protocol unreachable," will be sent by the web server host to Fred, in response to Fred's packet destined for 10.1.2.14.

- **Port unreachable**—The final code field value is more likely today. If the server is up but the web server software is not running, the packet can get to the server but cannot be delivered to the web server software. The ICMP unreachable message, with code "port unreachable," will be sent by the web server host to Fred, in response to Fred's packet destined for 10.1.2.14.

Time Exceeded ICMP Message

The ICMP Time Exceeded message notifies a host when a packet that it sent has been discarded because it was "out of time." Packets are not actually timed, but to prevent packets from being forwarded forever when there is a routing loop, each IP header uses a Time to Live (TTL) field. Routers decrement TTL by one every time they forward a packet; if a router decrements TTL to zero, it throws away the packet. This prevents packets from rotating forever. Figure 6-20 shows the basic process.

Figure 6-20 *TTL Decremented to Zero*



CA260620.eps

As you see in the figure, the router that discards the packet also sends an ICMP Time Exceeded message, with a code field of "time exceeded." That way, the sender knows that the packet was not delivered. Getting a Time Exceeded message can also help you when troubleshooting a network. Hopefully, you do not get too many of these; otherwise, you have routing problems.

The IOS **trace** command uses the Time Exceeded message and the IP TTL field to its advantage. By purposefully sending IP packets (with a UDP transport layer) with the TTL set to one, an ICMP Time Exceeded message is returned by the first router in the route. That's

because that router decrements TTL to zero, causing it to discard the packet, and also sends the Time Exceeded message. The **trace** command learns the IP address of the first router by receiving the Time Exceeded message from that router. (The **trace** command actually sends three successive packets with TTL = 1.) Another set of three IP packets, this time with TTL = 2, is sent by the **trace** command. These messages make it through the first router but are discarded by the second router because the TTL is decremented to zero. The original packets sent by the host **trace** command use a destination port number that is very unlikely to be used so that the destination host will return the Port Unreachable message. The ICMP Port Unreachable message signifies that the packets reached the true destination host without having time exceeded, so the **trace** command knows that the packets are getting to the true endpoint. Figure 6-21 outlines the process, in which Router A is using the **trace** command trying to find the route to Barney. Example 6-1 shows this **trace** command on Router A, with debug messages from Router B, showing the resulting Time Exceeded messages.

Figure 6-21 *Cisco IOS Software* **trace** *Command—Messages Generated*

Example 6-1 *ICMP* **debug** *on Router B When Running* **trace** *Command on Router A*

```
RouterA#trace 10.1.2.14

Type escape sequence to abort.
Tracing the route to 10.1.2.14

  1 10.1.3.253 8 msec 4 msec 4 msec
  2 10.1.2.14 12 msec 8 msec 4 msec
```

Example 6-1 *ICMP* **debug** *on Router B When Running* **trace** *Command on Router A (Continued)*

```
RouterA#
```

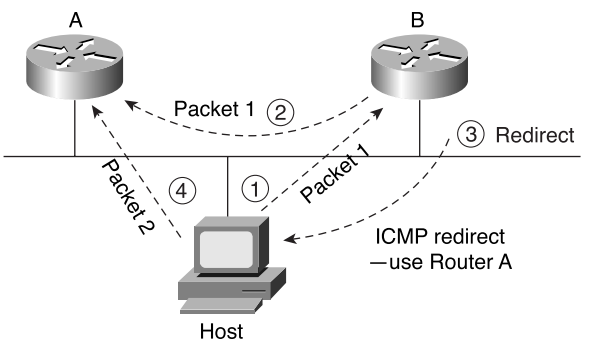
```
RouterB#
ICMP: time exceeded (time to live) sent to 10.1.3.251 (dest was 10.1.2.14)
ICMP: time exceeded (time to live) sent to 10.1.3.251 (dest was 10.1.2.14)
ICMP: time exceeded (time to live) sent to 10.1.3.251 (dest was 10.1.2.14)
```

Redirect ICMP Message

ICMP redirect messages provide a very important element in routed IP networks. Many hosts are preconfigured with a default router IP address. When sending packets destined to subnets other than the one to which they are directly connected, these hosts send the packets to their default router. If there is a better local router to which the host should send the packets, an ICMP redirect can be used to tell the host to send the packets to this different router.

For example, in Figure 6-22, the PC uses Router B as its default router. However, Router A's route to subnet 10.1.4.0 is a better route. (Assume use of mask 255.255.255.0 in each subnet in Figure 6-22.) The PC sends a packet to Router B (Step 1 in Figure 6-22). Router B then forwards the packet based on its own routing table (Step 2); that route points through A, which has a better route. Finally, Router B sends the ICMP redirect message to the PC (Step 3), telling it to forward future packets destined for 10.1.4.0 to Router A instead. Ironically, the host can ignore the redirect and keep sending the packets to Router B.

Figure 6-22 *Example of an ICMP Redirect*



In summary, ICMP defines several message types and several subtypes, called codes. Popular use of terminology treats each different code as a different message; the exam is likely to treat these codes as different messages as well, although it is unlikely that the level of granularity will be important in getting the right answer. Pay particular attention to the messages denoted with asterisks in Table 6-5. Finally, RFC 792 is a short and straightforward RFC to read if you want more information.

Start Extra Credit

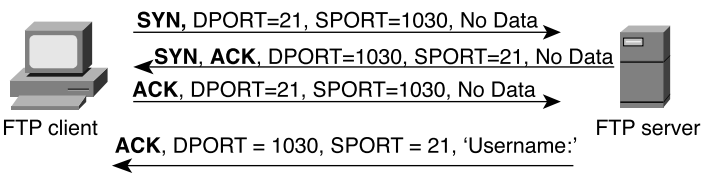
FTP and TFTP

The File Transfer Protocol (FTP) and the Trivial File Transfer Protocol (TFTP) are two popularly used file-transfer protocols in a typical IP network. Most users use FTP; whereas, router and switch administrators use TFTP. Which is "better" depends partially on what is being done. A more important question might be, "Which is supported on the devices that need to transfer the file?" Given a choice today, most users will choose FTP because it has more robust features. However, IOS did not support FTP for moving files in and out of the router originally, so many people continue to use TFTP out of habit.

FTP

FTP is a TCP-based application that has many options and features, including the capabilities to change directories, list files using wildcard characters, transfer multiple files with a single command, and use a variety of character sets or file formats. More important in this context is the basic operation of FTP. Figures 6-23 and 6-24 show a typical FTP connection—or, better stated, connections.

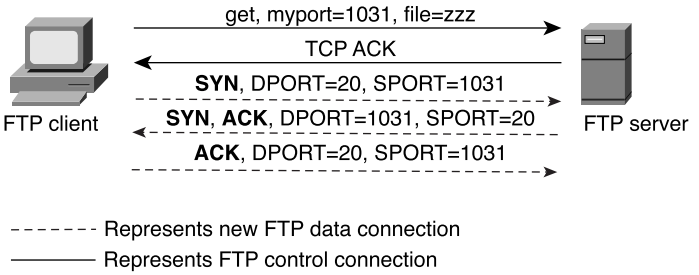
Figure 6-23 *FTP Control Connection*



The connection shown in Figure 6-23 is called an *FTP control connection*. When a user (FTP client) asks to connect to an FTP server, a TCP connection is established to the FTP server's well-known port (21). The connection is established like any other TCP connection. The user typically is required to enter a username and password, which the server uses to authenticate the files available to that user for read and write permissions. This security is based on the file security on the server's platform. All the commands used to control the transfer of a file are sent across this connection—hence the name *FTP control connection*.

At this point, the user has a variety of commands available to enable settings for transfer, change directories, list files, and so forth. However, whenever a **get** or a **put** command is entered (or **mget** or **mput**—*m* is for *multiple*) or the equivalent button is clicked, a file is transferred. The data is transferred over a separate *FTP data connection*. Figure 6-24 outlines the FTP data-connection process.

Figure 6-24 *FTP Data Connection*



As shown in Figure 6-24, another TCP connection is established for the actual data transfer, this time to well-known port 20. Using this convention, a file can be transferred without getting in the way of the control connection. If many files are to be transferred rather than making a single control/data connection for each file, the control connection is made once. The environment is defined using the control connection, and these settings affect the functioning of the data connection. For example, the default directory to use in future transfers can be defined using commands on the control connection as well as the type of data (binary or ASCII). The control connection stays up until the user breaks it or the connection times out. While the control connection is up, a separate data connection is established for each file transfer.

An additional step helps prevent hackers from breaking in and transferring files, as shown in Figure 6-24. Rather than just creating a new connection, the client tells the server with an application layer message over the control connection which port number will be used for the new data connection. The server will not transfer the file (zzz, in this case) over any other data connection except the one to the correct socket—the one with the client's IP address, TCP, and the port number declared to the server (1031, in this case).

TFTP

The Trivial File Transfer Protocol (TFTP) performs basic file transfer with a small set of features. One of the reasons that such an application is needed, even when the more robust FTP is available, is that TFTP takes little memory to load and little time to program. With the advent of extremely low-cost memory and processing, such advantages seem trivial. Practically speaking, if you intend to transfer files frequently from your PC, you probably will use FTP. However, Cisco has supported TFTP for a much longer time, many people still use it, and basic exams might still use TFTP rather than FTP in examples.

TFTP uses UDP, so there is no connection establishment and no error recovery by the transport layer. However, TFTP uses application layer recovery by embedding a small header between the UDP header and the data. This header includes codes—for example, read, write, and acknowledgment—along with a numbering scheme that numbers 512-byte blocks of data.

These block numbers are used to acknowledge receipt and resend the data. TFTP sends one block and waits on an acknowledgment before sending another block.

Table 6-6 summarizes some features of TFTP and FTP.

Table 6-6 *Comparison of FTP and TFTP*

FTP	TFTP
Uses TCP	Uses UDP
Uses robust control commands	Uses simple control commands
Sends data over a separate TCP connection from control commands	Uses no connections because of UDP
Requires more memory and programming effort	Requires less memory and programming effort

End Extra Credit

IP Addressing and Subnetting

20 Describe the different classes of IP addresses (and subnetting).

No one reading this book should be shocked to hear that IP addressing is one of the most important topics on the CCNA exam. You need a comfortable, confident understanding of IP addressing and subnetting for success on any Cisco certification. For CCNA, the exam questions will ask for an interpretation of an address, its network number, its subnet number, the other IP addresses in the same subnet, the broadcast address, and the other subnets that could be used if the same mask were in use. In other words, you had better know subnetting!

Network engineers need to understand subnetting very well. Engineers who work with multiple networks must decipher IP addresses quickly, without running off to use a subnet calculator tool. For example, someone with a problem might call and tell you his IP address. After finding out the mask that's used, you do a **show ip route** command on a router, but that typically lists subnets—so you need to be able to easily figure out the subnet of which the address is a member. And not all networks will be using nice, easy subnet masks.

You need the same understanding and tricks to get the right answers quickly and easily to pass the CCNA exam with confidence. After reading this section, you will be able to confidently understand subnetting and quickly derive all the information related to an IP address.

IP Addressing and Subnetting

Engineers use IP addressing terminology in many different ways—and sometimes people use the terms to mean slightly different things. Table 6-7 lists the IP terms used in the upcoming sections, giving an exact definition. Feel free to refer to this table as you read.

Table 6-7 *IP Addressing Terminology*

Term	Definition
IP address	A 32-bit number, usually written in dotted-decimal form, that uniquely identifies an interface of some computer.
Host address	Another term for IP address.
Network	A group of hosts, all of which have an identical beginning portion of their IP addresses.
Network number	A 32-bit number, usually written in dotted-decimal form, that represents a network. This number cannot be assigned as an IP address to an interface of some computer. The host portion of the network number has a value of all binary 0s.
Network address	Another term for the network number.
Broadcast address	A 32-bit number, usually written in dotted-decimal form, that is used to address all hosts in the network. This number cannot be assigned as an IP address to an interface of some computer. The host portion of the network number has a value of all binary 1s.
Subnet	A group of hosts, all of which have an identical beginning portion of their IP addresses. A subnet differs from a network in that a subnet is a further subdivision of a network, with a longer portion of the addresses being identical.
Subnet number	A 32-bit number, usually written in dotted-decimal form, that represents a subnet. This number cannot be assigned as an IP address to an interface of some computer. The host portion of the network number has a value of all binary 0s.
Subnet address	Another term for the subnet number.
Subnet broadcast address	A 32-bit number, usually written in dotted-decimal form, that is used to address all hosts in the subnet. This number cannot be assigned as an IP address to an interface of some computer. The host portion of the network number has a value of all binary 1s.

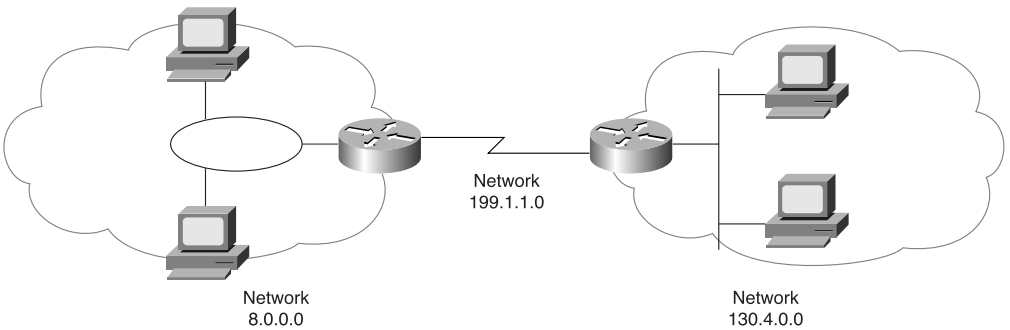
continues

Table 6-7 *IP Addressing Terminology (Continued)*

Term	Definition
Subnetting	The process of subdividing networks into smaller subnets. This is jargon—for example, "Are you subnetting your network?"
Network mask	A 32-bit number, usually written in dotted-decimal form. The mask is used by computers to calculate the network number of a given IP address by performing a Boolean AND of the address and mask. The mask also defines the number of host bits in an address.
Mask	A generic term for a mask, whether it is a default mask or a subnet mask.
Address mask	Another term for a mask.
Default Class A mask	The mask used for Class A networks when no subnetting is used. The value is 255.0.0.0.
Default Class B mask	The mask used for Class B networks when no subnetting is used. The value is 255.255.0.0.
Default Class C mask	The mask used for Class C networks when no subnetting is used. The value is 255.255.255.0.
Subnet mask	A nondefault mask that is used when subnetting.
Network part or network field	Term used to describe the first part of an IP address. The network part is 8, 16, or 24 bits for Class A, B, and C networks, respectively.
Host part or host field	Term used to describe the last part of an IP address. The host part is 24, 16, or 8 bits for Class A, B, and C networks, respectively, when subnetting is not used. When subnetting, the size of the host part depends on the subnet mask chosen for that network.
Subnet part or subnet field	Term used to describe the middle part of an IP address. The subnet part is variable in size, based on how subnetting is implemented.

To fully appreciate IP addressing, you first must understand the concepts behind the grouping of IP addresses. The first visions of what we call the Internet were for connecting research sites. A typical network diagram might have looked like Figure 6-25.

Figure 6-25 *Sample Network Using Class A, B, and C Network Numbers*



The conventions of IP addressing and IP address grouping make routing easy. For example, all IP addresses that begin with 8 are on the Token Ring on the left. Likewise, all IP addresses that begin with 130.4 are on the right. Along the same lines, 199.1.1 is the prefix on the serial link. By following this convention, the routers build a routing table with three entries, one for each prefix, or network number.

The convention fails if you do not follow these rules:

IP addresses are assigned to interfaces, not to entire computers. Therefore, routers will have multiple IP addresses, one per interface.

All IP addresses in the same group must not be separated by a router.

IP addresses separated by a router must be in different groups.

In Figure 6-25, all IP addresses assigned to interfaces on the Token Ring are in the same group. Likewise, both serial interfaces, one on each router, are in a second group. Finally, all the interfaces on the Ethernet are in the third IP address grouping.

IP addressing behaves similarly to ZIP codes. Everyone living in my ZIP code lives in my town. If some members of my ZIP code were in California, some of my mail might be sent out there (I live in Georgia, by the way). Just as it would be silly for addresses in the same ZIP code to be in different states, having IP addresses in the same network grouping be on different sites in the network topology is silly.

Classes of Networks

RFC 790 defines the IP protocol, including three different sizes of networks. By definition, all addresses in the same network have the same numeric value *network* portion of the addresses. The rest of the address is called the *host* portion of the address. Using the post office example, the network part is the ZIP code, and the host part is the street address. Just as a letter-sorting

machine three states away from you cares only about the ZIP code on a letter addressed to you, a router three hops away from you cares only about the subnet number that your address resides in. Individual IP addresses in the same network all have a different value in the host part of the addresses, but they have identical values in the network part, just as everyone in my town has a different street address but the same ZIP code.

Class A networks have a 1-byte-long network part. That leaves 24 bits for the rest of the address, or the host part. That means that 2^{24} addresses are numerically possible in a Class A network. Similarly, Class B networks have a 2-byte-long network part, leaving 16 bits for the host portion of the address. So 2^{16} possible addresses exist in a single Class B network. Finally, Class C networks have a 3-byte-long network part, leaving only 8 bits for the host part, which implies only 2^8 addresses in a Class C network. However, there are two reserved host addresses in each network, as shown in the last column of Table 6-8. The table summarizes the characteristics of Class A, B, and C networks.

Table 6-8 *Sizes of Network and Host Parts of IP Addresses with No Subnetting*

Any Network of This Class	Number of Network Bytes (Bits)	Number of Host Bytes (Bits)	Number of Addresses per Network*
A	1 (8)	3 (24)	2^{24}, minus two special cases
B	2 (16)	2 (16)	2^{16}, minus two special cases
C	3 (24)	1 (8)	2^8, minus two special cases

*There are two reserved host addresses per network.

For example, Figure 6-25 shows a small network with the IP network numbers filled in. Network 8.0.0.0 is a Class A network, network 130.4.0.0 is a Class B network, and network 199.1.1.0 is a Class C network.

Network numbers look like addresses because they are in dotted-decimal format. However, network numbers cannot be assigned to an interface as an IP address. Conceptually, network numbers represent the group of all IP addresses in the network, much like a ZIP code represents the group of all addresses in a community. Numerically, the network number is built with the network number in the network part, but with all binary 0s in the host part. Based on the three examples from Figure 6-25, Table 6-9 provides a closer look at the numerical version of the three network numbers: 8.0.0.0, 130.4.0.0, and 199.1.1.0.

Table 6-9 *Example Network Numbers, Decimal and Binary*

Network Number	Binary Representation, with Host Part Bold
8.0.0.0	0000 1000 **0000 0000 0000 0000 0000 0000**
130.4.0.0	1000 0010 0000 0100 **0000 0000 0000 0000**
199.1.1.0	1100 0111 0000 0001 0000 0001 **0000 0000**

The binary values provide a little more insight into the IP address structure. By convention, Class A networks have a one-octet network part and a three-octet host part. With 24 host bits, you can number 2^{24} hosts—except that two of those are reserved. Of course, who cares if you have to waste two addresses when you have more than 16 million addresses? One of the two reserved values is the network number itself—the value shown in Table 6-9. The other reserved value is the one with all binary 1s in the host part of the address—this number is the network broadcast address. And as you might guess, all the numbers between the network number and the broadcast address are the valid, useful IP addresses that can be used to address interfaces in the network.

Many different Class A, B, and C networks exist. If your firm connects to the Internet without using a form of Address Translating Gateway (such as the Cisco PIX), it must use registered, unique network numbers. To that end, the Network Information Center (NIC) assigns network numbers so that uniqueness is achieved. Table 6-10 summarizes the possible network numbers, the total number of each type, and the number of hosts in each Class A, B, and C network.

Table 6-10 *List of All Possible Valid Network Numbers**

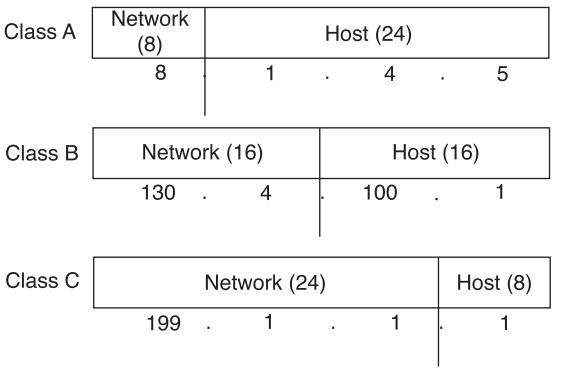
Class	First Octet Range	Valid Network Numbers	Total Number of This Class of Network	Number of Hosts per Network	Number of Bytes in Network Part of Address	Number of Bytes in Host Part of Address
A	1 to 126	1.0.0.0 to 126.0.0.0	2^7, minus two special cases	2^{24}, minus two special cases	1	3
B	128 to 191	128.1.0.0 to 191.254.0.0	2^{14}, minus two special cases	2^{16}, minus two special cases	2	2
C	192 to 223	192.0.1.0 to 223.255.254.0	2^{21}, minus two special cases	2^8, minus two special cases	3	1

*The Valid Network Numbers column shows actual network numbers. There are several reserved cases. For example, network 0.0.0.0 (originally defined for use as a broadcast address) and 127.0.0.0 (still available for use as the loopback address) are reserved. Networks 128.0.0.0, 191.255.0.0, 192.0.0.0, and 223.255.255.0 also are reserved.

Engineers should classify a network as Class A, B, or C with ease. If you have not already done so, you should memorize the first octet ranges in Table 6-11. It's on the exam—reason enough. However, if you plan on having a job for which you work with lots of different installations, as you would if you worked for a vendor, reseller, service provider, or consulting company, classifying a network as Class A, B, or C should become an instantaneous process. Also memorize the number of octets in the network part of Class A, B, and C addresses, as shown in Table 6-8.

For example, Figure 6-26 shows three example IP addresses, one for each of the three networks shown in Figure 6-25. One address is in a Class A network, one is in a Class B network, and one is in a Class C network.

Figure 6-26 *Example Class A, B, and C IP Addresses and Their Formats*



By definition, an address that begins with 8 is in a Class A network, so the network part of the address is the first byte, or first octet. (Refer to Table 6-10 for a reminder of the Class A, B, and C ranges in the first octet.) An address that begins with 130 is in a Class B network, and, by definition, Class B addresses have a two-byte network part, as shown. Finally, any address that begins with 199 is in a Class C network, which has a three-byte network part. Also by definition, a Class A address has a three-byte host part, Class B has a two-byte host part, and Class C has a three-byte host part.

Humans can simply remember the numbers in Table 6-10 and the concepts in Figure 6-26 and then quickly determine the network and host part of an IP address. Computers, however, use a mask to define the size of the network and host parts of an address. The logic behind the mask results in the same conventions of Class A, B, and C networks that you already know, but the computer can deal with it better as a binary math problem. The mask is a 32-bit binary number, usually written in dotted-decimal format, that defines the structure of an IP address. In short, the mask defines the size of the host parts of an IP address, representing the host part of the IP address with binary 0s. Class A mask has its last 24 bits as binary 0, which means that the last three octets of the mask, in decimal, are 0s. Table 6-11 summarizes the default masks and reflects the sizes of the two parts of an IP address.

Table 6-11 *Class A, B, and C Networks—Network and Host Parts and Default Masks*

Class of Address	Size of Network Part of Address, in Bits	Size of Host Part of Address, in Bits	Default Mask for Each Class of Network
A	8	24	255.0.0.0
B	16	16	255.255.0.0
C	24	8	255.255.255.0

IP Grouping Concepts and Subnetting

The creators of the Internet realized the impracticality of the original network-numbering conventions early on. Computing history shows many examples of people being unable to conceive the idea that computing technology would grow as fast as it has. Needless to say, the Internet would have run out of Class A, B, and C networks long ago if additional addressing features had not been created. Subnetting provided the first significant addressing feature that conserved the global IP address space.

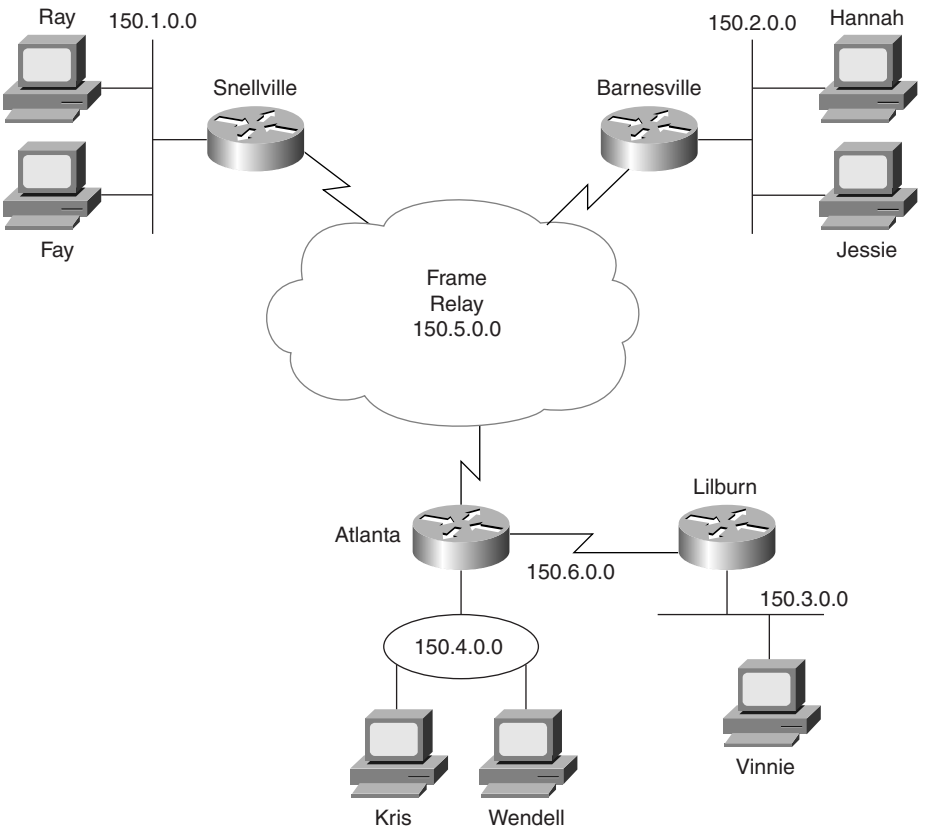
IP subnetting creates vastly larger numbers of smaller groups of IP addresses, compared with simply using Class A, B, and C conventions. The Class A, B, and C rules still exist—but now, a single Class A, B, or C network can be subdivided into many smaller groups. Subnetting treats a subdivision of a single Class A, B, or C network as if it were a network itself. By doing so, a single Class A, B, or C network can be subdivided into many nonoverlapping subnets.

The needs for subnetting are both technical and administrative, as documented in the following list:

- All organizations connected to the Internet (and not using IP address translation) are required to use IP networks registered with the NIC.
- IP protocols enforce the following grouping concept: All hosts in the same group must not be separated by an IP router.
- A corollary to the grouping concept is this: Hosts separated by an IP router must be in separate groups.
- Without subnetting, the smallest group is a single, entire Class A, B, or C network number.
- Without subnetting, the NIC would be woefully short of assignable networks.
- With subnetting, the NIC can assign one or a few network numbers to an organization, and then the organization can subdivide those networks into subnets of more usable sizes.

An example drives these points home. Consider all network interfaces in Figure 6-27, and note which ones are not separated by a router.

Figure 6-27 *Backdrop for Discussing Numbers of Different Networks/Subnetworks*

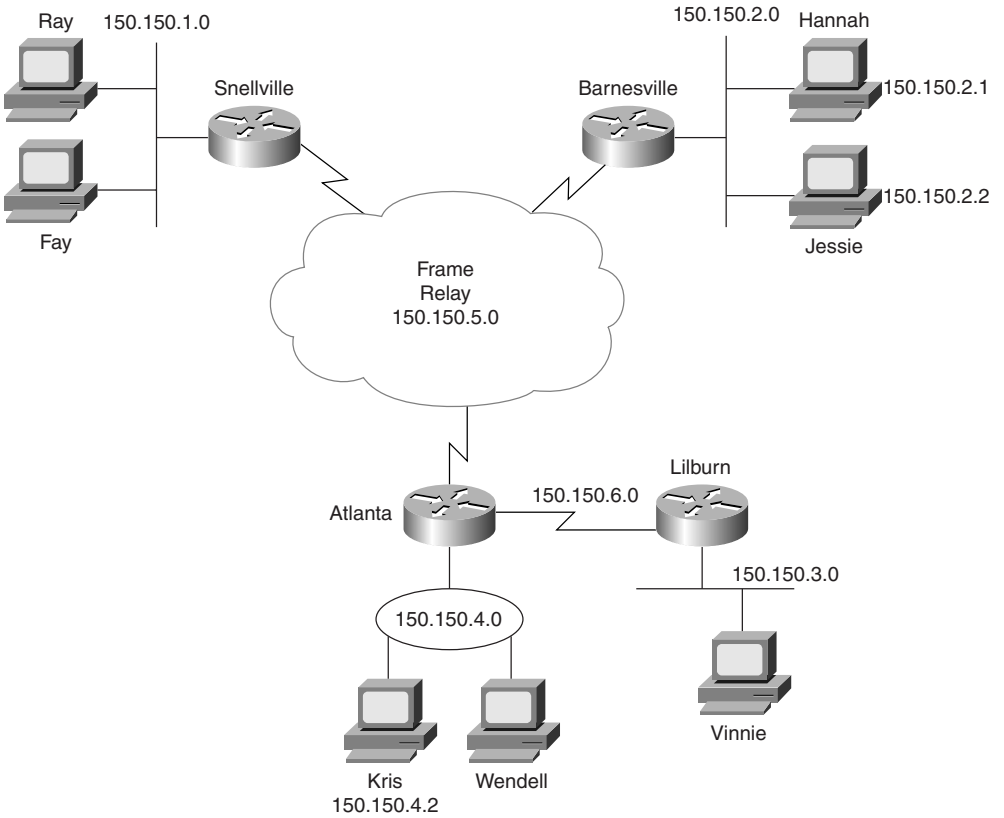


The design in Figure 6-27 requires six groups, each of which is a Class B network. The four LANs each use a single Class B network. In other words, the LANs attached to routers A, B, C, and D are each a separate network. Additionally, the two serial interfaces composing the point-to-point serial link between routers C and D use the same network because these two interfaces are not separated by a router. Finally, the three router interfaces composing the Frame Relay network with routers A, B, and C are not separated by an IP router and would compose the sixth network. (*Note:* Other Frame Relay IP addressing options would require one or two more IP network numbers for this physical network.)

However, this design would not be allowed if it were connected to the Internet. The NIC would not assign six separate registered Class B network numbers—in fact, you probably would not even get one Class B because most of the Class B addresses already are assigned. You are more likely to get a couple of Class C networks, and the NIC would expect you to use subnetting.

Figure 6-28 illustrates a more realistic example that uses basic subnetting.

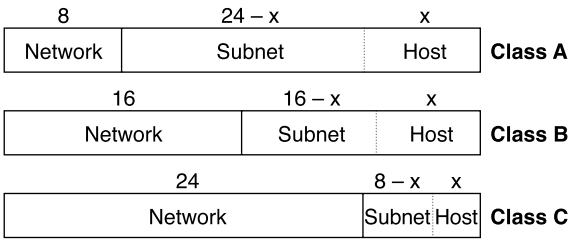
Figure 6-28 *Using Subnets*



As in Figure 6-27, the design in Figure 6-28 requires six groups. Unlike Figure 6-27, this figure uses six subnets, each of which is a subnet of a single Class B network. This design subnets Class B network 150.150.0.0, which has been assigned by the NIC. The IP network designer has chosen a mask of 255.255.255.0, the last octet of which implies 8 host bits. Because it is a Class B network, there are 16 network bits. Therefore, there are 8 subnet bits, which happen to be bits 17 through 24—in other words, the third octet. Notice that each subnet number in the figure shows a different value in the third octet, representing each different subnet number. In other words, this design numbers or identifies each different subnet using the third octet.

When subnetting, a third part of an IP address appears in the middle of the address—namely, the subnet part of the address. This field is created by "stealing" bits from the host part of the address. The size of the network part of the address never shrinks—in other words, Class A, B, and C rules still apply when defining the size of the network part of an address. Figure 6-29 shows the format of addresses when subnetting.

Figure 6-29 *Address Formats When Subnetting Is Used*



Three portions of the address now exist: network, subnet, and host. Class rules determine the size of the network part. The subnet mask determines the size of the host part—the number of bits of value 0 in the subnet mask defines the number of host bits. The remaining bits define the size of the subnet part of the address. For example, a mask of 255.255.255.224, used with a Class C network, implies five host bits. (The mask can be more easily converted to decimal using the table in Appendix B, "Decimal to Hexadecimal Binary Conversion Table.") The mask has five binary 0s at the end, implying five host bits. As shown in Figure 6-29, a Class C network has 24 network bits. That leaves three subnet bits—not many, but it still provides more groupings than if this Class C network was not subnetted at all!

The number of host bits implies how many valid host addresses exist in the subnet; $2^{hostbits}$ minus 2 special reserved cases is the formula. Similarly, the number of subnet bits implies the number of valid subnets of a network, assuming that the same mask is used on all subnets; $2^{subnetbits}$ is the formula. Two special subnets, the "zero subnet" and the "broadcast subnet," were reserved in years past but are now usable. However, when you get a test question about the number of possible subnets, the "right" answer is $2^{subnetbits} - 2$.

Binary View of Subnetting

Computers, especially routers, do not think about IP addresses in terms of conventions. They think in terms of 32-bit binary numbers, which is fine because, technically, that's what IP addresses really are. Also, computers use a mask to define the structure of these binary IP addresses. A full understanding of what that means is not difficult—in fact, if you understood the last section of the book, you already understand the structure of an IP address. However, getting accustomed to doing the binary math in your head is challenging for most of us, particularly if you don't do it every day.

Understanding Binary-to-Decimal and Decimal-to-Binary Conversion If you already know how binary works, how binary-to-decimal and decimal-to-binary conversion works, and how to convert IP addresses from decimal to binary and back, skip this note.

IP addresses are 32-bit binary numbers, written as a series of decimal numbers, separated by periods. To examine an address in its true form, binary, you need to convert from decimal to binary. To put a 32-bit binary number in the decimal form that is needed when configuring a router, you need to convert the 32-bit number back to decimal, 8 bits at a time.

One key to the conversion process for IP addresses is remembering these facts: When converting from one format to the other, each decimal number represents eight bits. When converting from decimal to binary, each decimal number converts to an eight-bit number. So address 150.150.2.1, 150, when converted to its eight-bit binary equivalent, is 10010010. How do you know that? For now, look in the conversion chart in Appendix B. The next byte, another decimal 150, is converted to 10010010. The third byte, decimal 2, is converted to 00000010; finally, the fourth byte, decimal 1, is converted to 00000001. The combined series of 8-bit numbers is the 32-bit IP address—in this case, 10010010 10010010 00000010 00000001.

If you start with the binary version of the IP address, you first separate it into four sets of eight digits. *Then you convert each set of eight binary digits to its binary equivalent*. For example, 10010010100100100000001000000001 is not a useful way to write an IP address. First, separate it into four sets of eight digits: 10010010 10010010 00000010 00000001. Then look in the conversion chart in Appendix B, and find that the first 8-bit number converts to 150, and so does the second set. The third set of 8 bits converts to 2, and the fourth converts to 1—giving you 150.150.2.1.

Using the chart in Appendix B makes this much easier—but you will not have the chart on the exam, of course! So you can do a couple of things. First, you can learn how to do the conversion. The book does not cover it, but a couple of web sites referenced at the end of this note can help. The other alternative is to use the chart when studying, and study the examples that show you how to manipulate IP addresses and find the right answers to the test questions without doing any binary math. If that works for you, you actually do not need to be speedy and proficient at doing binary-to-decimal and decimal-to-binary, conversions.

One last important fact: When subnetting, the subnet and host parts of the address might span only part of a byte of the IP address. So when converting from binary to decimal and decimal to binary, the rule of doing this using an eight-bit binary number is indeed true. However, when thinking about subnetting, you will need to ignore byte boundaries and think about IP addresses as 32-bit numbers without specific byte boundaries. But that will be explained later.

Some sites that might help you if you want more information are

For basic information on base 10, base 2 (binary), and conversion practice, visit www.ibilce.unesp.br/courseware/datas/numbers.htm#mark2.

For a description of the conversion process, try doit.ort.org/course/inforep/135.htm.

For another description of the conversion process, try www.goshen.edu/compsci/mis200/decbinary.htm.

For some free video classes that cover binary, conversion, and subnetting, go to www.learntosubnet.com.

An example of several IP addresses, in binary, will make subnetting a little clearer. The subnet part of an address identifies the different subdivisions of this network. An address with a different value in the subnet field, as compared with a second address, is considered to be in a different subnet. For example, examine the following three IP addresses that are part of Table 6-12 and are valid addresses in Figure 6-28.

Table 6-12 *Subnet Part of Sample Addresses*

Address in Decimal	Address in Binary
150.150.2.1	1001 0110 1001 0110 **0000 0010** 0000 0001
150.150.2.2	1001 0110 1001 0110 **0000 0010** 0000 0010
150.150.4.2	1001 0110 1001 0110 **0000 0100** 0000 0010

The example shows that the subnet field consists of bits 17 through 24 (the entire third byte). 150.150.2.1 and 150.150.2.2 are in the same subnet because they are in the same Class B network and because *their subnet fields have the same value* (0000 0010). 150.150.4.2 is in a different subnet of the same Class B network because the subnet field has a different value than the first two addresses (0000 0100). 150.150.4.2 must be physically located with at least one IP router between itself and 150.150.2.1 and 150.150.2.2. Similarly, addresses 150.150.2.1 and 150.150.2.2 must not be separated by a router, as is shown in Figure 6-28.

Routers understand the format of addresses using a mask. When subnetting is used, the mask is called a subnet mask. The mask is useful in that it defines the size and location of the host part of the address. In the example of network 150.150.0.0, using the third octet as the subnet part of the address, subnet mask 255.255.255.0 is used. Logically, the mask is interpreted this way:

- The subnet mask, 255.255.255.0, in binary is 11111111 11111111 11111111 00000000.
- The size of the network field is based on Class A, B, and C rules and does not consider the subnet mask. In this case, the network part is 16 bits long because the address is a Class B address.
- The size of the host part of the address is defined by the number of binary 0s in the subnet mask. There are eight binary 0s at the end of the mask.
- The subnet part is the remaining part of the address—bits 17 through 24, or, in other words, the third octet.

Routers and other computers take advantage of the subnet masks to answer a very frequently considered question:

What subnet is this address in?

For a computer, it's simple binary math—a Boolean AND between the IP address and the subnet mask. An example will help. Consider the following binary example. (*Note:* Appendix B has a binary-to-decimal conversion chart.)

Address	150.150.2.1	1001 0110 1001 0110 0000 0010 0000 0001
Mask	255.255.255.0	1111 1111 1111 1111 1111 1111 0000 0000
Result	150.150.2.0	1001 0110 1001 0110 0000 0010 0000 0000

A Boolean AND between the address and the mask results in the subnet number. The term *mask* comes from the idea that, because all the host bits are binary 0s in the mask, all those same bits in the IP address fall out to binary 0 as well. That's because, in a Boolean AND, both bits must be binary 1 for the result to be binary 1. Otherwise, the result is binary 0. So the mask of 255.255.255.0 masks out the last octet, leaving a subnet number that contains the same first three octets as the address, but a decimal 0 in the last octet.

If you understand the basic idea but would like additional examples to make it more clear, read on. In the next section, the four-step method to answer all IP addressing questions will be described. Also, in Appendix C, "Subnetting Practice: 25 Subnetting Questions," 25 IP addressing practice questions are available, each with the binary math worked out for performing the Boolean AND.

Finally, any Cisco-oriented IP addressing coverage would be incomplete without a discussion of *prefix notation*. Prefix notation describes a mask, but with fewer characters. Therefore, output from commands is briefer.

Prefix notation simply denotes the number of binary 1s in a mask, preceded by a "/". In other words, for subnet mask 255.255.255.0, the equivalent prefix notation would be /24 because there are 24 consecutive binary 1s in the mask. When talking about subnets, you can say things like, "That subnet uses a slash 24 prefix" instead of saying something like, "That subnet uses a mask of 255.255.255.0." The first option is simply easier to say, and it is often more convenient to use when you get used to it.

So far, you have read about all the basic information you need to understand IP addressing for a typical networking job. Now, for further preparation for the exam, the next section takes you through a process that will help you answer questions on the exam quickly and accurately.

Four Steps to Answering IP Addressing Questions

You must master IP addressing and subnetting to succeed as a network engineer. To pass the CCNA exam, you must at least be able to answer a few questions about subnetting. For most networking jobs, the ability to think about IP addresses and quickly decipher the structure and meaning of the address is a prerequisite for the job.

The exam will test your abilities with questions that go something like this:

- Given a network number and a mask, how many subnets are there, and how many hosts are there per subnet?
- Given an address and a mask, what is the subnet number?
- Given an address and a mask, what is the subnet broadcast address?
- Given an address and a mask, what are the valid IP addresses in the subnet?

This section teaches a process by which you can ignore the four preceding questions and think about IP addressing and subnetting like you normally would for your job. Coincidentally, if you follow the four-step process in this section, you would answer all of the preceding questions without much binary math. This four-step process is designed to help you learn how to do the math in your head—when you get the idea, you will not need to memorize every little step. In fact, with five examples in this chapter and 25 more in an appendix, you will be able to practice enough to easily memorize the process.

For reference, the steps in the process are as follows:

Step 1 Identify the structure of the IP address.

Step 2 Create the chart that will be used in Steps 3 and 4.

Step 3 Derive the subnet number and the first valid IP address.

Step 4 Derive the broadcast address and the last valid IP address.

In this section, you will see two different examples worked out in great detail and three examples worked out with less detail. With these five examples, plus the extra practice in Appendix C, you will be able to master the process of answering the previous questions about any IP address. If you already can answer those questions about the five examples, you know plenty about IP subnetting for passing the CCNA exam! Here are the examples:

8.1.4.5, mask 255.255.0.0
130.4.102.1, mask 255.255.255.0
199.1.1.100, mask 255.255.255.0
130.4.102.1, mask 255.255.252.0
199.1.1.100, mask 255.255.255.224

If you can confidently answer all four questions for all five example address/mask pairs, you might want to skip the rest of the IP addressing section. If you look at the examples and think that you simply need more practice, turn to Appendix C, which has 25 additional examples completely worked out. If you want to see how I got the answers to these five, along with the process to do it quickly without needing a binary-to-decimal conversion chart, read on! By the way, you can check your work against Table 6-13, which summarizes the results of all the steps for each of the five sample addresses and masks.

Table 6-13 *Five Addresses/Masks, with IP Addressing Information Explained*

	8.1.4.5/16	130.4.102.1/24	199.1.1.100/24	130.4.102.1/22	199.1.1.100/27
Mask	255.255.0.0	255.255.255.0	255.255.255.0	255.255.252.0	255.255.255.224
Number of network bits	8	16	24	16	24
Number of host bits	16	8	8	10	5
Number of subnet bits	8	8	0	6	3
Number of hosts per subnet	$2^{16} - 2$	$2^8 - 2$	$2^8 - 2$	$2^{10} - 2$	$2^5 - 2$
Number of subnets	$2^8 - 2$	$2^8 - 2$	0	$2^6 - 2$	$2^3 - 2$
Subnet number	8.1.0.0	130.4.102.0	199.1.1.0	130.4.100.0	199.1.1.96
First valid IP address	8.1.0.1	130.4.102.1	199.1.1.1	130.4.100.1	199.1.1.97
Broadcast address	8.1.255.255	130.4.102.255	199.1.1.255	130.4.103.255	199.1.1.127
Last valid IP address	8.1.255.254	130.4.102.254	199.1.1.254	130.4.103.254	199.1.1.126
Range of valid IP addresses	8.1.0.1–8.1.255.254	130.4.102.1–130.4.102.254	199.1.1.1–199.1.1.254	130.4.100.1–130.4.103.254	199.1.1.97–199.1.1.126

Step 1: Identify the Structure of the IP Address

Two tasks must be performed in Step 1. First, the rules that define the three parts of an IP address must be remembered and applied. In other words, you must decide how many of the 32 bits in the address comprise the network, subnet, and host portions of the address. The second task is to remember and apply two easy formulas that tell you how many subnets exist and how many hosts there are per subnet.

The first task requires that you remember these three facts:

The network part of the address is always defined by class rules.

The host part of the address is always defined by the mask; binary 0s in the mask mean that the corresponding address bits are part of the host field.

The subnet part of the address is what's left over in the 32-bit address.

Table 6-14 lists these three key facts, beside the first example. If you have forgotten the ranges of values in the first octet for addresses in Class A, B, and C networks, refer to Table 6-10 earlier in the chapter.

Table 6-14 *First Example, with Rules for Learning Network, Subnet, and Host Part Sizes*

Step	Example	Rules to Remember
Address	8.1.4.5	N/A
Mask	255.255.0.0	N/A
Number of network bits	8	Always defined by Class A, B, C
Number of host bits	16	Always defined as number of binary 0s in mask
Number of subnet bits	8	32 – (network size + host size)

In this example, there are eight network bits because the address is in a Class A network, 8.0.0.0. There are 16 host bits because, when you convert 255.255.0.0 to binary, there are 16 binary 0s—the last 16 bits in the mask. (If you do not believe me, look at Appendix B, in the binary-to-decimal conversion chart. 255 decimal is eight binary 1s, and 0 decimal is eight binary 0s.) The size of the subnet part of the address is what's left over, or 8 bits.

The same logic applies to all five examples in Table 6-13. For example, 130.4.102.1 is in a Class B network, so there are 16 network bits. The prefix of /24 has only eight binary 0s, implying eight host bits, which leaves eight subnet bits. The third example, 199.1.1.100, with mask 255.255.255.0, is not using subnetting at all. How do you know? Well, 199.1.1.100 is in a Class C network, which means that there are 24 network bits. The mask has eight binary 0s, yielding eight host bits, with no bits remaining for the subnet part of the address. In fact, if you remembered that the default mask for Class C networks is 255.255.255.0, you might have already realized that no subnetting was being used.

Most of us can calculate the number of host bits easily if the mask uses only decimal 255s and 0s. When the mask uses other values besides 0 and 255, deciphering the number of host bits is more difficult. Examining the subnet masks in binary help overcome the challenge. Consider the masks used in the last two examples of Table 6-13, as shown in Table 6-15.

Table 6-15 *Masks Used in Examples 4 and 5*

Mask in Decimal	Mask in Binary
255.255.252.0	1111 1111 1111 1111 1111 1100 0000 0000
255.255.255.224	1111 1111 1111 1111 1111 1111 1110 0000

The number of host bits implied by a mask becomes more apparent after converting the mask to binary. In the first mask, 255.255.252.0, there are 10 binary 0s, implying a 10-bit host field. Because that mask is used with a Class B address (130.4.102.1), implying 16 network bits, there are six remaining subnet bits. In the last example, the mask has only five binary 0s, for five host bits. Because the mask is used with a Class C address, there are 24 network bits, leaving only three subnet bits.

The process so far is straightforward: The class rules define the network part, the mask binary 0s define the host part, and what's left over defines the size of the subnet part. The only big problem occurs when the mask is tricky, which is true in examples 4 and 5. When the mask is tricky, you have two alternatives for deciding how many host bits are defined:

Convert the mask to binary, using any method for conversion at your disposal, and count the number of zeros.

Convert the mask to binary after memorizing the nine decimal and binary values in Table 6-16—these are the only nine valid decimal values used in a subnet mask. Converting a mask to binary without the use of any tools will be much faster.

Table 6-16 lists the only valid decimal values in a mask and their binary equivalents.

Table 6-16 *Decimal and Binary Values in a Single Octet of a Valid Subnet Mask*

Decimal	Binary
0	0000 0000
128	1000 0000
192	1100 0000
224	1110 0000
240	1111 0000
248	1111 1000
252	1111 1100
254	1111 1110
255	1111 1111

Binary conversion of a subnet mask, without the use of a calculator, PC, or decimal-to-binary conversion chart, becomes easy after memorizing this chart. The binary equivalents of 255 and decimal 0 are obvious. The other seven values are not! But notice the values in succession: Each value has an additional binary 1 and one less binary 0. Each mask value, in succession, shows a mask value that reduces the number of host bits by 1 and adds 1 to the size of the subnet field. If you simply memorize each decimal value and its binary equivalent, converting masks from decimal to binary will be a breeze.

This section on subnetting outlines a four-step process that helps you answer most subnetting questions. Step 1 includes two tasks, as mentioned in the first paragraph of the description of Step 1. The first task is to figure out how many network, host, and subnet bits exist in the address. However, the exam probably will not include simple questions like, "Given this address and mask, how big is the host part of the address?" A question that could be on your exam would be something more like this: "Given an address and mask, how many subnets are there? And how many hosts are there in a single subnet?" Well, two simple formulas provide the answers, and the formulas are based on the information that you just learned how to derive:

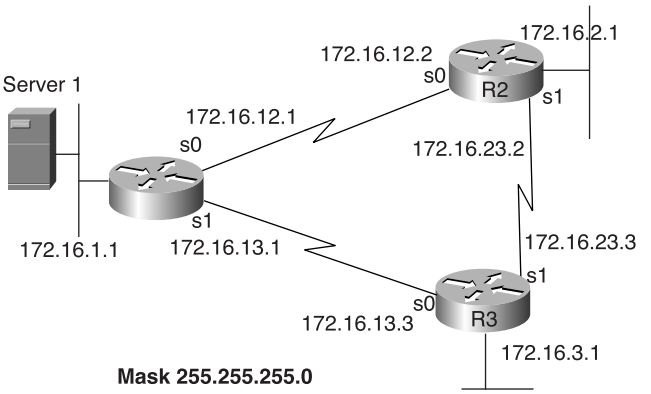
$$\text{Number of subnets} = 2^{\text{number-of-subnet-bits}} - 2$$
$$\text{Number of hosts per subnet} = 2^{\text{number-of-host-bits}} - 2$$

The formulas calculate the number of things that can be numbered using a binary number and subtract 2 for the two special cases. IP addressing conventions define that two subnets per network not be used and that two hosts per subnet not be used. One reserved subnet, the subnet that has all binary 0s in the subnet field, is called the zero subnet. The subnet with all binary 1s in the subnet field is called the broadcast subnet—and it also is reserved. (Well, in fact, you can use both these subnets on a Cisco router, but it is recommended that you avoid using them. On the exam, the "right" answer is that you do not use them—hence the $2^{\text{number-of-subnet-bits}} - 2$ formula.) IP addressing conventions also reserve two IP addresses per subnet: the first (all binary 0s in the host field) and last (all binary 1s in the host field) addresses. No tricks exist to make these two addresses usable—they are indeed always reserved.

Now you have seen how to identify the components of an IP address and to derive how many subnets are allowed and how many hosts exist per subnet. However, the right answer in real life might not be so simple! In some networks, the network design engineer chooses an identical subnet mask to be used for all subnets of the same network. However, the engineer could choose different masks for each and every subnet. Consider, for example, the network in Figure 6-30.

This design requires only two hosts in the subnet for each serial link, but the mask allows for 2^8–2, or 254, hosts per subnet! If the network engineer had wanted to conserve the address space because the network was going to grow a lot, he might have chosen to use a different mask for the serial link's subnet. Likewise, the LAN where Server1 resides might have more than 254 hosts on it—in which case a single subnet with 8 host bits would not be big enough. A mask such as 255.255.252.0 with 10 host bits might have been a better choice.

Figure 6-30 *Sample Network, with Router IP Addresses Shown*



The term *variable-length subnet mask* (VLSM) describes a single Class A, B, or C network in which more than one subnet mask is used. Before you complete your CCNP certification, you will need to fully understand how to implement a VLSM design. If you understood the previous paragraph, you already understand the concepts behind VLSM. Planning, implementing, and operating a network that uses VLSM requires a full, comfortable understanding of subnetting.

On the CCNA exam, if you are asked, "How many subnets of network *x* are there when using mask *y*?", assume that the mask is identical for all subnets of the network and that no VLSM is in use.

Now you can complete a chart similar to Table 6-14 for each of the five examples. Completing charts such as Table 6-14 covers all the parts of Step 1 in the four-step process of answering all IP address questions. Table 6-13 contains answers for all five example for Step 1. The details of Step 1 are as follows:

Step 1 Identify the structure of the IP address.

- **A** Identify the size of the network part of the address, based on Class A, B, and C rules.
- **B** Identify the size of the host part of the address, based on the number of binary 0s in the mask. If the mask is tricky, use the chart of typical mask values to convert the mask to binary more quickly.
- **C** The size of the subnet part is what's "left over"; mathematically, it is 32 – (number of network + host bits).
- **D** Declare the number of subnets, which is $2^{\text{number-of-subnet-bits}} - 2$.
- **E** Declare the number of hosts per subnet, which is $2^{\text{number-of-host-bits}} - 2$.

Step 2: Create the Chart to Be Used in Steps 3 and 4

Step 2 simply introduces you to a tool that you can use to teach yourself how to quickly derive the subnet number and first valid address (Step 3) and the broadcast address and last valid address (Step 4). This tool, which is a chart that you will complete, organizes the given information (address and mask) along with the information that you need to derive. This tool will help you learn to perform subnetting with no binary math, which you will need to do in most networking jobs.When you become accustomed to the logic, you will simply skip this step mentally.

All the examples in this chapter, as well as those in Appendix C, use this chart to explain the process of deriving the subnet number, broadcast address, and valid IP addresses in the subnet.

Table 6-17 is an example subnet chart

Table 6-17 *Subnet Chart—Generic*

	Octet 1	Octet 2	Octet 3	Octet 4
Address				
Mask				
Subnet number				
First valid address				
Broadcast				
Last valid address				

A typical exam question might supply the address and mask and then ask you for one of the other four values—the subnet number, the first valid address, the broadcast address, or the last valid address. The subnet chart begins with the address and mask. The subnet number and the subnet broadcast address are what you are after, along with the first and the last valid IP addresses in the subnet. In real life, you seldom care about the broadcast address, but you do care about the range of valid IP addresses in the subnet. Finding the last valid IP address in the subnet is much easier if you know the broadcast address, so calculating the broadcast is still very useful. And of course, it might be on the exam.

Four tasks must be completed in Step 2:

A Create a generic subnet chart, as in Table 6-17.

B Write in the IP address and subnet mask.

C Draw a vertical line between the columns with 255 and 0 in the mask if the mask is “easy” (those with all mask octets either 0 or 255). Draw a box around the column with the mask octet that is not 0 or 255 if the mask is not easy.

D Copy the values of the address to the left of the line or box into the final four rows. After that, you are ready for the more important concepts of Steps 3 and 4.

Tables 6-18 and 6-19 list two of the examples from Table 6-13, after the first two tasks of Step 1 have been completed.

Table 6-18 *Subnet Chart—130.102.1/24, After Two of Four Tasks in Step 2 Are Complete*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	255	0
Subnet number				
First valid address				
Broadcast				
Last valid address				

Table 6-19 *Subnet Chart—130.102.1/22, After Two of Four Tasks in Step 2 Are Complete*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	252	0
Subnet number				
First valid address				
Broadcast				
Last valid address				

The table simply lists the information provided in a typical question, up to this point. The only difficulty might be if the question lists the prefix, as did Table 6-13, and the subnet chart wants the mask. The prefix is the number of binary 1s that start the mask. With a 24-bit prefix (Table 6-18's example), the mask is pretty easy to derive—you need three 255s to start the mask to get 24 prefix bits. The example in Table 6-19, using a 22-bit prefix, is tougher—two 255s to start the mask get you 16 prefix bits. You need 6 more prefix bits, so you remember that 252 decimal has six binary 1s, to create a mask with 22 prefix bits. Refer to Table 6-16 for a reminder of the mask values in decimal and binary.

The third task of Step 2 requires you to draw either a line or a box. You use a line when the mask is "easy," and you use a box when the mask is "hard." Easy masks use only 255s or 0s in the mask; whereas, hard masks have one octet in which the mask is something besides 255 or 0. I call a mask octet that's not a 255 or 0 the *interesting* octet because it is the octet that gives everyone heartburn when first learning subnetting. The line separates the 255s from the 0s when the mask is easy, and the box both separates the host part from the rest of the address and draws attention to the tricky part of the logic used in this four-step process.

You decide where to draw either the line or the box based on some easy rules. When an easy mask is used, you draw a vertical line between the host part of the address and the rest—in other words, a line between the 255s and the 0s in the mask. When a hard mask is used, a decimal math trick can be used to complete the subnet chart—and the math trick applies to the octet with the non-255 or 0 value in the mask. So, with a hard mask, instead of drawing a line, you draw a box around the column of the interesting octet. Table 6-20 gives an example of using a line when the mask is easy, and Table 6-21 gives an example of using a box around the interesting octet when the mask is hard.

Table 6-20 *Subnet Chart—130.102.1/24, After Drawing the Vertical Line*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	255	0
Subnet number				
First valid address				
Broadcast				
Last valid address				

Table 6-21 *Subnet Chart—130.102.1/22, After Drawing a Box Around the Interesting Octet*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	252	0
Subnet number				
First valid address				
Broadcast				
Last valid address				

Finally, you should complete the chart for everything to the left of the line or the box. To complete the chart, look at the original IP address octets to the left of the line or box, and copy those into the subnet, first valid address, broadcast, and last valid address fields. Note that only octets to the left of the line or box should be copied—the interesting octet, which is inside the box, should not be copied. Tables 6-22 and 6-23 show the same two examples, at the end of Step 2 of the process.

Table 6-22 *Subnet Chart—130.102.1/24, at the End of Step 2*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	255	0
Subnet number	130	4	102	
First valid address	130	4	102	
Broadcast	130	4	102	
Last valid address	130	4	102	

Table 6-23 *Subnet Chart—130.102.1/22, at the End of Step 2*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	252	0
Subnet number	130	4		
First valid address	130	4		
Broadcast	130	4		
Last valid address	130	4		

Keep in mind that Step 2 gives you a tool to derive the right answers without binary math. At the end of Step 2, you do not have the answers yet! Step 3 will give you two more answers—namely, the subnet number and first valid IP address. Step 4 will give you the broadcast address, along with the last valid IP address in the subnet. So be patient—the usefulness of the chart will be apparent in the coming steps.

For completeness, Tables 6-24, 6-25, and 6-26 show the completed subnet charts for the other three examples in Table 6-13.

Table 6-24 *Subnet Chart—8.1.4.5/16, at the End of Step 2*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	8	1	4	5
Mask	255	255	0	0
Subnet number	8	1		
First valid address	8	1		
Broadcast	8	1		
Last valid address	8	1		

Table 6-25 *Subnet Chart—199.1.1.100/24, at the End of Step 2*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	199	1	1	100
Mask	255	255	255	0
Subnet number	199	1	1	
First valid address	199	1	1	
Broadcast	199	1	1	
Last valid address	199	1	1	

Table 6-26 *Subnet Chart—199.1.1.100/27, at the End of Step 2*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	199	1	1	100
Mask	255	255	255	224
Subnet number	199	1	1	
First valid address	199	1	1	
Broadcast	199	1	1	
Last valid address	199	1	1	

Now you are ready to see how to complete the chart in Steps 3 and 4. First, a summary of Steps 1 and 2:

Step 1 Identify the structure of the IP address.

A Identify the size of the network part of the address, based on Class A, B, and C rules.

B Identify the size of the host part of the address, based on the number of binary 0s in the mask. If the mask is tricky, use the chart of typical mask values to convert the mask to binary more quickly.

C The size of the subnet part is what's "left over"; mathematically, it is 32 – (number of network + host bits).

D Declare the number of subnets, which is $2^{\text{number-of-subnet-bits}} - 2$.

E Declare the number of hosts per subnet, which is $2^{\text{number-of-host-bits}} - 2$.

Step 2 Create the chart that will be used in Steps 3 and 4.

A Create a generic subnet chart.

B Write down the IP address and subnet mask in the first two rows of the chart.

C If an easy mask is used, draw a vertical line between the 255s and the 0s in the mask, from top to bottom of the chart. If a hard mask is used, draw a box around the column of the interesting octet.

D Copy the address octets to the left of the line or the box into the final four rows of the chart.

Step 3: Derive the Subnet Number and the First Valid IP Address

NOTE Before beginning the explanations in Steps 3 and 4 of this process, you might want to choose to not read parts of this section. If you are not yet comfortable with subnetting when easy masks are used, you might benefit from focusing on the example that covers the easy masks used and ignoring the hard example. Then, when you are comfortable with the easy ones, reread all of Steps 3 and 4.

To help you focus on the easy example, the parts of the text that cover the difficult example are shaded. If you want to focus on the easier part, get good at that, and then read these steps a second time, you should ignore the parts that are shaded—for now.

Earlier, you learned that computers perform a Boolean AND of the address and mask to find the subnet number. If you want a way to find the subnet number without using any binary math or binary conversions, read on! In Step 3, you will see how to quickly derive the subnet number, even when the mask is a tricky one.

You should understand how to perform the Boolean AND, so the ANDs for each of the five examples from Table 6-13 follow. However, if you already understand Boolean ANDs, feel free to go right past these next five examples and on to the rest of the four-step process.

Address	8.1.4.5	0000 1000 0000 0001 0000 0100 0000 0101
Mask	255.255.0.0	1111 1111 1111 1111 **0000 0000 0000 0000**
AND result (subnet number)	8.1.0.0	0000 1000 0000 0001 0000 0000 0000 0000

Address	130.4.102.1	1000 0010 0000 0100 0110 0110 0000 0001
Mask	255.255.255.0	1111 1111 1111 1111 1111 1111 **0000 0000**
AND result (Subnet number)	130.4.102.0	1000 0010 0000 0100 0110 0110 0000 0000

Address	199.1.1.100	1100 0111 0000 0001 0000 0001 0110 0100
Mask	255.255.255.0	1111 1111 1111 1111 1111 1111 **0000 0000**
AND result (subnet number)	199.1.1.0	1100 0111 0000 0001 0000 0001 0000 0000

Address	130.4.102.1	1000 0010 0000 0100 0110 0110 0000 0001
Mask	255.255.252.0	1111 1111 1111 1111 1111 11**00 0000 0000**
AND result (subnet number)	130.4.100.0	1000 0010 0000 0100 0110 0100 0000 0000

Address	199.1.1.100	1100 0111 0000 0001 0000 0001 0110 0100
Mask	255.255.255.224	1111 1111 1111 1111 1111 1111 111**0 0000**
AND result (subnet number)	199.1.1.96	1100 0111 0000 0001 0000 0001 0110 0000

The four-step process avoids all binary conversions and Boolean AND operations. On the exam, you will not have a binary conversion chart, and in real life, you will not always be sitting at your desk near your tools for deriving the subnet number. Mastering this simple process gives you the subnet number, if you already know the address and mask.

At Step 3 of the four-step process, you must complete one more task to derive the subnet number if the mask is easy, or two more tasks if the mask is hard. When that is finished, a single task derives the first valid IP address in the subnet. The hardest part requires you to do a little arithmetic.

First, in the subnet number row, write down a decimal 0 for all octets to the right of the line or the box. Do not write down a 0 in the octet inside the box. If the mask is easy (all 255s or 0s), you should have a complete subnet number. If the mask is hard (one mask octet with something besides a 255 or a 0), you should have three octets of the subnet number completed, with the interesting octet not yet filled in. Tables 6-27 and 6-28 show two examples, with this first task in Step 3 completed.

The subnet number in example in Table 6-27 is complete. In Step 2, you wrote down all the parts of the subnet number to the left of the line in the chart—the ones that were identical to the IP address. In Step 3, you wrote 0s to the right of the line—that's it! In fact, because the mask is easy, many of you probably would have been able to answer the question, "Given 130.4.102.1/24, what is the subnet number?" well before this point in the process. If you are new to subnetting, feel free to use the subnet chart even for examples with an easy mask—it still gives you the right answer.

Table 6-27 *Subnet Chart—130.4.102.1/24, at Step 3's First Task, Placing 0s After the Line*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	255	0
Subnet number	130	4	102	**0**
First valid address	130	4	102	
Broadcast	130	4	102	
Last valid address	130	4	102	

Table 6-28 *Subnet Chart—130.4.102.1/22, after Step 3's Task of Placing 0s After the Box*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	252	0
Subnet number	130	4		**0**
First valid address	130	4		
Broadcast	130	4		
Last valid address	130	4		

You must complete the second task of Step 3 for the example in Table 6-18 because this is a difficult example. The box in the chart should remind you that additional work needs to be done. Now, if you could already look at the mask and derive the subnet number of 130.4.100.0, you probably would not need to use this process!

For the rest of you who do not do this often enough to be able to do the math in your head, the next task gives you the right value for the subnet number's interesting octet. First, you find what I will call the *magic number*—which is 256 minus the mask's interesting octet. In this case, you have 256 – 252, or a magic number of 4. Then you find the multiple of the magic number that is the closest to the address's interesting octet but less than or equal to it. In this example, 100 is a multiple of the magic number (4×25), and this multiple is ≤ 102. The next multiple of the magic number, which is 104, is, of course, more than 102, so that's not the right number. The multiple of the magic number closest to but not more than the address's interesting octet is the subnet's interesting octet value.

In other words, plug in 100 for the third octet of the subnet number in Table 6-28.

Finally, the third task in Step 3 requires you to derive the first valid IP address in the subnet. You might recall that two values are reserved in each subnet. The first of these reserved values is the subnet number itself. The number that is one bigger than the subnet number is the first valid IP address in the subnet. As you might guess, finding the first valid IP address is easy when you know the subnet number:

> To find the first valid IP address in the subnet, copy the subnet number, but add 1 to the fourth octet.

That's all! Table 6-29 shows the same examples as Table 6-27 at the completion of Step 3.

Table 6-29 *Subnet Chart—130.4.102.1/24, After Step 3*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	255	0
Subnet number	130	4	102	**0**
First valid address	130	4	102	**1**
Broadcast	130	4	102	
Last valid address	130	4	102	

In Table 6-29, the first three octets of the subnet number and first valid address were already filled in. Because this example uses an easy mask, you just write down a 0 in all the octets to the right of the line—the last octet—to get the subnet number. To get the first valid address, just add 1 to the last octet of the subnet number.

Tables 6-30 shows the same examples as Table 6-28 at the completion of Step 3.

Table 6-30 *Subnet Chart—130.4.102.1/22, After Step 3*

	Octet 1	Octet 2	Octet 3	Octet 4	Comments
Address	130	4	102	1	
Mask	255	255	252	0	
Subnet number	130	4	**100**	**0**	Magic = 256 – 252 = 4; 4 × 25 = 100, closest multiple ≤ 102.
First valid address	130	4	**100**	**1**	Add 1 to subnet's last octet
Broadcast	130	4			
Last valid address	130	4			

In Table 6-30, the first two octets of the subnet number and first valid address were already filled in because they are to the left of the box around the third octet—the interesting octet, in this case. In the subnet number, the last octet is 0 because it is to the right of the box. To find the interesting octet value, just find the closest multiple of the magic number that's not bigger than that octet of the address—100 in this case. The subnet number is 130.4.100.0. To get the first valid address, just add 1 to the last octet of the subnet number, giving you 130.4.100.1.

For completeness, Tables 6-31, 6-32, and 6-33 list the other three examples from Table 6-13, with the subnet number and first IP address rows completed.

Table 6-31 *Subnet Chart—8.1.4.5/16, at the End of Step 3*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	8	1	4	5
Mask	255	255	0	0
Subnet number	8	1	**0**	**0**
First valid address	8	1	**0**	**1**
Broadcast	8	1		
Last valid address	8	1		

Table 6-32 *Subnet Chart—199.1.1.100/24, at the End of Step 3*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	199	1	1	100
Mask	255	255	255	0
Subnet number	199	1	1	**0**
First valid address	199	1	1	**1**
Broadcast	199	1	1	
Last valid address	199	1	1	

Table 6-33 *Subnet Chart—199.1.1.100/27, At the End of Step 3*

	Octet 1	Octet 2	Octet 3	Octet 4	Comments
Address	199	1	1	100	
Mask	255	255	255	224	
Subnet number	199	1	1	**96**	Magic = 256 – 224 = 32; 3 ×32 = 96, closest multiple ≤ 100
First valid address	199	1	1	**97**	Add 1 to fourth octet of subnet
Broadcast	199	1	1		
Last valid address	199	1	1		

For Step 3, you have two main alternatives to find the subnet number: Use a Boolean AND or use the process with the subnet chart. The following summary encapsulates the logic for using first three steps of the four-step process that avoids any binary math.

Step 1 Identify the structure of the IP address.

- **A** Identify the size of the network part of the address, based on Class A, B, and C rules.
- **B** Identify the size of the host part of the address, based on the number of binary 0s in the mask. If the mask is tricky, use the chart of typical mask values to convert the mask to binary more quickly.
- **C** The size of the subnet part is what's "left over"; mathematically, it is 32 – (number of network + host bits).
- **D** Declare the number of subnets, which is $2^{\text{number-of-subnet-bits}} - 2$.
- **E** Declare the number of hosts per subnet, which is $2^{\text{number-of-host-bits}} - 2$.

Step 2 Create the chart that will be used in Steps 3 and 4.

- **A** Create a generic subnet chart.
- **B** Write down the IP address and subnet mask in the first two rows of the chart.
- **C** If an easy mask is used, draw a vertical line between the 255s and the 0s in the mask, from top to bottom of the chart. If a hard mask is used, draw a box around the column of the interesting octet.
- **D** Copy the address octets to the left of the line or the box into the final four rows of the chart.

Step 3 Derive the subnet number and the first valid IP address.

- **A** Write down 0s in the subnet octets to the right of the line or the box.
- **B** If the mask is hard and there is a box in the chart, use the magic number trick to find the value of the subnet's interesting octet.
- **C** To derive the first valid IP address, copy the first three octets of the subnet number, and add 1 to the fourth octet of the subnet number.

Step 4: Derive the Subnet Broadcast Address and the Last Valid IP Address

NOTE *Reminder:* If you want to focus on the easy example and then come back and reread this section when you are comfortable with the easy one, continue to avoid reading the shaded areas of this section.

Step 4 completes the process using the subnet chart to answer the most typical subnetting questions. You need to know how to calculate the subnet broadcast address as well as the last IP address in the subnet, which is one less than the broadcast address. After you calculate the broadcast address, finding the last valid IP address is easy.

You should know how to derive the subnet broadcast address using binary math, and you also should know how to do it with this four-step decimal math process. The binary math is easy when you have the subnet number in binary: Simply change all the host bit values in the subnet number to binary 1s. You can examine the math behind the five examples from Table 6-13 from the information that follows. The host parts of the addresses, masks, subnet numbers, and broadcast addresses are in bold.

Address	8.1.4.5	0000 1000 0000 0001 **0000 0100 0000 0101**
Mask	255.255.0.0	1111 1111 1111 1111 **0000 0000 0000 0000**
AND result (subnet number)	8.1.0.0	0000 1000 0000 0001 **0000 0000 0000 0000**
Broadcast address	8.1.255.255	0000 1000 0000 0001 **1111 1111 1111 1111**

Address	130.4.102.1	1000 0010 0000 0100 0110 0110 **0000 0001**
Mask	255.255.255.0	1111 1111 1111 1111 1111 1111 **0000 0000**
AND result (subnet number)	130.4.102.0	1000 0010 0000 0100 0110 0110 **0000 0000**
Broadcast address	130.4.102.255	1000 0010 0000 0100 0110 0110 **1111 1111**

Address	199.1.1.100	1100 0111 0000 0001 0000 0001 0110 0100
Mask	255.255.255.0	1111 1111 1111 1111 1111 1111 **0000 0000**
AND result (subnet number)	199.1.1.0	1100 0111 0000 0001 0000 0001 0000 0000
Broadcast address	199.1.1.255	1100 0111 0000 0001 0000 0001 **1111 1111**

Address	130.4.102.1	1000 0010 0000 0100 0110 01**10 0000 0001**
Mask	255.255.252.0	1111 1111 1111 1111 1111 11**00 0000 0000**
AND result (subnet number)	130.4.100.0	1000 0010 0000 0100 0110 01**00 0000 0000**
Broadcast address	130.4.103.255	1000 0010 0000 0100 0110 01**11 1111 1111**

Address	199.1.1.100	1100 0111 0000 0001 0000 0001 011**0 0100**
Mask	255.255.255.224	1111 1111 1111 1111 1111 1111 111**0 0000**
AND result (subnet number)	199.1.1.96	1100 0111 0000 0001 0000 0001 011**0 0000**
Broadcast address	199.1.1.127	1100 0111 0000 0001 0000 0001 011**1 1111**

The four-step process avoids all binary conversions and Boolean AND operations. On the exam, you will not have a binary conversion chart, and in real life, you will not always be sitting at your desk near your tools for deriving the subnet number and broadcast address. So mastering this four-step process gives you the broadcast address and highest valid IP address.

For Step 4 in the four-step process, you must complete one more task to derive the broadcast address if the mask is easy, or two more tasks if the mask is hard. When that is finished, a single step is used to derive the last valid IP address in the subnet. The hardest part requires you to do a little arithmetic.

First, in the broadcast address, write down a decimal 255 for all octets to the right of the line or the box. Do not write down a 255 in the octet inside the box. If the mask is easy (all 255s or 0s), you should have a complete broadcast address. If the mask is hard (one mask octet with something besides a 255 or a 0), you should have three octets of the broadcast address, with the interesting octet not yet filled in. Tables 6-34 and 6-35 show the same two familiar examples, with this first task in Step 4 completed.

Table 6-34 *Subnet Chart—130.4.102.1/24, After Step 4's First Task, Placing 255s After the Line*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	255	0
Subnet number	130	4	102	0
First valid address	130	4	102	1
Broadcast	130	4	102	**255**
Last valid address	130	4	102	

Table 6-35 *Subnet Chart—130.4.102.1/22, After Step 4's First Task, Placing 255s After the Box*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	252	0
Subnet number	130	4	100	0
First valid address	130	4	100	1
Broadcast	130	4		**255**
Last valid address	130	4		

The broadcast address in example in Table 6-34 is complete. In Step 2, you wrote down all the parts of the broadcast address to the left of the line in the chart—the ones that were identical to the IP address. In Step 4, you wrote 255s to the right of the line—that's it! In fact, because the

mask is easy, many of you probably would have been able to answer the question, "Given 130.4.102.1/24, what is the broadcast address?" well before this point in the process. If you are new to subnetting, feel free to use the subnet chart even for examples with an easy mask—it still gives you the right answer.

For the more difficult example, you must complete the second task of Step 4 for the example in Table 6-36. The box in the chart, as well as the mask value of 252, should remind you that additional work needs to be done, and it is obvious that one octet of the broadcast address simply is not filled in yet.

The next task in Step 4 gives you the right value for the broadcast address's interesting octet in the difficult example. First, remind yourself about the magic number. The magic number is 256 minus the mask's interesting octet. In this case, you have 256 – 252, or a magic number of 4. Then you add the magic number to the interesting octet value of the subnet number and subtract 1. The result is the broadcast address's value in the interesting octet. In this case, the value is 100 + 4 (magic number) – 1 = 103.

Finally, the third task in Step 4 requires you to derive the last valid IP address in the subnet. You might recall that two values are reserved in each subnet. The first of these values is the subnet number itself, and the other reserved value is the broadcast address. Numerically, the subnet number has the lowest numeric value, and the broadcast address has the largest. The last valid IP address in the subnet can be calculated easily: *Copy the broadcast address, except subtract 1 from the fourth octet*. That's all! Tables 6-36 and 6-37 show the same examples as Tables 6-34 and 6-35, at the completion of Step 4.

Table 6-36 *Subnet Chart—130.4.102.1/24, After Step 4*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	255	0
Subnet number	130	4	102	0
First valid address	130	4	102	1
Broadcast	130	4	102	**255**
Last valid address	130	4	102	**254**

Table 6-37 *Subnet Chart—130.4.102.1/22, After Step 4*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	130	4	102	1
Mask	255	255	252	0
Subnet number	130	4	100	0
First valid address	130	4	100	1
Broadcast	130	4	**103**	**255**
Last valid address	130	4	**103**	**254**

On the exam, you are more likely to see questions that ask you about the range of valid IP addresses in a subnet than those that simply ask you for the broadcast address for the subnet. By finding the subnet number and broadcast addresses, however, you can easily derive the lowest valid IP address and the highest, and easily answer the questions.

Tables 6-38, 6-39, and 6-40 shows the remaining three examples from Table 6-13.

Table 6-38 *Subnet Chart—8.1.4.5/16, at the End of Step 4*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	8	1	4	5
Mask	255	255	0	0
Subnet number	8	1	0	0
First valid address	8	1	0	1
Broadcast	8	1	**255**	**255**
Last valid address	8	1	**255**	**254**

Table 6-39 *Subnet Chart—199.1.1.100/24, at the End of Step 4*

	Octet 1	Octet 2	Octet 3	Octet 4
Address	199	1	1	100
Mask	255	255	255	0
Subnet number	199	1	1	0
First valid address	199	1	1	1
Broadcast	199	1	1	**255**
Last valid address	199	1	1	**254**

Table 6-40 *Subnet Chart—199.1.1.100/27, at the End of Step 4*

	Octet 1	Octet 2	Octet 3	Octet 4	Comments
Address	199	1	1	100	N/A
Mask	255	255	255	224	N/A
Subnet number	199	1	1	96	Magic = 256 – 224 = 32; 3 × 32 = 96, closest multiple ≤ 100
First valid address	199	1	1	97	Add 1 to fourth octet of subnet
Broadcast	199	1	1	**127**	96 + magic (32) – 1 = 127
Last valid address	199	1	1	**126**	Subtract 1 from fourth octet of broadcast address

The four-step process for dissecting IP addresses is now complete. The following list summarizes the tasks in each step:

Step 1 Identify the structure of the IP address.

- **A** Identify the size of the network part of the address, based on Class A, B, and C rules.
- **B** Identify the size of the host part of the address, based on the number of binary 0s in the mask. If the mask is tricky, use the chart of typical mask values to convert the mask to binary more quickly.
- **C** The size of the subnet part is what's "left over"; mathematically, it is 32 – (number of network + host bits).
- **D** Declare the number of subnets, which is $2^{\text{number-of-subnet-bits}} - 2$.
- **E** Declare the number of hosts per subnet, which is $2^{\text{number-of-host-bits}} - 2$.

Step 2 Create the chart that will be used in Steps 3 and 4.

- **A** Create a generic subnet chart.
- **B** Write down the IP address and subnet mask in the first two rows of the chart.
- **C** If an easy mask is used, draw a vertical line between the 255s and the 0s in the mask, from top to bottom of the chart. If a hard mask is used, draw a box around the column of the interesting octet.
- **D** Copy the address octets to the left of the line or the box into the final four rows of the chart.

Step 3 Derive the subnet number and the first valid IP address.

- **A** Write down 0s in the subnet octets to the right of the line or the box.
- **B** If the mask is hard and there is a box in the chart, use the magic number trick to find the value of the subnet's interesting octet.
- **C** To derive the first valid IP address, copy the first three octets of the subnet number, and add 1 to the fourth octet of the subnet number.

Step 4 Derive the broadcast address and the last valid IP address.

- **A** Write down 255s in the broadcast address octets to the right of the line or the box.
- **B** If the mask is hard and there is a box in the chart, use the magic number trick to find the value of the broadcast address's interesting octet.
- **C** To derive the last valid IP address, copy the first three octets of the broadcast address, and subtract 1 from the fourth octet of the broadcast address.

So now what? Well, if you are confident that you can answer subnetting questions easily, move on to the next major section. If not, you might want to work on Appendix C, which has many practice examples.

Deciding What the Other Subnets Are

When I wrote the four-step process section for IP subnetting, I had two goals in mind. As I mentioned earlier, you need to answer subnetting questions quickly and confidently on the exam. Network engineers also think about subnetting every day, so it's great to be able to do the math in your head.

So far, you have dealt with questions regarding a single subnet. You might also need to address the question: "What are the other valid subnets of this network?" The details of answering this question follow next.

First, the question needs a better definition—or at least a more complete one. The question you might get on the CCNA exam might be better stated like this: "If the same subnet mask is used for all subnets of this Class A, B, or C network, what are the valid subnets?" IP design conventions do not require the engineer to use the same mask for every subnet. If you recall, the topic of VLSMs was explained briefly back in Step 1 of the four-step process to analyze a single address. On the CCNA exam, the question "What are all the subnets" assumes that

VLSMs are not in use. The question is, simply put, if the same mask is used for all subnets, what are the subnet numbers?

Another easy decimal process (three steps, this time!) lists all the valid subnets, given the network number and the only mask used on that network. This three-step process assumes that the size of the subnet part of the address is, at most, eight bits in length. The same general process can be expanded to work when the size of the subnet part of the address is more than eight bits, but that expanded process is not described here.

The three-step process uses a chart that I call the subnet list chart. I made up the name just for this book, simply as another tool to use. Table 6-41 presents a generic version of the subnet list chart.

Table 6-41 *Three-Step Process Generic Subnet List Chart*

	Octet 1	Octet 2	Octet 3	Octet 4
Network number				
Mask				
Subnet zero				
First valid subnet				
Next valid subnet				
Last valid subnet				
Broadcast subnet				

You list the known network number and subnet mask as the first step in the process. If the question gives you an IP address and mask instead of the network number and mask, just write down the network number of which that IP address is a member. (Remember, this three-step process assumes that the subnet part of the addresses is eight bits or less.) The second of the three steps is to copy the network number into the row labeled Subnet zero. *Subnet zero*, or the *zero subnet*, is numerically the first subnet, but it is one of the two reserved subnet numbers in a network. You can use the zero subnet on a Cisco router if you configure the global configuration command **ip zero-subnet**. For the purposes of answering questions on the exam about the number of valid subnets in a network, consider the zero subnet unusable. In real life, do not use the zero subnet if you do not have to.

Tables 6-42 and 6-43 list two familiar examples, with the first two steps completed.

Table 6-42 *Subnet List Chart—130.4.0.0/24*

	Octet 1	Octet 2	Octet 3	Octet 4
Network number	130	4	0	0
Mask	255	255	255	0
Subnet zero	130	4	0	0

Table 6-43 *Subnet List Chart—130.4.0.0/22*

	Octet 1	Octet 2	Octet 3	Octet 4
Network number	130	4	0	0
Mask	255	255	252	0
Subnet zero	130	4	0	0

Step 3 is the last step, but it is repeated many times. This last step uses the magic number—remember, the magic number is 256 minus the mask octet value in the interesting octet. With this process of finding all the subnet numbers, the interesting octet is the octet that contains *all* of the subnet part of the addresses. (Remember, the process assumes eight or less subnet bits!) In both Tables 6-42 and 6-43, the interesting octet is the third octet.

The third and final step in the process to find all the subnet numbers goes like this: Starting with the last row that's completed in the table, do the following:

1. Copy all three noninteresting octets from the previous line.
2. Add the magic number to the previous interesting octet, and write that down as the value of the interesting octet.
3. Repeat the last two tasks until the next number that you would write down in the interesting column is 256. (Don't write that one down—it's not valid.)

The idea behind the process of finding all the subnets becomes apparent by reviewing the same two examples used earlier. First, Table 6-44 lists the example with the easy mask.

Table 6-44 *Subnet List Chart—130.4.0.0/24 Completed*

	Octet 1	Octet 2	Octet 3	Octet 4
Network number	130	4	0	0
Mask	255	255	255	0
Subnet zero	130	4	0	0
First valid subnet	130	4	1	0
Next valid subnet	130	4	2	0
Next valid subnet	130	4	3	0
Next valid subnet	130	4	4	0
(Skipping a bunch)	130	4	X	0
Last valid subnet	130	4	254	0
Broadcast subnet	130	4	255	0

The logic behind how the process works might be better understood by looking at the first few entries and then the last few entries. The zero subnet is easily found because it's the same number as the network number. The magic number is 256 – 255 = 1 in this case. Essentially, you increment the third octet (in this case) by the magic number for each successive subnet number.

In the middle of the table, one row is labeled Skipping a bunch. Rather than make the book even bigger, I left out several entries but included enough that you could see that the subnet number's third octet just gets bigger by 1, in this case, for each successive subnet number.

Looking at the end of the table, the last entry lists 255 in the third octet. 256 decimal is never a valid value in any IP address, and the directions said to not write down a subnet with 256 in it, so the last number in the table is 130.4.255.0. *The last subnet is the broadcast subnet, which is the other reserved subnet number. The subnet before the broadcast subnet is the highest, or last, valid subnet number.*

Many people might even refer to these subnets using just the third octet. If all subnets of a particular organization were in network 130.4.0.0, with mask 255.255.255.0, you might simply say "subnet five" when referring to subnet 130.4.5.0.

You might see an exam question such as, "How many valid subnets are there for network 130.4.0.0, mask 255.255.255.0?" Table 6-44 lists them—well, at least some of them. Intuitively, there are 254 valid subnets with this numbering scheme—subnets 1 through 254, using common terminology.

The process works the same with difficult subnet masks, even though the answers are not as intuitive. Table 6-45 lists the answers for the second example, using a 22-bit prefix.

Table 6-45 *Subnet List Chart—130.4.0.0/22*

	Octet 1	Octet 2	Octet 3	Octet 4
Network number	130	4	0	0
Mask	255	255	252	0
Subnet zero	130	4	0	0
First valid subnet	130	4	4	0
Next valid subnet	130	4	8	0
Skip a lot	130	4	X	0
Last valid subnet	130	4	248	0
Broadcast subnet	130	4	252	0

Most of us would not guess that 130.4.252.0 was the broadcast subnet for this latest example. However, adding the magic number 4 to 252 would give you 256 as the next subnet number, which is not valid—so, 130.4.252.0 is indeed the broadcast subnet.

The three-step process to find all the subnet numbers of a network is:

1. Write down the network number and subnet mask in the first two rows of the subnet list chart.
2. Write down the network number in the third row. This is the zero subnet, which is one of the two reserved subnets.
3. Do the following two tasks, stopping when the next number that you would write down in the interesting column is 256. (Don't write that one down—it's not valid.)
   - Copy all three noninteresting octets from the previous line.
   - Add the magic number to the previous interesting octet, and write that down as the value of the interesting octet.

Start Extra Credit

CIDR, Private Addressing, and NAT

Connecting to the Internet using only a registered network number or several registered network numbers uses a very straightforward and obvious convention. With registered network numbers, no other organization connected to the Internet will have conflicting IP addresses. In fact, this convention is part of the reason the global Internet functions well.

In the early and mid-1990s, concern arose that the available networks would be completely assigned so that some organizations would not be capable of connecting to the Internet. This one fact was the most compelling reason for the advent of IP Version 6 (IPv6). (The version discussed in this book is Version 4. Version 5 was defined for experimental reasons and was never deployed.) Version 6 calls for a much larger address structure so that the convention of all

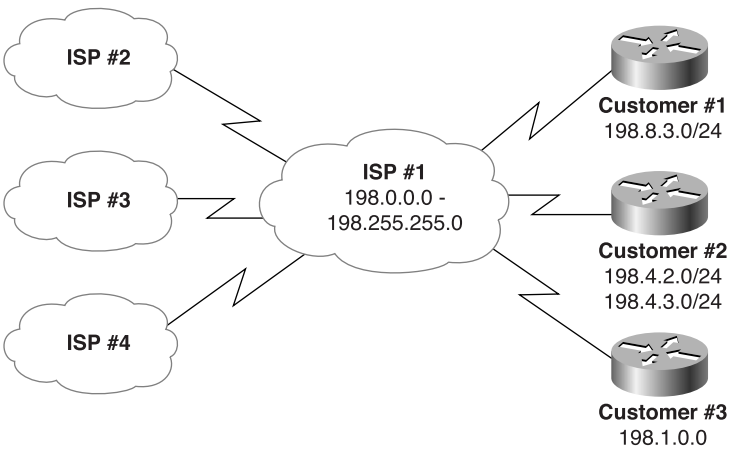
organizations using unique groupings (networks) of IP addresses would still be reasonable—the numbers of IPv6-style networks would reach into the trillions and beyond. That solution is still technically viable and possibly one day will be used because IPv6 is still evolving in the marketplace.

Three other functions of IP have been used to reduce the need for IP Version 4 (IPv4) registered network numbers. Network Address Translation (NAT), often used in conjunction with private addressing, allows organizations to use unregistered IP network numbers internally and still communicate well with the Internet. Classless interdomain routing (CIDR) is a feature used by Internet service providers (ISPs) to reduce the wasting of IP addresses and to delay the day when we run out of IP addresses.

CIDR

CIDR is a convention, defined in RFC 1817 (www.ietf.org/rfc/rfc1817.txt), that calls for aggregating multiple network numbers into a single routing entity. CIDR was actually created to help the scalability of Internet routers—imagine a router in the Internet with a route to every Class A, B, and C network on the planet! By aggregating the routes, fewer routes would need to exist in the routing table. Consider Figure 6-31.

Figure 6-31 *Typical Use of CIDR*



Class C networks 198.0.0.0 through 198.255.255.0 (they might look funny, but they are valid Class C network numbers) are registered networks for an ISP. All other ISPs' routing tables would have a separate route to each of the 2^{16} networks without CIDR. However, as the illustration shows, now the other ISPs' routers will have a single route to 198.0.0.0/8—in other words, a route to all hosts whose IP address begins with 198. More than two million Class C

networks alone exist, but CIDR has helped Internet routers reduce their routing tables to a more manageable size, in the range of 70,000 routes at the end of 1999.

By using a routing protocol that exchanges the mask as well as the subnet/network number, a "classless" view of the number can be attained. In other words, treat the grouping as a math problem, ignoring the Class A, B, and C rules. For instance, 198.0.0.0/8 (198.0.0.0, mask 255.0.0.0) defines a set of addresses whose first 8 bits are equal. This route is advertised by ISP 1 to the other ISPs, who need a route only to 198.0.0.0/8. In its routers, ISP 1 knows which Class C networks are at which customer sites. This is how CIDR gives Internet routers a much more scalable routing table, by reducing the number of entries in the tables.

Historically, ISPs found ways to use CIDR to allow better use of the IP Version 4 address space. Imagine that Customer 1 and Customer 3 will need a maximum of 10 and 20 IP addresses, respectively—ever. Each customer has only a router and a single Ethernet. Each customer could register its own Class C network, but if both did so, it would not be in the range already registered to the ISP.

To help CIDR work in the Internet, ISP 1 wants its customers to use IP addresses in the 198.*x*.*x*.*x* range. As a service, the ISP suggests to Customer 1 something like this: Use IP subnet 198.8.3.16/28, with assignable addresses 198.8.17 to 198.8.30. To Customer 3, who needs 20 addresses, ISP 1 suggests 198.8.3.32/27, with 30 assignable addresses (198.8.3.33 to 198.8.3.62). (Feel free to check the math with the IP addressing algorithms listed earlier.)

NOTE The notation with the " / " followed by the number is a common designation on Cisco routers meaning that the mask has that number of 1 bits. This number of 1 bits is called the *prefix*. In this case, the mask implied with prefix /27 would be 255.255.255.224.

The need for registered IP network numbers is reduced through CIDR. Instead of the two customers consuming two whole Class C networks, each consumes a small portion of a single Class C network. The ISP has customers use its IP addresses in a convenient range of values, so CIDR works well and enables the Internet to grow.

Private Addressing

A legitimate need exists for IP addresses that will never be used in the interconnected IP networks called the Internet. So when designing the IP addressing convention for such a network, an organization could pick and use any network number(s) it wanted, and all would be well. Of course, that's true until the organization decides to connect to the Internet—but that will be covered in the section "Network Address Translation."

When building a private network that will have no Internet connectivity, you can use IP network numbers called private Internets, as defined in RFC 1918, "Address Allocation for Private Internets" (www.ietf.org/rfc/rfc1918.txt). This RFC defines a set of networks that will never be assigned to any organization as a registered network number. Instead of using someone else's registered network numbers, you can use numbers in a range that are not used by anyone else. Table 6-46 shows the private address space defined by RFC 1918.

Table 6-46 *RFC 1918 Private Address Space*

Range of IP Addresses	Class of Networks	Number of Networks
10.0.0.0 to 10.255.255.255	A	1
172.16.0.0 to 172.31.255.255	B	16
192.168.0.0 to 192.168.255.255	C	256

In other words, any organization can use these network numbers. However, no organization is allowed to advertise these networks using a routing protocol on the Internet.

The IP Version 4 address space is conserved if all organizations use private addresses in cases for which there will never be a need for Internet connectivity. So the dreaded day of exhausting the registered IP Version 4 network numbers has been delayed again, in part by CIDR and in part by private addressing.

Private addressing's requirement that the privately addressed hosts cannot communicate with others through the Internet can be a particularly onerous restriction. The solution: private addressing with the use of Network Address Translation (NAT).

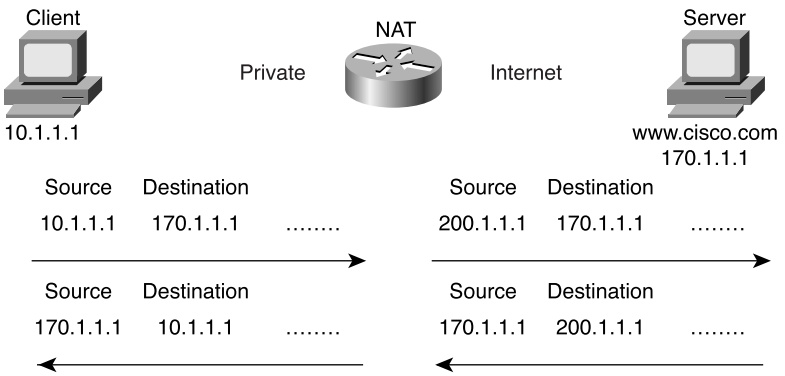
Network Address Translation

NAT is an RFC-defined function implemented in IOS that allows a host that does not have a valid registered IP address to communicate with other hosts through the Internet. The hosts might be using private addresses or addresses assigned to another organization; in either case, NAT allows these addresses that are not Internet-ready to continue to be used but still allow communication with hosts across the Internet.

NAT achieves its goal by using a valid registered IP address to represent the invalid address to the rest of the Internet. The NAT function changes the IP addresses as necessary inside each IP packet, as shown in Figure 6-32.

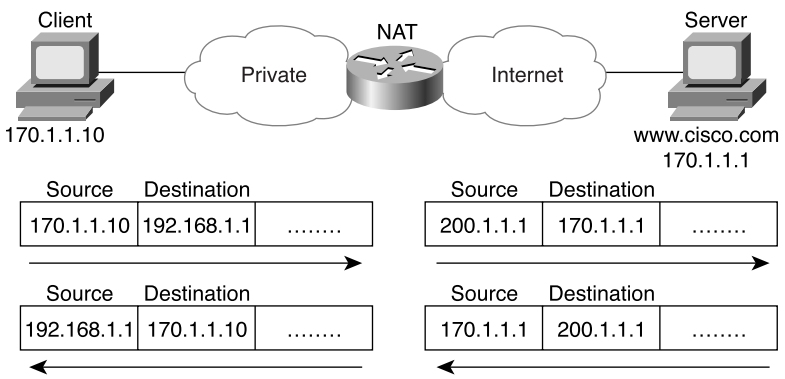
Notice that the packet's source IP address is changed when leaving the private organization, and the destination address is changed each time a packet is forwarded back into the private network. Network 200.1.1.0 has been registered as a network owned by Cisco in Figure 6-32, with address 200.1.1.1 configured as part of the NAT configuration. The NAT feature, configured in the router labeled NAT, performs the translation. As you might expect, NAT certainly requires more processing than simply routing the packet. Cisco does not recommend using NAT for a large volume of different hosts.

Figure 6-32 *NAT IP Address Swapping—Private Addressing*



NAT also can be used when the private organization is not using private addressing but is instead using a network number registered to another company. (A client company of mine originally had done just that—ironically, the company was using a network number registered to Cabletron, which my client saw used in a presentation by an ex-Cabletron employee who then worked at 3COM. The 3COM SE explained IP addressing using the Cabletron registered network number; my client liked the design and took him at his word—literally.) If one company inappropriately uses the same network number that is registered appropriately to a different company, NAT can be used, but both the source and the destination IP addresses will need to be translated. For instance, consider Figure 6-33, with Company A using a network that is registered to Cisco (170.1.0.0).

Figure 6-33 *NAT IP Address Swapping—Unregistered Networks*



In this case, the client in Company A cannot send a packet to 170.1.1.1—or, at least, if it did, the packet would never get to the real 170.1.1.1 in Cisco's network. That is because there is a very reasonable possibility that the private network has a route matching 170.1.1.1 in its routing table that points to some subnet inside the private company. When the DNS reply comes back

past the NAT router, the DNS reply is changed by NAT so that the client in Company A thinks that www.cisco.com's IP address is 192.168.1.1. NAT not only translates the source IP address in outgoing packets, but it also translates the destination. Likewise, packets returning to Company A have both the source and the destination IP addresses changed.

NAT uses terminology to define the various IP addresses used for translation. Table 6-47 summarizes the terminology and meaning.

Table 6-47 *NAT Addressing Terms*

Term	Meaning	Value in Figure 6-32
Inside local	Address of the host in the private network. When NAT is needed, this address is typically a private address or an address in a network registered to another organization.	170.1.1.10
Inside global	The Internet (global network) view of the inside local address. This address is in a network registered to the company responsible for the NAT router.	200.1.1.1
Outside global	This is the Internet (global network) view of the address of the host correctly attached to the Internet.	170.1.1.1
Outside local	When the private company reuses a network number registered to someone else, the outside local address represents the outside global address in the local (private) network. Because this address is used only in the private organization, it can be any IP address.	192.168.1.1

End Extra Credit

IP Configuration

23 Configure IP addresses.

24 Verify IP addresses.

You can easily configure a Cisco router to forward IP traffic when you know the details covered in this chapter so far. Tables 6-48 and 6-49 summarize many of the most common commands used for IP configuration and verification. Two sample network configurations, with both configuration and exec command output, follow. The Cisco IOS documentation is an excellent reference for additional IP commands; the Cisco Press book *Interconnecting Cisco Network Devices* is an excellent reference, particularly if you are not able to attend the instructor-led version of the class.

Table 6-48 *IP Configuration Commands*

Command	Configuration Mode
ip address *ip-address mask* [**secondary**]	Interface mode
ip host *name* [*tcp-port-number*] *address1* [*address2...address8*]	Global
ip route *network-number network-mask* {*ip-address* | *interface*} [*distance*] [**name** *name*]	Global
ip name-server *server-address1* [[*server- address2*]*…server-address6*]	Global
ip domain-lookup	Global
ip routing	Global
ip netmask-format {**bitcount** | **decimal** | **hexadecimal**}	Interface mode
ip default-network *network*	Global
ip classless	Global
ip host ***name*** [*tcp-port-number*] *address1* [*address2…address8*]	Global

Table 6-49 *IP Exec Commands*

Command	Function
show hosts	Lists all host names and corresponding IP addresses
show interfaces [*type number*]	Lists interface statistics, including IP address
show ip interface [*type number*]	Provides a detailed view of IP parameter settings, per interface
show ip interface brief	Provides a summary of all interfaces and their IP addresses
show ip route [*ip-address* [*mask*] [**longer-prefixes**]] | [*protocol* [*process-id*]]	Shows entire routing table or a subset if other parameters are entered
show ip arp [*ip-address*] [*host-name*] [*mac-address*] [*type number*]	Displays IP ARP cache
debug ip packet	Issues log messages for each IP packet

continues

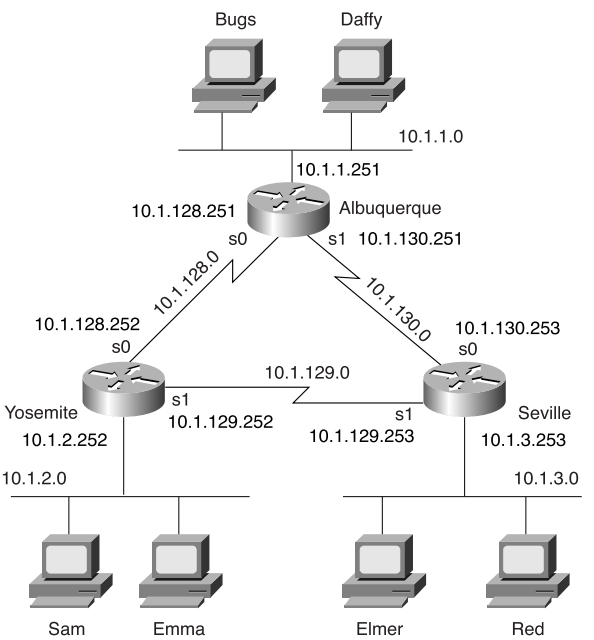
Table 6-49 *IP Exec Commands (Continued)*

Command	Function
terminal ip netmask-format {**bitcount** | **decimal** | **hexadecimal**}	Sets type of display for subnet masks in **show** commands
ping [*protocol* | **tag**] {*host-name* | *system-address*}	Sends and receives ICMP echo messages to verify connectivity
trace [*protocol*] [*destination*]	Sends a series of UDP packets with increasing TTL values to verify the current route to a host

Collectively, Figure 6-34 and Examples 6-2 through 6-4 show three sites, each with two serial links and one Ethernet. The following site guidelines were used when choosing configuration details:

- Use name servers at 10.1.1.100 and 10.1.2.100.
- Use host names from Figure 6-34.
- The router's IP addresses are to be assigned from the last few valid IP addresses in their attached subnets; use a mask of 255.255.255.0.

Figure 6-34 *Sample Network with Three Routers, with Point-to-Point Serial Links*



Example 6-2 *Albuquerque Router Configuration and Exec Commands*

```
Albuquerque#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Albuquerque(config)#interface serial 0
Albuquerque(config-if)#ip address 10.1.128.251 255.255.255.0
Albuquerque(config)#interface serial 1
Albuquerque(config-if)#ip address 10.1.130.251 255.255.255.0
Albuquerque(config)#interface ethernet 0
Albuquerque(config-if)#ip address 10.1.1.251 255.255.255.0

Albuquerque#show running-config
Building configuration...

Current configuration : 872 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Albuquerque
!
enable secret 5 $1$J3Fz$QaEYNIiI2aMu.3Ar.q0Xm.
!
!
ip name-server 10.1.1.100
ip name-server 10.1.2.100
!
interface Serial0
 ip address 10.1.128.251 255.255.255.0
!
interface Serial1
 ip address 10.1.130.251 255.255.255.0
!
interface Ethernet0
 ip address 10.1.1.251 255.255.255.0
!
no ip http server

banner motd ^C
  Should've taken a left turn here! This is Albuquerque...  ^C
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

continues

Example 6-2 *Albuquerque Router Configuration and Exec Commands (Continued)*

```
Albuquerque#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 3 subnets
C       10.1.1.0 is directly connected, Ethernet0
C       10.1.130.0 is directly connected, Serial1
C       10.1.128.0 is directly connected, Serial0

Albuquerque#terminal ip netmask-format decimal
Albuquerque#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0 255.255.255.0 is subnetted, 3 subnets
C       10.1.1.0 is directly connected, Ethernet0
C       10.1.130.0 is directly connected, Serial1

C       10.1.128.0 is directly connected, Serial0
Albuquerque#
```

Example 6-3 *Yosemite Router Configuration and Exec Commands*

```
Yosemite#show running-config
Building configuration...

Current configuration : 867 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Yosemite
!
enable secret 5 $1$J3Fz$QaEYNIiI2aMu.3Ar.q0Xm.
!
!
ip name-server 10.1.1.100
```

Example 6-3 *Yosemite Router Configuration and Exec Commands (Continued)*

```
ip name-server 10.1.2.100
!
interface Serial0
 ip address 10.1.128.252 255.255.255.0
 no fair-queue
!
interface Serial1
 ip address 10.1.129.252 255.255.255.0
!
interface Ethernet0
 ip address 10.1.2.252 255.255.255.0
!
no ip http server

banner motd ^C
   This is the Rootin-est Tootin-est Router in these here parts!  ^C
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
!
end

Yosemite#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
Serial0                10.1.128.252    YES manual up                    up
Serial1                10.1.129.252    YES manual up                    up
Ethernet0              10.1.2.252      YES manual up                    up
Yosemite#
```

Example 6-4 *Seville Router Configuration and Exec Commands*

```
Seville#show running-config
Building configuration...

Current configuration : 869 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Seville
!
!
enable secret 5 $1$J3Fz$QaEYNIiI2aMu.3Ar.q0Xm.
!
ip name-server 10.1.1.100
```

continues

Example 6-4 *Seville Router Configuration and Exec Commands (Continued)*

```
ip name-server 10.1.2.100
!
interface Serial0
 ip address 10.1.130.253 255.255.255.0
 no fair-queue
!
interface Serial1
 ip address 10.1.129.253 255.255.255.0
!
Ethernet0
 ip address 10.1.3.253 255.255.255.0
!
no ip http server
banner motd ^C
  Take a little off the top, Wabbit!  (Elmer)   ^C
!
line con 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
!
end

Seville#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route


Gateway of last resort is not set


10.0.0.0/24 is subnetted, 3 subnets
C       10.1.3.0 is directly connected, Ethernet0
C       10.1.130.0 is directly connected, Serial0
C       10.1.129.0 is directly connected, Serial1


Seville#show ip interface serial 1
Serial1 is up, line protocol is up
  Internet address is 10.1.129.253/24
  Broadcast address is 255.255.255.255
  Address determined by non-volatile memory
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
```

Example 6-4 *Seville Router Configuration and Exec Commands (Continued)*

```
  Outgoing access list is not set
  Inbound  access list is not set
  Proxy ARP is enabled
  Security level is default
  Split horizon is disabled
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  IP fast switching is enabled
  IP fast switching on the same interface is enabled
  IP Flow switching is disabled
  IP Feature Fast switching turbo vector
  IP multicast fast switching is disabled
  IP multicast distributed fast switching is disabled
  IP route-cache flags are Fast
  Router Discovery is disabled
  IP output packet accounting is disabled
  IP access violation accounting is disabled
  TCP/IP header compression is disabled
  RTP/IP header compression is disabled
  Probe proxy name replies are disabled
  Policy routing is disabled
  Network address translation is disabled
  WCCP Redirect outbound is disabled
  WCCP Redirect inbound is disabled
  WCCP Redirect exclude is disabled
  BGP Policy Mapping is disabled

Seville#show interface serial 0
Serial0 is up, line protocol is up
  Hardware is HD64570
  Internet address is 10.1.130.253/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

continues

Example 6-4 *Seville Router Configuration and Exec Commands (Continued)*

```
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

Seville#show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.1.3.102              0   0060.978b.1301  ARPA   Ethernet0
Internet  10.1.3.253              -   0000.0c3e.5183  ARPA   Ethernet0

Seville#debug ip packet
IP packet debugging is on
Seville#ping 10.1.130.251
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.130.251, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 80/81/84 ms
Seville#
00:09:38: IP: s=10.1.130.251 (local), d=10.1.130.251 (Serial1), len 100, sending
00:09:38: IP: s=10.1.130.251 (Serial1), d=10.1.130.253 (Serial1), len 100, rcvd 3
00:09:38: IP: s=10.1.130.253 (local), d=10.1.130.251 (Serial1), len 100, sending
00:09:38: IP: s=10.1.130.251 (Serial1), d=10.1.130.253 (Serial1), len 100, rcvd 3
00:09:38: IP: s=10.1.130.253 (local), d=10.1.130.251 (Serial1), len 100, sending
00:09:38: IP: s=10.1.130.251 (Serial1), d=10.1.130.253 (Serial1), len 100, rcvd 3
00:09:38: IP: s=10.1.130.253 (local), d=10.1.130.251 (Serial1), len 100, sending
00:09:38: IP: s=10.1.130.251 (Serial1), d=10.1.130.253 (Serial1), len 100, rcvd 3
00:09:38: IP: s=10.1.130.253 (local), d=10.1.130.251 (Serial1), len 100, sending
00:09:38: IP: s=10.1.130.251 (Serial1), d=10.1.130.253 (Serial1), len 100, rcvd 3
Seville#
```

As you see in Figure 6-34, the IP addresses chosen for the interfaces are shown. At the beginning of Example 6-2, the engineer uses configuration mode to configure the IP addresses. The **show running-config** command displays the results of the configuration, along with some other details that were already configured.

Notice that the configuration matches the output of the **show interface**, **show ip interface**, and **show ip interface brief** commands. In Example 6-3, the IP addresses in the configuration match the output of **show ip interface brief**. If these details did not match, one common oversight would be that you are looking at the configuration in NVRAM, not in RAM. Be sure to use the **show running-config** or **write terminal** commands to see the active configuration.

The subnet mask in the output of **show** commands uses prefix notation. For example, 10.1.4.0/24 means 24 network and subnet bits, leaving 8 host bits with this subnetting scheme. The **terminal ip netmask** command can be used to change this formatting for screen output during this session, as seen in Example 6-2.

Example 6-4 shows the ARP cache generated by the **show ip arp** output. The first entry shows the IP address (10.1.3.102) and MAC address of another host on the Ethernet. The timer value of 0 implies that the entry is very fresh—the value grows with disuse and eventually times out. One entry is shown for the router's Ethernet interface itself, which never times out of the ARP table.

The **debug ip packet** output in Example 6-4 lists one entry per IP packet sent and received. This command is a very dangerous one—it could crash almost any production router because of the added overhead of processing the debug messages. Imagine a router that forwards 50,000 packets per second, needing to send 50,000 messages per second to a console that's running at 9600 bps! The router would buffer the messages and exhaust all its memory doing so, and the router would crash. Also notice that the output shows both the source and destination IP addresses.

As compared with a working, real configuration in a production network, these examples omit the needed routing protocol configuration. The routing table in Example 6-4 does not list all subnets because the routing protocol configuration has not been added—notice that all the routers have a value of C beside them, which, according to the legend shown at the beginning of the command, means that the route describes a connected subnet. However, because you also need to know how to configure static routes rather than show the configuration of a routing protocol next, the **ip route** commands in Example 6-5 have been added to Albuquerque, which adds static routes. Examples 6-6 and 6-7 contain **show** commands executed after the new configuration was added.

Example 6-5 *Static Routes Added to Albuquerque*

```
ip route 10.1.2.0 255.255.255.0 10.1.128.252
ip route 10.1.3.0 255.255.255.0 10.1.130.253
```

Example 6-6 *Albuquerque Router Exec Commands, After Adding Static Routes for 10.1.2.0 and 10.1.3.0*

```
Albuquerque#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route


Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 5 subnets
S       10.1.3.0 [1/0] via 10.1.130.253
S       10.1.2.0 [1/0] via 10.1.128.252
C       10.1.1.0 is directly connected, Ethernet0
C       10.1.130.0 is directly connected, Serial1
C       10.1.128.0 is directly connected, Serial0
Albuquerque#ping 10.1.128.252

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.128.252, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/8 ms
```

continues

Example 6-6 *Albuquerque Router Exec Commands, After Adding Static Routes for 10.1.2.0 and 10.1.3.0 (Continued)*

```
! Note: the following extended ping command will result in some debug messages
! on Yosemite in Example 6-7.

Albuquerque#ping
Protocol [ip]:
Target IP address: 10.1.2.252
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 10.1.1.251
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.2.252, timeout is 2 seconds:
. . . . .
Success rate is 0 percent (0/5)
Albuquerque#
```

Example 6-7 **show ip route** *on Yosemite, After Adding Static Routes to Albuquerque*

```
Yosemite#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 3 subnets
C       10.1.2.0 is directly connected, Ethernet0
C       10.1.129.0 is directly connected, Serial1
C       10.1.128.0 is directly connected, Serial0
Yosemite#ping 10.1.128.251

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.128.251, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/8 ms
Yosemite#ping 10.1.1.251

Type escape sequence to abort.
```

Example 6-7 **show ip route** *on Yosemite, After Adding Static Routes to Albuquerque (Continued)*

```
Sending 5, 100-byte ICMP Echos to 10.1.1.251, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Yosemite#debug ip icmp
ICMP packet debugging is on
Yosemite#
Yosemite#show debug
Generic IP:
  ICMP packet debugging is on
Yosemite#

!NOTE: the following debug messages are a result of the extended ping
!command issued on Albuquerque in Example 6-6;
!these messages are generated by Yosemite!

ICMP: echo reply sent, src 10.1.2.252, dst 10.1.1.251
ICMP: echo reply sent, src 10.1.2.252, dst 10.1.1.251
ICMP: echo reply sent, src 10.1.2.252, dst 10.1.1.251
ICMP: echo reply sent, src 10.1.2.252, dst 10.1.1.251
ICMP: echo reply sent, src 10.1.2.252, dst 10.1.1.251
```

First, you should examine the static routes in Example 6-5. On Albuquerque, one route defines a route to 10.1.2.0, off Yosemite, so the next-hop IP address is configured as 10.1.128.252, which is Yosemite's Serial 0 IP address. Similarly, a route to 10.1.3.0, the subnet off Seville, points to Seville's Serial 0 IP address, 10.1.130.253. Presumably, Albuquerque can forward packets to these subnets now; as seen in Example 6-6's first **ping** command, the ping works.

However, if you look at just the syntax but not the rest of these two examples, you will miss an important concept about routing and some necessary details about the **ping** command. First, two subtleties of the **ping** command are used in these two example console dialogs of Examples 6-6 and 6-7:

- Cisco **ping** commands use the output interface's IP address as the source address of the packet, unless otherwise specified in an extended ping. The first ping in Example 6-6 uses a source of 10.1.128.251; the extended ping uses the source address that the user typed in (10.1.1.251).
- ICMP Echo Reply messages (ping responses) reverse the IP addresses used in the ICMP Echo Request to which they are responding.

To make the ping options appear more obvious, this configuration does not contain routes on Yosemite or Seville, pointing back to the subnet off Albuquerque—namely, 10.1.1.0. In a real network, routing protocols would be used instead. If static routes were used, you would need routes pointing in both directions. But because the needed static route on Yosemite, pointing back to 10.1.1.0, is missing, packets from subnet 10.1.1.0 can get to 10.1.2.0 but cannot get back.

When you troubleshoot this network, you can use the extended **ping** command to act like you issued a ping from a computer on that subnet, without having to call a user and ask him to type a **ping** command for you on his PC. The extended version of the **ping** command can be used to more fully refine the underlying cause of the problem. In fact, when a ping from a router works but a ping from a host does not, the extended ping could help in re-creating the problem without needing to work with the end user on the phone. For example, the extended **ping** command on Albuquerque sent an Echo Request from 10.1.1.251 (Albuquerque's Ethernet) to 10.1.2.252 (Yosemite's Ethernet); no response was received by Albuquerque. Normally, the echoes are sourced from the IP address of the outgoing interface; with the use of the extended **ping** source address option, the source IP address of the echo packet can be changed. Because the ICMP echo generated by the extended ping is sourced from an address in 10.1.1.0, it looks more like a packet from an end user in that subnet. It appears that the ICMP Echo Requests generated by the extended ping were received by Yosemite because the debug messages on Yosemite imply that it sent ICMP Echo Replies back to 10.1.1.251. Somewhere between Yosemite creating the ICMP echo replies and Albuquerque receiving them, a problem occurred.

The problem, as mentioned earlier, is that Yosemite has no routes that tell it how to forward packets back to 10.1.1.0. An examination of the steps after the Echo Replies were created by Yosemite is needed to understand the problem in this example. ICMP asks the IP software in Yosemite to deliver the packets. The IP code performs IP routing table lookup to find the correct route for these packets, whose destination is 10.1.1.251. However, the **show ip route** command output in Example 6-7 shows that Yosemite has no route to subnet 10.1.1.0. It seems that Yosemite created the Echo Reply messages but failed to send them because it has no route to 10.1.1.0/24. This is just one example in which the route in one direction is working fine, but the route in the reverse direction is not.

Other options for extended ping are also quite useful. The Don't Fragment (DF) bit can be set, along with the amount of data to send in the echo so that the MTU for the entire route can be discovered through experimentation. Echo packets that are too large to pass over a link because MTU restrictions will be discarded because the DF bit is set. The timeout value can be set so that the **ping** command will wait longer than the default two seconds for an Echo Reply. Furthermore, not only can a single size for the ICMP Echo be set, but a range of sizes can be used to give a more realistic set of packets.

One key to troubleshooting with the **ping** command is understanding the various codes the command uses to signify the various responses that it can receive. Table 6-50 lists the various codes that the Cisco IOS Software **ping** command can supply.

Table 6-50 *Explanation of the Codes That the* **ping** *Command Receives in Response to Its ICMP Echo Request*

ping Command Code	Explanation
!	ICMP Echo Reply received
.	Nothing received

Table 6-50 *Explanation of the Codes That the* **ping** *Command Receives in Response to Its ICMP Echo Request (Continued)*

ping Command Code	Explanation
U	ICMP unreachable (destination) received
N	ICMP unreachable (network) received
P	ICMP unreachable (port) received
Q	ICMP source quench received
M	ICMP Can't Fragment message received
?	Unknown packet received

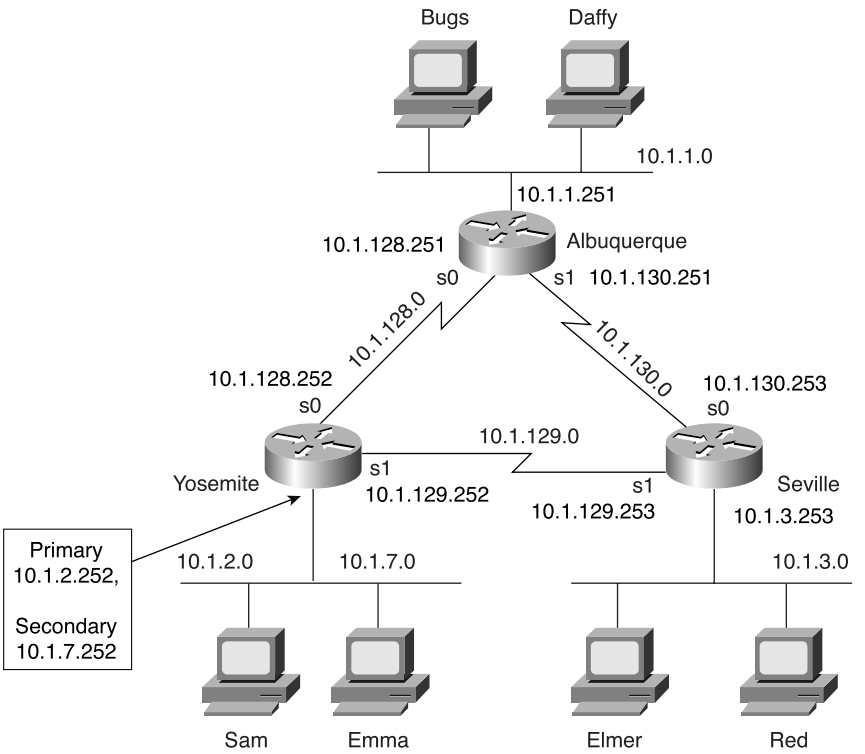
Using Secondary Addresses

As a CCNA, Cisco expects you to be comfortable and familiar with IP address planning issues. One such issue involves what to do when there are no more unassigned IP addresses in a subnet. One alternative solution is to change the mask used on that subnet, making the existing subnet larger. However, changing the mask could cause an overlap. For example, if 10.1.4.0/24 is running out of addresses and you make a change to mask 255.255.254.0 (9 host bits, 23 network/subnet bits), an overlap can occur. 10.1.4.0/23 includes addresses 10.1.4.0 to 10.1.5.255; this is indeed an overlap with a different existing subnet, 10.1.5.0/24. If subnet 10.1.5.0/24 already exists, using 10.1.4.0/23 would not work.

Another alternative for continued growth is to place all the existing addresses in the mostly full subnet into another larger subnet. There must be a valid subnet number that is unassigned, that does not create an overlap, and that is larger than the old subnet. However, this solution causes administrative effort to change the IP addresses. In either case, both solutions that do not use secondary addressing imply a strategy of using different masks in different parts of the network. Use of these different masks is called variable-length subnet masking (VLSM), which brings up another set of complex routing protocol issues.

The issue of running out of addresses in this subnet can be solved by the use of IP secondary addressing. Secondary addressing uses multiple subnets on the same data link. Secondary IP addressing is simple in concept. Because more than one subnet is used on the same medium, the router needs to have more than one IP address on the interface attached to that medium. For example, Figure 6-35 has subnet 10.1.2.0/24; assume that the subnet has all IP addresses assigned. Assuming secondary addressing to be the chosen solution, subnet 10.1.7.0/24 also could be used on the same Ethernet. Example 6-8 shows the configuration for secondary IP addressing on Yosemite.

Figure 6-35 *TCP/IP Network with Secondary Addresses*



Example 6-8 *Secondary IP Addressing Configuration and* **show ip route** *Command on Yosemite*

```
! Excerpt from show running-config follows...
Hostname Yosemite
ip domain-lookup
ip name-server 10.1.1.100 10.1.2.100
interface ethernet 0
ip address 10.1.7.252  255.255.255.0 secondary
ip address 10.1.2.252  255.255.255.0
interface serial 0
ip address 10.1.128.252  255.255.255.0
interface serial 1
ip address 10.1.129.252  255.255.255.0

Yosemite#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

Example 6-8 *Secondary IP Addressing Configuration and* **show ip route** *Command on Yosemite (Continued)*

```
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 4 subnets
C       10.1.2.0 is directly connected, Ethernet0
C       10.1.7.0 is directly connected, Ethernet0
C       10.1.129.0 is directly connected, Serial1
C       10.1.128.0 is directly connected, Serial0
Yosemite#
```

The router has routes to subnets 10.1.2.0/24 and 10.1.7.0/24, so it can forward packets to each subnet. The router also can receive packets from hosts in one subnet and can forward the packets to the other subnet using the same interface.

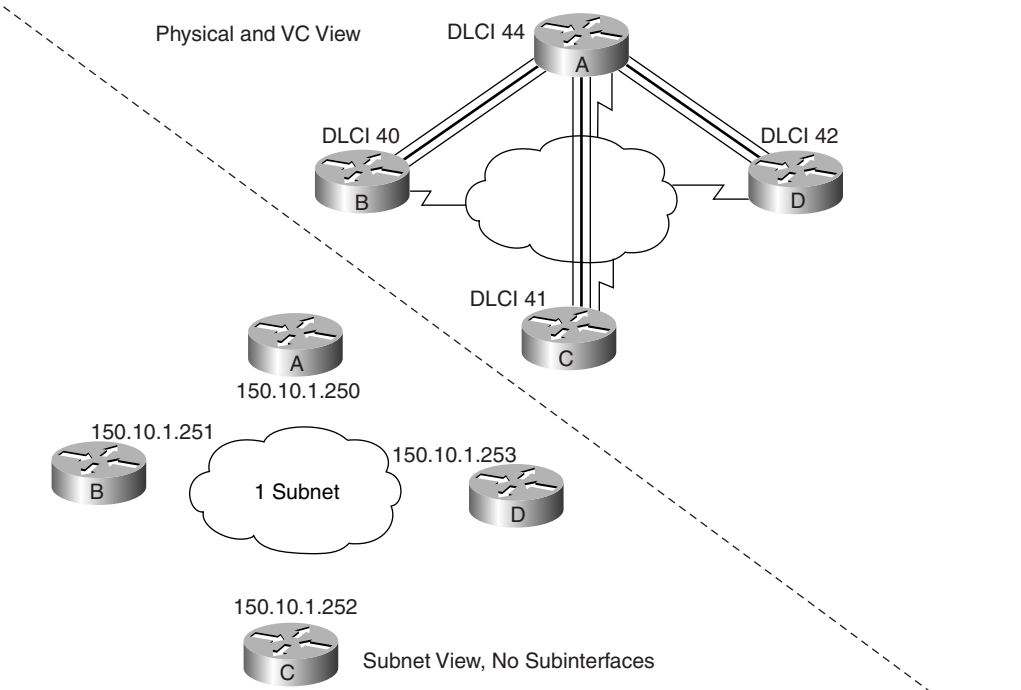
IP Addressing with Frame Relay Subinterfaces

Frame Relay behaves like a WAN in some ways and more like a LAN in other ways. To overcome some routing protocol issues that will be discussed in Chapter 7, "Routing and Routing Protocols," and Chapter 10, "Frame Relay Concepts and Configuration," Cisco provides three different ways to configure IP addresses on Frame Relay serial interfaces:

1. Configure the IP addresses on the normal physical interface, just like for other interfaces. By doing so, all routers on the Frame Relay network are in the same subnet.
2. Configure IP addresses on a point-to-point subinterface of the physical interface. A subinterface is a logical subdivision of the physical interface, which, in this case, allows you to correlate a single VC to the point-to-point subinterface. The IP address is configured under the subinterface. Every VC results in a separate subnet.
3. Configure IP addresses on a multipoint subinterface of the physical interface. A subinterface is a logical subdivision of the physical interface, which, in this case, allows you to correlate multiple VCs to the multipoint subinterface. (The addressing subtleties in this case are better left until Chapter 10.)

Figures 6-36 and 6-37 give a graphical representation of the first two options for the same network. Examples 6-9, 6-10, 6-11, and 6-12 show the configurations on routers A, B, C, and D when point-to-point subinterfaces are used, respectively.

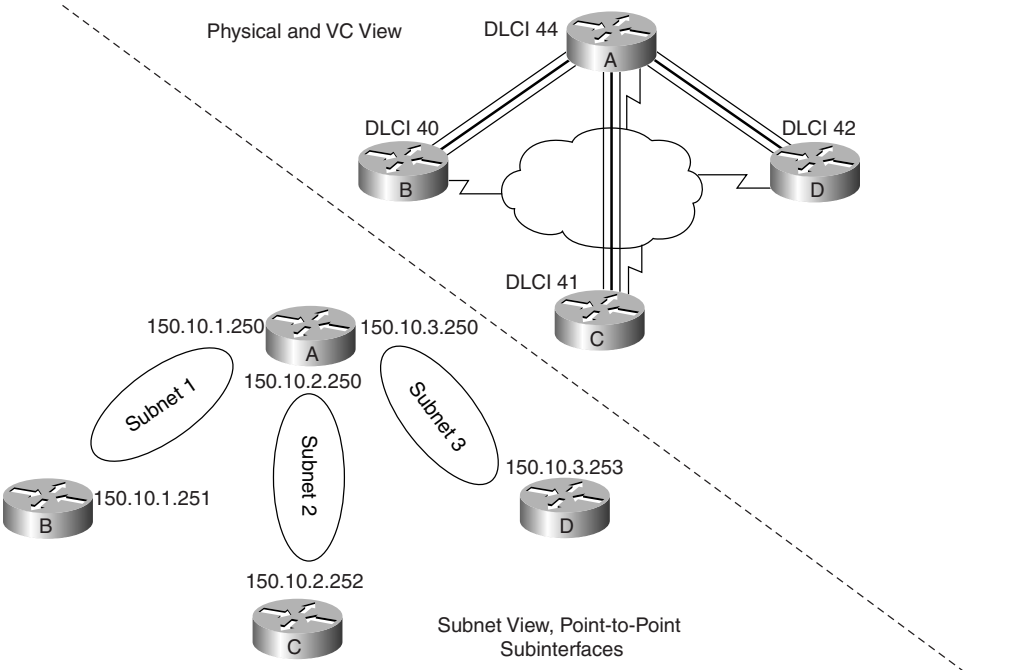
Figure 6-36 *Frame Relay Subnets with No Subinterfaces*



Example 6-9 *Router A Configuration*

```
hostname routerA
interface serial 0
encapsulation frame-relay
!
interface serial 0.1 point-to-point
ip address 150.10.1.250   255.255.255.0
frame-relay interface-dlci 40
description this is for the VC to site B
!
interface serial 0.2 point-to-point
ip address 150.10.2.250   255.255.255.0
frame-relay interface-dlci 41
description this is for the VC to site C
!
interface serial 0.3 point-to-point
ip address 150.10.3.250   255.255.255.0
frame-relay interface-dlci 42
description this is for the VC's to sites D
```

Figure 6-37 *Frame Relay Subnets with Point-to-Point Subinterfaces*



Example 6-10 *Router B Configuration*

```
hostname routerB
!
interface serial 0
encapsulation frame-relay
!
interface serial 0.1 point-to-point
ip address 150.10.1.251   255.255.255.0
frame-relay interface-dlci 44
description this is for the VC to site A
```

Example 6-11 *Router C Configuration*

```
hostname routerC
!
interface serial 0
encapsulation frame-relay
!
```

continues

Example 6-11 *Router C Configuration (Continued)*

```
interface serial 0.2 point-to-point
ip address 150.10.2.252   255.255.255.0
frame-relay interface-dlci 44
description this is for the VC to site A
```

Example 6-12 *Router D Configuration*

```
hostname routerD
!
interface serial 0
encapsulation frame-relay
!
interface serial 0.3 point-to-point
ip address 150.10.3.253   255.255.255.0
frame-relay interface-dlci 44
description this is for the VC to site A
```

For a more complete review of the concepts behind IP addressing over Frame Relay, refer to Chapter 10.

MTU and Fragmentation

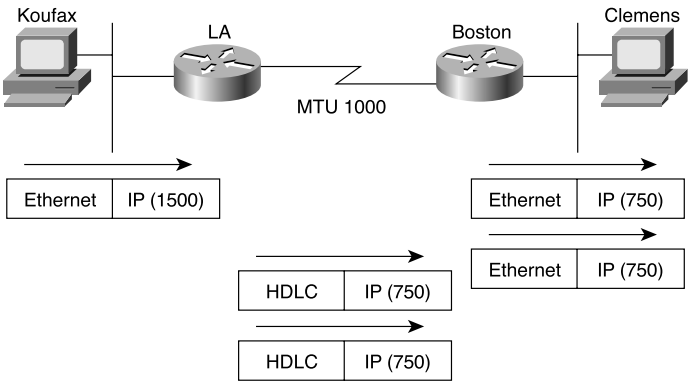
The maximum transmission unit (MTU) is a concept that implies the largest Layer 3 packet that can be forwarded out an interface. The maximum MTU value allowed is based on the data-link protocol; essentially, the maximum size of the data portion of the data-link frame (where the packet is placed) is the maximum setting for the MTU on an interface. The default MTU value on Ethernet and serial interfaces is 1,500.

If an interface's MTU is smaller than a packet that must be forwarded, fragmentation is performed by the router. *Fragmentation* is the process of simply breaking the packet into smaller packets, each of which is less than or equal to the MTU value. For example, consider Figure 6-38, with a point-to-point serial link whose MTU has been lowered to 1000.

As Figure 6-38 illustrates, Koufax threw a 1500-byte packet toward Router LA. LA removed the Ethernet header but could not forward the packet because it was 1500 bytes and the HDLC link supported only an MTU of 1000. LA fragmented the original packet into two packets. After forwarding the two packets, Boston receives the packets and forwards them, without reassembling them—reassembly is done by the endpoint host, which, in this case, is Clemens.

The IP header contains fields useful for reassembly of the fragments into the original packet. The IP header includes an ID value that is the same in each fragmented packet, as well as an offset value that defines which part of the original packet is held in each fragment. Fragmented packets arriving out of order can be identified as part of the same original packet and can be reassembled into the correct order using the offset field in each fragment.

Figure 6-38 *IP Fragmentation*



Two configuration commands can be used to change the IP MTU size on an interface: the **mtu** interface subcommand and the **ip mtu** interface subcommand. The **mtu** command sets the MTU for all Layer 3 protocols; unless there is a need to vary the setting per Layer 3 protocol, this command is preferred. If a different setting is desired for IP, the **ip mtu** command sets the value used for IP.

A few nuances relate to the two MTU-setting commands. If both are configured on an interface, the IP MTU setting takes precedence on the interface. However, if the **mtu** command is configured after the **ip mtu** is configured, the **ip mtu** value is reset to the same value as that of the **mtu** command. Care must be taken when changing these values.

IP Naming Commands and Telnet

When using the IOS CLI, you will want to refer to names instead of IP addresses. Particularly for the **trace**, **ping**, and **telnet** commands, the IP address or host name must be supplied. This section describes the use of host names on an IOS-based device. Along the way, some nuances of the use of Telnet are covered.

The IOS can use statically configured names as well as refer to one or more DNSs. Example 6-13 shows some names statically configured, with configuration pointing to two different DNSs.

Example 6-13 *IP Naming Configuration and* **show ip host** *Command*

```
hostname Cooperstown
!
ip host Mays 10.1.1.1
ip host Aaron 10.2.2.2
ip host Mantle 10.3.3.3
!
ip domain-name lacidar.com
ip name-server 10.1.1.200  10.2.2.200
```

continues

Example 6-13 *IP Naming Configuration and* **show ip host** *Command (Continued)*

```
ip domain-lookup

Seville#show hosts
Default domain is lacidar.com
Name/address lookup uses static mappings

Host                      Flags      Age Type   Address(es)
Mays                      (perm, OK)  0   IP    10.1.1.1
Aaron                     (perm, OK)  0   IP    10.2.2.2
Mantle                    (perm, OK)  0   IP    10.3.3.3
Seville#
```

Router Cooperstown will use any of the three statically configured host name–to–IP address mappings. Three names are statically configured in this case—Mays, Aaron, and Mantle. Any command referring to Mays, Aaron, or Mantle will resolve into the IP addresses shown in the **ip host** command.

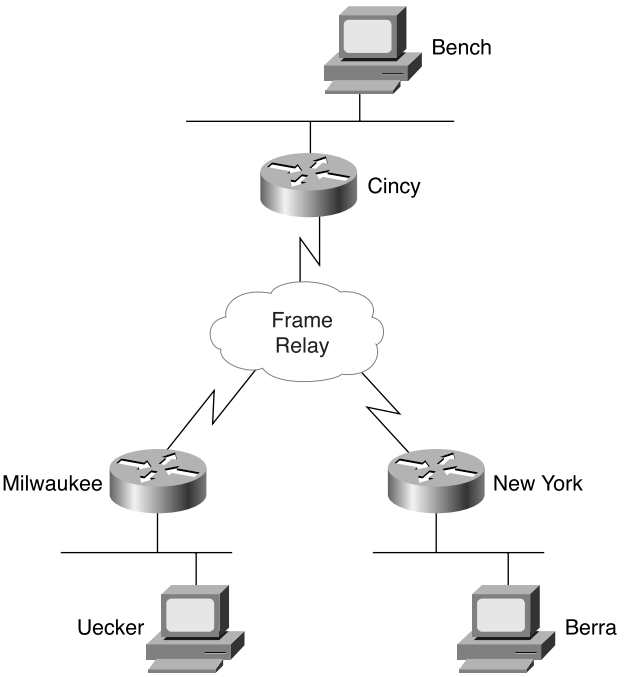
Router Cooperstown also will ask a DNS for name resolution if it does not know the name and IP address already. The DNS configuration is shown toward the end of the configuration. The IP addresses of the name servers are shown in the **ip name-server** command. Up to six DNSs can be listed; they are searched for each request sequentially based on the order in the command. Finally, the **ip domain-lookup** command enables IOS to ask a name server. IP domain lookup is the default; **no ip domain-lookup** disables the DNS client function. For names that do not include the full domain name, the **ip domain-name** command defines the domain name that should be assumed by the router.

The **show ip host** command lists the static entries, in addition to any entries learned from a DNS request. Only the three static entries were in the table, in this case. The term *perm* in the output implies that the entry is static.

Telnet and Suspend

The **telnet** IOS exec command allows you to Telnet from one Cisco device to another; in practical use, it is typically to another Cisco device. One of the most important features of the **telnet** command is the *suspend* feature. To understand the suspend function, you will need to refer to the network diagram in Figure 6-39.

Figure 6-39 *Telnet Suspension*



In the figure, the router administrator is using Bench to Telnet into the Cincy router. When in Cincy, the user Telnets to Milwaukee. When in Milwaukee, the user suspends the Telnet by pressing Ctrl-Shift-6, followed by pressing the letter *x*. The user then Telnets to New York and again suspends the connection. The example begins with Bench already logged into Cincy. Example 6-14 shows example output, with annotations to the side.

Example 6-14 *Telnet Suspensions*

```
Cincy#telnet milwaukee              (User issues command to Telnet to Milwaukee)
Trying Milwaukee (10.1.4.252)... Open


User Access Verification

Password:                (User plugs in password, can type commands at Milwaukee)
```

continues

Example 6-14 *Telnet Suspensions (Continued)*

```
Milwaukee>
Milwaukee>
Milwaukee>
                        (Note: User pressed Ctrl-Shift-6 and then x)
Cincy#telnet NewYork             (User back at Cincy because Telnet was suspended)
Trying NewYork (10.1.6.253)... Open
               (User is getting into New York now, based on telnet NewYork command)


User Access Verification

Password:
NewYork>                                  (User can now type commands on New York)
NewYork>
NewYork>
NewYork>
                                    (Note: User pressed Ctrl-Shift-6 and then x)

Cincy#show sessions                (This command lists suspended Telnet sessions)
Conn Host               Address             Byte  Idle Conn Name
   1 milwaukee          10.1.4.252             0     0 milwaukee
*  2 NewYork            10.1.6.253             0     0 NewYork

Cincy#where                          (where does the same thing)
Conn Host               Address             Byte  Idle Conn Name
   1 milwaukee          10.1.4.252             0     0 milwaukee
*  2 NewYork            10.1.6.253             0     0 NewYork

Cincy#resume 1       (Resume connection 1 (see show session) to Milwaukee)
[Resuming connection 1 to milwaukee ... ]

Milwaukee>                              (User can type commands on Milwaukee)
Milwaukee>
Milwaukee>
(Note: User pressed Ctrl-Shift-6 and then x)    (User wants to go back to Cincy)
Cincy#         (WOW! User just pressed Enter and resumes the last Telnet)
 [Resuming connection 1 to milwaukee ... ]

Milwaukee>
Milwaukee>
Milwaukee>
                                  (Note: User pressed Ctrl-Shift-6 and then x)
                               (Tired of Milwaukee again - can't imagine why!)
Cincy#disconnect 1           (No more need to use Milwaukee – Telnet terminated!)
Closing connection to milwaukee [confirm]         (User presses Enter to confirm)
Cincy#
[Resuming connection 2 to NewYork ... ]
                  (Pressing Enter resumes most recently suspended active Telnet)
```

Example 6-14 *Telnet Suspensions (Continued)*

```
NewYork>
NewYork>
NewYork>
                                        (Note: User pressed Ctrl-Shift-6 and then x)
Cincy#disconnect 2                           (Done with New York, terminate Telnet)
Closing connection to NewYork [confirm]           (Just press Enter to confirm)
Cincy#
```

The play-by-play notes in the example explain most of the details. Example 6-14 begins with the Cincy command prompt that would be seen in Bench's Telnet window because the user at Bench Telnetted into Cincy first. After Telnetting to Milwaukee, the Telnet connection was suspended. Then, after Telnetting to New York, that connection was suspended. The two connections can be suspended or resumed easily. The **resume** command can be used to resume either connection; however, the **resume** command requires a connection ID, which is shown in the **show sessions** command. (The **where** command provides the same output.)

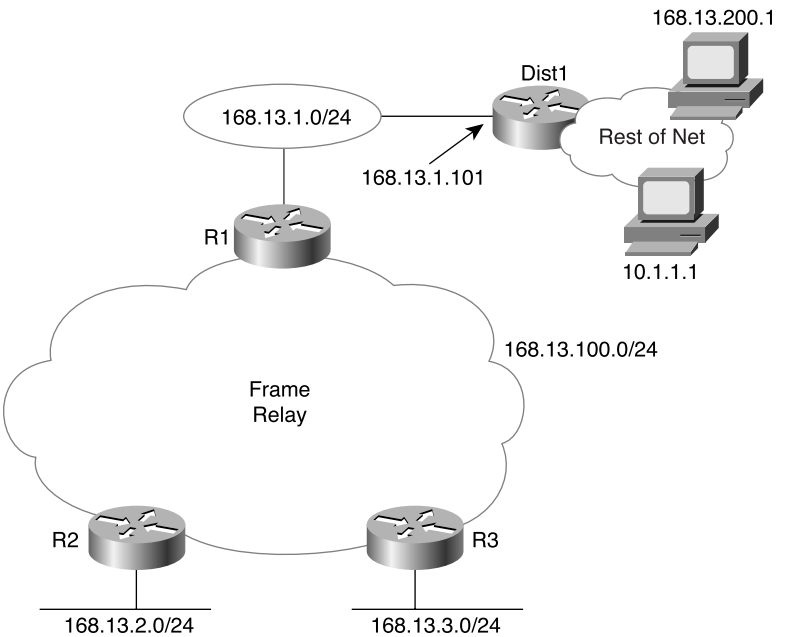
The interesting and potentially dangerous nuance here is that if a Telnet session is suspended and you simply press Enter, *Cisco IOS Software resumes the connection to the most recently suspended Telnet connection*. That is fine, until you realize how much you tend to press the Enter key occasionally to clear some of the clutter from the screen. With a suspended Telnet connection, you also just happened to reconnect to another router. This is particularly dangerous when you are changing the configuration or using potentially damaging exec commands—be careful about what router you are actually using when you have suspended Telnet connections!

Default Routes and the ip classless Command

When a router needs to route a packet and there is no route matching that packet's destination in the routing table, the router discards the packet. Default routing lets the router forward the packet to some default next-hop router. Default routing is that simple! However, two configuration options for default routing make it a little tricky. Also one other option changes the algorithm of how the router decides whether there is a routing table match, which affects when the default route is used.

First, default routes work best when there is one path to a part of the network. In Figure 6-40, R1, R2, and R3 are connected to the rest of this network only through R1's Token Ring interface. All three routers can forward packets to the rest of the network, as long as the packets get to R1, which forwards them to Dist1.

Figure 6-40 *Example Network Using a Default Route*



By coding a default route on R1 that points to router Dist1 in Figure 6-40 and having R1 advertise the default to R2 and R3, default routing can be accomplished. Examples 6-15 and 6-16, along with Figure 6-40, show an example of a default route on R1.

Example 6-15 *R1 Static Default Route Configuration and Routing Table*

```
R1(config)#ip route 0.0.0.0 0.0.0.0 168.13.1.101

R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 168.13.1.101 to network 0.0.0.0

     168.13.0.0/24 is subnetted, 4 subnets
C       168.13.1.0 is directly connected, TokenRing0
R       168.13.3.0 [120/1] via 168.13.100.3, 00:00:05, Serial0.1
R       168.13.2.0 [120/1] via 168.13.100.2, 00:00:21, Serial0.1
C       168.13.100.0 is directly connected, Serial0.1
S*   0.0.0.0/0 [1/0] via 168.13.1.101
R1#
```

Example 6-16 *R3—Nuances with Successful Use of Static Route on R1*

```
R3#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 168.13.100.1 to network 0.0.0.0

     168.13.0.0/24 is subnetted, 4 subnets
R       168.13.1.0 [120/1] via 168.13.100.1, 00:00:13, Serial0.1
C       168.13.3.0 is directly connected, Ethernet0
R       168.13.2.0 [120/1] via 168.13.100.2, 00:00:06, Serial0.1
C       168.13.100.0 is directly connected, Serial0.1
R3#ping 10.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 84/89/114 ms
R3#
R3#ping 168.13.200.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 168.13.200.1, timeout is 2 seconds:
…..
Success rate is 0 percent (0/5)
R3#
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#ip classless
R3(config)#^Z
R3#ping 168.13.200.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 168.13.200.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 80/88/112 ms
R3#
```

The default route is defined with a static **ip route** command on R1, with destination 0.0.0.0, mask 0.0.0.0. This route matches all destinations, by convention. R1 advertises to R2 and R3, as seen in the output of the **show ip route** command on R3 in Example 6-16. So the **ping 10.1.1.1** command on R3 works, just like it should. However, the **ping 168.13.200.1** command does not work—why?

The key to knowing why one ping worked and one did not is based on what Cisco IOS Software thinks is a "match" of the routing table. If the router first matches the Class A, B, or C network

number that a destination resides in and then looks for the specific subnet, the router is considered to be *classful*. If the router simply looks for the best subnet match, ignoring Class A, B, and C rules, the router is *classless*. What you need in this case is a router with no class! Seriously, here's R3's logic when the ping failed:

1. I need to send a packet to 168.13.200.1.
2. I match Class B network 168.13.0.0, so there is a match.
3. I do not match a specific subnet that contains 168.13.200.1.
4. I use the default route only if there is no match, and there was a match, so I discard the packet.

If you make R3 act in a classless manner, the ping will work, as in the second **ping 168.13.200.1** command in Example 6-16. The logic works something like this:

1. I need to send a packet to 168.13.200.1.
2. I do not match a specific subnet that contains 168.13.200.1.
3. I use the default route only if there is no match, and there was no match, so I use the default route.

The **no ip classless** command makes the router behave as a classful router. The **ip classless** command makes it act as a classless router, which would be preferred in this case. It seems like classless would always be better, but it is not. What if the networks on the other side of Dist1 were on the Internet and 168.13.0.0 was your registered Class B network? Well, there should not be any part of 168.13.0.0 to the right of Dist1, so it would be pointless to send packets for unknown subnets of 168.13.0.0 to Dist1 because they will be discarded at some point anyway. There are uses for both modes—just be aware of how each works.

The gateway of last resort, highlighted in the **show ip route** command output, sounds like a pretty desperate feature. There are worse things than having to discard a packet in a router, and "gateway of last resort" simply references the current default route. It is possible that several default routes have been configured and then distributed with a routing protocol; the gateway of last resort is the currently used default on a particular router. Be careful—multiple defaults can cause a routing loop.

Another style of configuration for the default route uses the **ip default-network** command. This command is used most typically when you want to reach other Class A, B, or C networks by default, but all the subnets of your own network are expected to be in your own routing tables. For example, imagine that the cloud next to Dist1 in Figure 6-40 has subnets of network 10.0.0.0 in it as well as other networks. (Dist1 could be an ISP router.) The network in Figure 6-40 is still in use, but instead of using the **ip route 0.0.0.0 0.0.0.0 168.13.1.101** command, the **ip default-network 10.0.0.0** command is used on R1. R1 uses its route to network 10.0.0.0 as its default and advertises this route as a default route to other routers. Examples 6-17 and 6-18 show several details on R1 and R3.

Example 6-17 *R1's Use of the* **ip default-network** *Command*

```
R1#configure terminal
R1(config)#ip default-network 10.0.0.0
R1(config)#exit
R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 168.13.1.101 to network 10.0.0.0

     168.13.0.0/24 is subnetted, 5 subnets
R       168.13.200.0 [120/1] via 168.13.1.101, 00:00:12, TokenRing0
C       168.13.1.0 is directly connected, TokenRing0
R       168.13.3.0 [120/1] via 168.13.100.3, 00:00:00, Serial0.1
R       168.13.2.0 [120/1] via 168.13.100.2, 00:00:00, Serial0.1
C       168.13.100.0 is directly connected, Serial0.1
R*   10.0.0.0/8 [120/1] via 168.13.1.101, 00:00:12, TokenRing0
R1#
```

Example 6-18 *R3 Routing Table and* **trace** *Command Samples*

```
R3#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 168.13.100.1 to network 0.0.0.0

     168.13.0.0/24 is subnetted, 5 subnets
R       168.13.200.0 [120/2] via 168.13.100.1, 00:00:26, Serial0.1
R       168.13.1.0 [120/1] via 168.13.100.1, 00:00:26, Serial0.1
C       168.13.3.0 is directly connected, Ethernet0
R       168.13.2.0 [120/1] via 168.13.100.2, 00:00:18, Serial0.1
C       168.13.100.0 is directly connected, Serial0.1
R    10.0.0.0/8 [120/2] via 168.13.100.1, 00:00:26, Serial0.1
R*   0.0.0.0/0 [120/2] via 168.13.100.1, 00:00:26, Serial0.1
R3#trace 168.13.222.2

Type escape sequence to abort.
Tracing the route to 168.13.222.2

  1 168.13.100.1 68 msec 56 msec 52 msec
  2 168.13.1.101 52 msec 56 msec 52 msec
R3#trace 10.1.1.1
```

continues

Example 6-18 *R3 Routing Table and* **trace** *Command Samples (Continued)*

```
Type escape sequence to abort.
Tracing the route to 10.1.1.1

  1 168.13.100.1 68 msec 56 msec 52 msec
  2 168.13.1.101 48 msec 56 msec 52 msec
R3#
```

Both R1 and R3 have default routes, but they are shown differently in their respective routing tables. R1 shows a route to network 10.0.0.0 with an *, meaning that it is a candidate to be the default route. In R3, 0.0.0.0 shows up in the routing table as the candidate default route. The reason that R3 shows this information differently is that RIP advertises default routes using network number 0.0.0.0. If IGRP or EIGRP were in use, there would be no route to 0.0.0.0 on R3, and network 10.0.0.0 would be the candidate default route. That's because IGRP and EIGRP would flag 10.0.0.0 as a candidate default route in their routing updates rather than advertise the special case of 0.0.0.0.

The default route on R3 is used for destinations in network 168.13.0.0, 10.0.0.0, or any other network because **ip classless** is still configured. The **trace** commands in Example 6-18, which show destinations in two different networks, both succeed. The **trace** commands each show that the first router in the route was R1, then comes Dist1, and then the command finished. If *n* other routers had been present in the network of Figure 6-40, these routers could have shown up in the **trace** output as well. (In each case, the destination address was the address of some loopback interface in Dist1, so there were no routers beyond Dist1.) **ip classless** still was configured; it is recommended that you configure **ip classless** if using any form of default routes.

Cisco Discovery Protocol

The Cisco Discovery Protocol (CDP) discovers basic information about neighboring routers and switches, without needing to know the passwords for the neighboring devices. CDP supports any LAN, HDLC, Frame Relay, and ATM interface. CDP supports any interface that supports the use of SNAP headers. The router or switch can discover Layer 2 and Layer 3 addressing details of neighboring routers without even configuring that Layer 3 protocol—this is because CDP is not dependent on any particular Layer 3 protocol.

CDP discovers several useful details from the neighboring device:

- **Device identifier**—Typically the host name
- **Address list**—Network and data-link addresses
- **Port identifier**—Text that identifies the port, which is another name for an interface

- **Capabilities list**—Information on what type of device it is—for instance, a router or a switch
- **Platform**—The model and OS level running in the device

CDP is enabled in the configuration by default. The **no cdp run** global command disables CDP for the entire device, and the **cdp run** global command re-enables CDP. Likewise, the **no cdp enable** interface subcommand disables CDP just on that interface, and the **cdp enable** command switches back to the default state of CDP being enabled.

A variety of **show cdp** command options are available. Example 6-19 lists the output of the commands, with some commentary following.

Example 6-19 show cdp *Command Options*

```
Seville#show cdp neighbor
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
fred               Ser 1           172          R        2500      Ser 1
Yosemite           Ser 0.2         161          R        2500      Ser 0.2

Seville#show cdp entry fred
-------------------------
Device ID: fred
Entry address(es):
  IP address: 163.5.8.3
Platform: cisco 2500,  Capabilities: Router
Interface: Serial1,  Port ID (outgoing port): Serial1
Holdtime : 168 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-DS-L), Version 12.2(3), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Wed 18-Jul-01 21:10 by pwade

advertisement version: 2

Seville#show cdp neighbor detail
-------------------------
Device ID: fred
Entry address(es):
  IP address: 163.5.8.3
Platform: cisco 2500,  Capabilities: Router
Interface: Serial1,  Port ID (outgoing port): Serial1
Holdtime : 164 sec

Version :
```

continues

Example 6-19 show cdp *Command Options (Continued)*

```
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-DS-L), Version 12.2(3), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Wed 18-Jul-01 21:10 by pwade

advertisement version: 2

-------------------------
Device ID: Yosemite
Entry address(es):
  IP address: 10.1.5.252
  Novell address: 5.0200.bbbb.bbbb
Platform: cisco 2500,  Capabilities: Router
Interface: Serial0.2,  Port ID (outgoing port): Serial0.2
Holdtime : 146 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-DS-L), Version 12.2(3), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Wed 18-Jul-01 21:10 by pwade

advertisement version: 2

Seville#show cdp interface
Ethernet0 is up, line protocol is down
  Encapsulation ARPA
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
Serial0.2 is up, line protocol is up
  Encapsulation FRAME-RELAY
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
Serial1 is up, line protocol is up
  Encapsulation HDLC
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds

Seville#show cdp traffic
CDP counters :
        Total packets output: 31, Input: 41
        Hdr syntax: 0, Chksum error: 0, Encaps failed: 9
        No memory: 0, Invalid packet: 0, Fragmented: 0
        CDP version 1 advertisements output: 0, Input: 0
        CDP version 2 advertisements output: 31, Input: 41
```

The commands provide information about both the neighbors and the behavior of the CDP protocol itself. In the **show cdp entry fred** command in Example 6-19, all the details learned by CDP are shown and highlighted. To know that fred is the device identifier of a neighbor, the **show cdp neighbor** command can be used to summarize the information about each neighbor. **show cdp neighbor detail** lists the detail of all neighbors, in the same format as **show cdp entry**. In addition, **show cdp traffic** lists the overhead that CDP introduces to perform its functions.

Foundation Summary

The "Foundation Summary" is a collection of tables and figures that provides a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final prep before the exam, these tables and figures will hopefully be a convenient way to review the day before the exam.

Figure 6-41 outlines the basic process with DNS name resolution and IP ARP.

Figure 6-41 *Sample Network, DNS and ARP Processes*

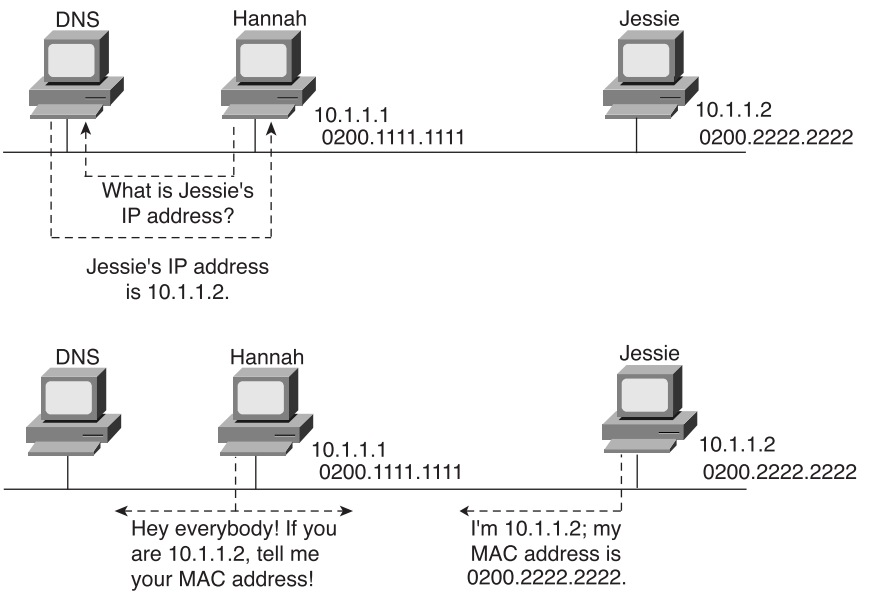


Figure 6-42 shows an example of TCP connection establishment flow.

Figure 6-42 *TCP Connection Establishment*

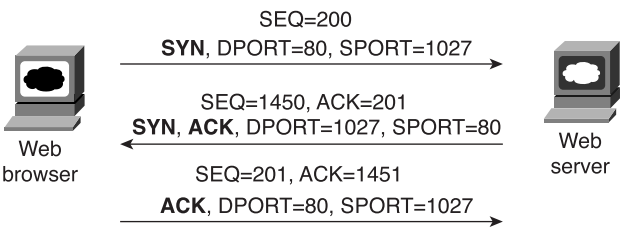


Table 6-51 summarizes TCP functions.

Table 6-51 *TCP Function Summary*

Function	Description
Multiplexing	Function that allows receiving hosts to decide the correct application for which the data is destined, based on the port number
Error recovery (reliability)	Process of numbering and acknowledging data with sequence and acknowledgment header fields
Flow control using windowing	Process that uses window sizes to protect buffer space and routing devices
Connection establishment and termination	Process used to initialize port numbers and sequence and acknowledgement fields
Ordered data transfer	Continuous stream of bytes from upper-layer process, "segmented" for transmission

Table 6-52 contrasts typical transport layer functions as performed (or not performed) by UDP or TCP.

Table 6-52 *TCP and UDP Functional Comparison*

Function	Description (TCP)	Description (UDP)
Data transfer	This involves a continuous stream of ordered data.	This involves message (datagram) delivery.
Multiplexing	Receiving hosts decide the correct application for which the data is destined, based on the port number.	Receiving hosts decide the correct application for which the data is destined, based on the port number.
Reliable transfer	Acknowledgment of data uses the sequence and acknowledgment fields in the TCP header.	This is not a feature of UDP.
Flow control	This process is used to protect buffer space and routing devices.	This is not a feature of UDP.
Connections	This process is used to initialize port numbers and other TCP header fields.	UDP is connectionless.

Table 6-53 summarizes some features of TFTP and FTP.

Table 6-53 *Comparison of FTP and TFTP*

FTP	TFTP
Uses TCP	Uses UDP
Uses robust control commands	Uses simple control commands
Sends data over a separate TCP connection from control commands	Uses no connections because of UDP
Requires more memory and programming effort	Requires less memory and programming effort

Table 6-54 lists the IP terms used in the upcoming sections, giving an exact definition.

Table 6-54 *IP Addressing Terminology*

Term	Definition
IP address	A 32-bit number, usually written in dotted-decimal form, that uniquely identifies an interface of some computer.
Host address	Another term for IP address.
Network	A group of hosts, all of which have an identical beginning portion of their IP addresses.
Network number	A 32-bit number, usually written in dotted-decimal form, that represents a network. This number cannot be assigned as an IP address to an interface of some computer. The host portion of the network number has a value of all binary 0s.
Network address	Another term for network number.
Broadcast address	A 32-bit number, usually written in dotted-decimal form, that is used to address all hosts in the network. This number cannot be assigned as an IP address to an interface of some computer. The host portion of the network number has a value of all binary 1s.
Subnet	A group of hosts, all of which have an identical beginning portion of their IP addresses. A subnet differs from a network in that a subnet is a further subdivision of a network, with a longer portion of the addresses being identical.
Subnet number	A 32-bit number, usually written in dotted-decimal form, that represents a subnet. This number cannot be assigned as an IP address to an interface of some computer. The host portion of the network number has a value of all binary 0s.
Subnet address	Another term for subnet number.

continues

Table 6-54 *IP Addressing Terminology (Continued)*

Term	Definition
Subnet broadcast address	A 32-bit number, usually written in dotted-decimal form, that is used to address all hosts in the subnet. This number cannot be assigned as an IP address to an interface of some computer. The host portion of the network number has a value of all binary 1s.
Subnetting	The process of subdividing networks into smaller subnets. This is jargon—for example, "Are you subnetting your network?"
Network mask	A 32-bit number, usually written in dotted-decimal form. The mask is used by computers to calculate the network number of a given IP address by performing a Boolean AND of the address and mask. The mask also defines the number of host bits in an address.
Mask	A generic term for a mask, whether it is a default mask or a subnet mask.
Address mask	Another term for a mask.
Default Class A mask	The mask used for Class A networks when no subnetting is used. The value is 255.0.0.0.
Default Class B mask	The mask used for Class B networks when no subnetting is used. The value is 255.255.0.0.
Default Class C mask	The mask used for Class C networks when no subnetting is used. The value is 255.255.255.0.
Subnet mask	A nondefault mask that is used when subnetting.
Network part or network field	Term used to describe the first part of an IP address. The network part is 8, 16, or 24 bits for Class A, B, and C networks, respectively.
Host part or host field	Term used to describe the last part of an IP address. The host part is 24, 16, or 8 bits for Class A, B, and C networks, respectively, when subnetting is not used. When subnetting, the size of the host part depends on the subnet mask chosen for that network.
Subnet part of subnet field	Term used to describe the middle part of an IP address. The subnet part is variable in size, based on how subnetting is implemented.

Table 6-55 summarizes the possible network numbers, the total number of each type, and the number of hosts in each Class A, B, and C network.

Table 6-55 *List of All Possible Valid Network Numbers*

Class	First Octet Range	Valid Network Numbers	Total Number of This Class of Network	Number of Hosts per Network
A	1 to 126	1.0.0.0 to 126.0.0.0	2^7, minus two special cases	2^{24}, minus two special cases

Table 6-55 *List of All Possible Valid Network Numbers (Continued)*

Class	First Octet Range	Valid Network Numbers	Total Number of This Class of Network	Number of Hosts per Network
B	128 to 191	128.1.0.0 to 191.254.0.0	2^{14}, minus two special cases	2^{16}, minus two special cases
C	192 to 223	192.0.1.0 to 223.255.254.0	2^{21}, minus two special cases	2^{8}, minus two special cases

Figure 6-43 shows the format of addresses when subnetting.

Figure 6-43 *Address Formats When Subnetting Is Used*

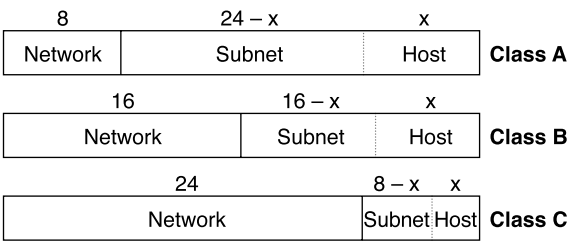


Table 6-56 lists the key rules to deconstructing an IP address, alongside an example.

Table 6-56 *First Example, with Rules for Learning Network, Subnet, and Host Part Sizes*

Step	Example	Rules to Remember
Address	8.1.4.5	
Mask	255.255.0.0	
Number of network bits	8	Always defined by Class A, B, C
Number of host bits	16	Always defined as number of binary 0s in mask
Number of subnet bits	8	32 – (network size + host size)

Table 6-57 lists the only valid decimal values in a mask and their binary equivalents.

Table 6-57 *Decimal and Binary Values in a Single Octet of a Valid Subnet Mask*

Decimal	Binary
0	0000 0000
128	1000 0000

continues

Table 6-57 *Decimal and Binary Values in a Single Octet of a Valid Subnet Mask (Continued)*

Decimal	Binary
192	1100 0000
224	1110 0000
240	1111 0000
248	1111 1000
252	1111 1100
254	1111 1110
255	1111 1111

Table 6-58 is an example subnet chart, used to help learn the easy subnetting process using no binary math.

Table 6-58 *Subnet Chart—Generic*

	Octet 1	Octet 2	Octet 3	Octet 4
Address				
Mask				
Subnet number				
First valid address				
Broadcast				
Last valid address				

The four-step process for dissecting IP addresses is summarized in the following list:

Step 1 Identify the structure of the IP address.

- **A** Identify the size of the network part of the address, based on Class A, B, and C rules.
- **B** Identify the size of the host part of the address, based on the number of binary 0s in the mask. If the mask is tricky, use the chart of typical mask values to convert the mask to binary more quickly.
- **C** The size of the subnet part is what's "left over"; mathematically, it is 32 – (number of network + host bits).
- **D** Declare the number of subnets, which is $2^{\text{number-of-subnet-bits}} - 2$.
- **E** Declare the number of hosts per subnet, which is $2^{\text{number-of-host-bits}} - 2$.

Step 2 Create the chart that will be used in Steps 3 and 4.

- **A** Create a generic subnet chart.
- **B** Write down the IP address and subnet mask in the first two rows of the chart.
- **C** If an easy mask is used, draw a vertical line between the 255s and the 0s in the mask, from top to bottom of the chart. If a hard mask is used, draw a box around the column of the interesting octet.
- **D** Copy the address octets to the left of the line or the box into the final four rows of the chart.

Step 3 Derive the subnet number and the first valid IP address.

- **A** Write down 0s in the subnet octets to the right of the line or the box.
- **B** If the mask is hard and there is a box in the chart, use the magic number trick to find the value of the subnet's interesting octet.
- **C** To derive the first valid IP address, copy the first three octets of the subnet number, and add 1 to the fourth octet of the subnet number.

Step 4 Derive the broadcast address and the last valid IP address.

- **A** Write down 255s in the broadcast address octets to the right of the line or the box.
- **B** If the mask is hard and there is a box in the chart, use the magic number trick to find the value of the broadcast address's interesting octet.
- **C** To derive the last valid IP address, copy the first three octets of the broadcast address, and subtract 1 from the fourth octet of the broadcast address.

The three-step process to find all the subnet numbers of a network is listed next:

Step 1 Write down the network number and subnet mask in the first two rows of the subnet list chart.

Step 2 Write down the network number in the third row. This is the zero subnet, which is one of the two reserved subnets.

Step 3 Do the following two tasks, stopping when the next number that you would write down in the interesting column is 256. (Don't write that one down—it's not valid.)

A Copy all three noninteresting octets from the previous line.

B Add the magic number to the previous interesting octet, and write that down as the value of the interesting octet.

Tables 6-59 and 6-60 summarize many of the most common commands used for IP configuration and verification.

Table 6-59 *IP Configuration Commands*

Command	Configuration Mode
ip address *ip-address mask* [**secondary**]	Interface mode
ip host *name* [*tcp-port-number*] *address1* [*address2...address8*]	Global
ip route *network-number network-mask* {*ip-address* | *interface*} [*distance*] [**name** *name*]	Global
ip name-server *server-address1* [[*server- address2*] *…server-address6*]	Global
ip domain-lookup	Global
ip routing	Global
ip netmask-format {**bitcount** | **decimal** | **hexadecimal**}	Interface mode
ip default-network *network*	Global
ip classless	Global
ip host ***name*** [*tcp-port-number*] *address1* [*address2…address8*]	Global

Table 6-60 *IP Exec Commands*

Command	Function
show hosts	Lists all host names and corresponding IP addresses
show interfaces [*type number*]	Lists interface statistics, including IP address
show ip interface [*type number*]	Provides a detailed view of IP parameter settings, per interface
show ip interface brief	Provides a summary of all interfaces and their IP addresses

Table 6-60 *IP Exec Commands (Continued)*

Command	Function
show ip route [*ip-address* [*mask*] [**longer-prefixes**]]	[*protocol* [*process-id*]]	Shows entire routing table or a subset if other parameters are entered
show ip arp [*ip-address*] [*host-name*] [*mac-address*] [*type number*]	Displays IP ARP cache
debug ip packet	Issues log messages for each IP packet
terminal ip netmask-format {**bitcount** | **decimal** | **hexadecimal**}	Sets type of display for subnet masks in **show** commands
ping [*protocol* | **tag**] {*host-name* | *system-address*}	Sends and receives ICMP echo messages to verify connectivity
trace [*protocol*] [*destination*]	Sends series of UDP packets with increasing TTL values to verify the current route to a host

Q&A

As mentioned in Chapter 1, "All About the Cisco Certified Network Associate Certification," the questions and scenarios in this book are more difficult than what you should experience on the actual exam. The questions do not attempt to cover more breadth or depth than the exam; however, they are designed to make sure that you know the answer. Rather than allowing you to derive the answer from clues hidden inside the question itself, the questions challenge your understanding and recall of the subject. Questions from the "Do I Know This Already?" quiz from the beginning of the chapter are repeated here to ensure that you have mastered the chapter's topic areas. Hopefully, these questions will help limit the number of exam questions on which you narrow your choices to two options and then guess.

The answers to these questions can be found in Appendix A.

1 What do *TCP*, *UDP*, *IP*, and *ICMP* stand for? Which protocol is considered to be Layer 3–equivalent when comparing TCP/IP to the OSI protocols?

_____

_____

_____

2 Name the parts of an IP address.

_____

_____

_____

3 Define the term *subnet mask*. What do the bits in the mask whose values are binary 0 tell you about the corresponding IP address(es)?

_____

_____

_____

4 Given the IP address 134.141.7.11 and the mask 255.255.255.0, what is the subnet number?

_____

_____

_____

5 Given the IP address 193.193.7.7 and the mask 255.255.255.0, what is the subnet number?

6 Given the IP address 10.5.118.3 and the mask 255.255.0.0, what is the subnet number?

7 Given the IP address 190.1.42.3 and the mask 255.255.255.0, what is the subnet number?

8 Given the IP address 200.1.1.130 and the mask 255.255.255.224, what is the subnet number?

9 Given the IP address 220.8.7.100 and the mask 255.255.255.240, what is the subnet number?

10 Given the IP address 140.1.1.1 and the mask 255.255.255.248, what is the subnet number?

11 Given the IP address 167.88.99.66 and the mask 255.255.255.192, what is the subnet number?

_____

_____

_____

12 Given the IP address 134.141.7.11 and the mask 255.255.255.0, what is the subnet broadcast address?

_____

_____

_____

13 Given the IP address 193.193.7.7 and the mask 255.255.255.0, what is the broadcast address?

_____

_____

_____

14 Given the IP address 10.5.118.3 and the mask 255.255.0.0, what is the broadcast address?

_____

_____

_____

15 Given the IP address 190.1.42.3 and the mask 255.255.255.0, what is the broadcast address?

_____

_____

_____

16 Given the IP address 200.1.1.130 and the mask 255.255.255.224, what is the broadcast address?

_____

_____

_____

17 Given the IP address 220.8.7.100 and the mask 255.255.255.240, what is the broadcast address?

18 Given the IP address 140.1.1.1 and the mask 255.255.255.248, what is the broadcast address?

19 Given the IP address 167.88.99.66 and the mask 255.255.255.192, what is the broadcast address?

20 Given the IP address 134.141.7.11 and the mask 255.255.255.0, what are the assignable IP addresses in this subnet?

21 Given the IP address 193.193.7.7 and the mask 255.255.255.0, what are the assignable IP addresses in this subnet?

22 Given the IP address 10.5.118.3 and the mask 255.255.0.0, what are the assignable IP addresses in this subnet?

_____

_____

_____

23 Given the IP address 190.1.42.3 and the mask 255.255.255.0, what are the assignable IP addresses in this subnet?

_____

_____

_____

24 Given the IP address 200.1.1.130 and the mask 255.255.255.224, what are the assignable IP addresses in this subnet?

_____

_____

_____

25 Given the IP address 220.8.7.100 and the mask 255.255.255.240, what are the assignable IP addresses in this subnet?

_____

_____

_____

26 Given the IP address 140.1.1.1 and the mask 255.255.255.248, what are the assignable IP addresses in this subnet?

_____

_____

_____

27 Given the IP address 167.88.99.66 and the mask 255.255.255.192, what are the assignable IP addresses in this subnet?

_____

_____

_____

28 Given the IP address 134.141.7.7 and the mask 255.255.255.0, what are all the subnet numbers if the same (static) mask is used for all subnets in this network?

29 Given the IP address 10.5.118.3 and the mask 255.255.255.0, what are all the subnet numbers if the same (static) mask is used for all subnets in this network?

30 Given the IP address 220.8.7.100 and the mask 255.255.255.240, what are all the subnet numbers if the same (static) mask is used for all subnets in this network?

31 Given the IP address 220.8.7.1 and the mask 255.255.255.240, what are all the subnet numbers if the same (static) mask is used for all subnets in this network?

32 How many IP addresses could be assigned in each subnet of 134.141.0.0, assuming that a mask of 255.255.255.0 is used? If the same (static) mask is used for all subnets, how many subnets are there?

33 How many IP addresses could be assigned in each subnet of 10.0.0.0, assuming that a mask of 255.255.255.0 is used? If the same (static) mask is used for all subnets, how many subnets are there?

34 How many IP addresses could be assigned in each subnet of 220.8.7.0, assuming that a mask of 255.255.255.240 is used? If the same (static) mask is used for all subnets, how many subnets are there?

_____

_____

_____

35 How many IP addresses could be assigned in each subnet of 140.1.0.0, assuming that a mask of 255.255.255.248 is used? If the same (static) mask is used for all subnets, how many subnets are there?

_____

_____

_____

36 You design a network for a customer, and the customer insists that you use the same subnet mask on every subnet. The customer will use network 10.0.0.0 and needs 200 subnets, each with 200 hosts maximum. What subnet mask would you use to allow the largest amount of growth in subnets? Which mask would work and would allow for the most growth in the number of hoses per subnet?

_____

_____

_____

37 Create a minimal configuration enabling IP on each interface on a 2501 router (two serial, one Ethernet). The NIC assigned you network 8.0.0.0. Your boss says that you need, at most, 200 hosts per subnet. You decide against using VLSM. Your boss also says to plan your subnets so that you can have as many subnets as possible rather than allow for larger subnets later. When choosing the actual IP address values and subnet numbers, you decide to start with the lowest numerical values. Assume that point-to-point serial links will be attached to this router and that RIP is the routing protocol.

_____

_____

_____

38 In the previous question, what would be the IP subnet of the link attached to serial 0? If another user wanted to answer the same question but did not have the enable password, what command(s) might provide this router's addresses and subnets?

39 Describe the question and possible responses in setup mode when a router wants to know the mask used on an interface. How can the router derive the correct mask from the information supplied by the user?

40 Name the three classes of unicast IP addresses and list their default masks, respectively. How many of each type could be assigned to companies and organizations by the NIC?

41 Describe how TCP performs error recovery. What role do the routers play?

42 Define the purpose of an ICMP redirect message.

43 Define the purpose of the **trace** command. What type of messages does it send, and what type of ICMP messages does it receive?

_____

_____

_____

44 What does *IP* stand for? What does *ICMP* stand for? Which protocol is considered to be a Layer 3 protocol when comparing TCP/IP to the OSI protocols?

_____

_____

_____

45 What causes the output from an IOS **ping** command to display "UUUUU?"

_____

_____

_____

46 Describe how to view the IP ARP cache in a Cisco router. Also describe the three key elements of each entry.

_____

_____

_____

47 How many hosts are allowed per subnet if the subnet mask used is 255.255.255.192? How many hosts are allowed for 255.255.255.252?

_____

_____

_____

48 How many subnets could be created if using static-length masks in a Class B network when the mask is 255.255.255.224? What about when the mask is 255.255.252.0?

_____

_____

_____

49 Name the two commands typically used to create a default gateway for a router.

50 Assume that subnets of network 10.0.0.0 are in the IP routing table in a router but that no other network and subnets are known, except that there is also a default route (0.0.0.0) in the routing table. A packet destined for 192.1.1.1 arrives at the router. What configuration command determines whether the default route will be used in this case?

51 Assume that subnets of network 10.0.0.0 are in the IP routing table in a router but that no other network and their subnets are known, except that there is also a default route (0.0.0.0) in the routing table. A packet destined for 10.1.1.1 arrives at the router, but there is no known subnet of network 10 that matches this destination address. What configuration command determines whether the default route will be used in this case?

52 What does the acronym *CIDR* stand for? What is the original purpose of CIDR?

53 Define the term *private addressing* as defined in RFC 1918.

54 Define the acronym *NAT* and the basics of its operation.

_____

_____

_____

55 Which requires more lines of source code, FTP or TFTP? Justify your answer.

_____

_____

_____

56 Does FTP or TFTP perform error recovery? If so, describe the basics of how they perform error recovery.

_____

_____

_____

57 Describe the process used by IP routers to perform fragmentation and reassembly of packets.

_____

_____

_____

58 How many TCP segments are exchanged to establish a TCP connection? How many are required to terminate a TCP connection?

_____

_____

_____

59 How many Class B–style networks are reserved by RFC 1918 private addressing?

_____

_____

_____

Scenarios

Scenario 6-1: IP Addressing and Subnet Calculation

Assume that you just took a new job. No one trusts you yet, so they will not give you any passwords to the router. Your mentor at your new company has left you at his desk while he goes to a meeting. He has left up a Telnet window, logged in to one router in user mode. In other words, you can issue only user-mode commands.

Assuming that you had issued the following commands (see Example 6-20), draw the most specific network diagram that you can. Write the subnet numbers used on each link onto the diagram.

Example 6-20 *Command Output on Router Fred*

```
fred>show interface
Serial0 is up, line protocol is up
  Hardware is HD64570
  Internet address is 199.1.1.65/27
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     27 packets input, 2452 bytes, 0 no buffer
     Received 27 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     29 packets output, 2044 bytes, 0 underruns
     0 output errors, 0 collisions, 28 interface resets
     0 output buffer failures, 0 output buffers swapped out
     7 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
Serial1 is up, line protocol is up
  Hardware is HD64570
  Internet address is 199.1.1.97/27
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
```

continues

Example 6-20 *Command Output on Router Fred (Continued)*

```
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     125 packets input, 7634 bytes, 0 no buffer
     Received 124 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     161 packets output, 9575 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     4 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
Ethernet0 is up, line protocol is up
Hardware is MCI Ethernet, address is 0000.0c55.AB44 (bia 0000.0c55.AB44)
  Internet address is 199.1.1.33/27
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:00, output hang never
     Output queue 0/40, 0 drops; input queue 0/75, 0 drops
     Five minute input rate 4000 bits/sec, 4 packets/sec
     Five minute output rate 6000 bits/sec, 6 packets/sec
           22197 packets input, 309992 bytes, 0 no buffer
           Received 2343 broadcasts, 0 runts, 0 giants
           0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
           4456 packets output, 145765 bytes, 0 underruns
           3 output errors, 10 collisions, 2 interface resets, 0 restarts


fred>show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     199.1.1.0/27 is subnetted, 6 subnets
R       199.1.1.192 [120/1] via 199.1.1.98, 00:00:01, Serial1
R       199.1.1.128 [120/1] via 199.1.1.98, 00:00:01, Serial1
                    [120/1] via 199.1.1.66, 00:00:20, Serial0
R       199.1.1.160 [120/1] via 199.1.1.66, 00:00:20, Serial0
C       199.1.1.64 is directly connected, Serial0
C       199.1.1.96 is directly connected, Serial1
C       199.1.1.32 is directly connected, Ethernet0

fred>show ip protocol
```

Example 6-20 *Command Output on Router Fred (Continued)*

```
Routing Protocol is "rip"
  Sending updates every 30 seconds, next due in 23 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
 Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: rip
  Default version control: send version 1, receive any version
    Interface        Send  Recv   Key-chain
    Serial0          1     1 2
    Serial1          1     1 2
    Ethernet0        1     1 2
  Automatic network summarization is in effect
  Maximum path: 4
 Routing for Networks:

    199.1.1.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    199.1.1.66           120      00:00:04
    199.1.1.98           120      00:00:14
  Distance: (default is 120)

fred>show cdp neighbor detail
------------------------
Device ID: dino
Entry address(es):
  IP address: 199.1.1.66
Platform: Cisco 2500,  Capabilities: Router
Interface: Serial0,  Port ID (outgoing port): Serial0
Holdtime : 148 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-DS-L), Version 12.2(1), RELEASE SOFTWARE (fc2)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Fri 27-Apr-01 14:43 by cmong

advertisement version: 2
------------------------
Device ID: Barney
Entry address(es):
  IP address: 199.1.1.98
Platform: Cisco 2500,  Capabilities: Router
Interface: Serial1,  Port ID (outgoing port): Serial0
Holdtime : 155 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-DS-L), Version 12.2(1), RELEASE SOFTWARE (fc2)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Fri 27-Apr-01 14:43 by cmong

advertisement version: 2
```

Scenario 6-2: IP Subnet Design with a Class B Network

Your job is to plan a new network. The topology required includes three sites, one Ethernet at each site, and point-to-point serial links for connectivity, as shown in Figure 6-44. The network might grow to need at most 100 subnets, with 200 hosts per subnet maximum. Use network 172.16.0.0, and use the same subnet mask for all subnets. Use Table 6-61 to record your choices, or use a separate piece of paper.

Figure 6-44 *Scenario 6-2 Network Diagram*

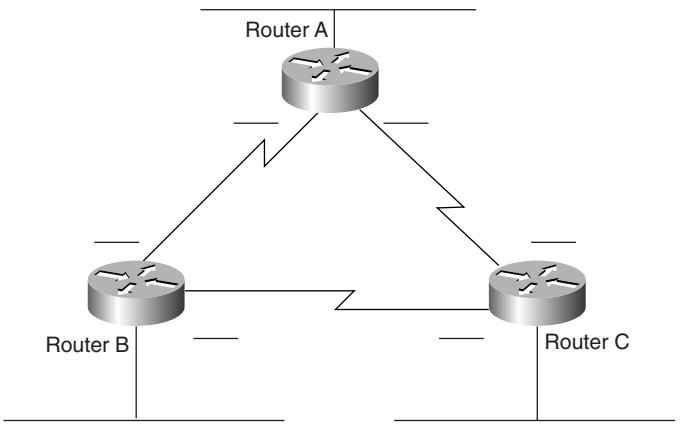


Table 6-61 *Scenario 6-2 Planning Chart*

Location of Subnet Geographically	Subnet Mask	Subnet Number	Router's IP Address
Ethernet off Router A			
Ethernet off Router B			
Ethernet off Router C			
Serial between A and B			
Serial between A and C			
Serial between B and C			

Given the information in Figure 6-44 and Table 6-61, perform the following activities:

1. Determine all subnet masks that meet the criteria in the introduction to this scenario.
2. Choose a mask and pick enough subnets to use for the original topology (refer to Figure 6-44).
3. Create IP-related configuration commands for each router.

Scenario 6-3: IP Subnet Design with a Class C Network

Your job is to plan yet another network. The topology required includes four sites, one Ethernet at each site, and partially meshed Frame Relay for connectivity, as shown in Figure 6-45. The number of subnets will never grow. Choose a mask that will maximize the number of hosts per subnet. Use network 200.1.1.0. Use Table 6-62 to record your choices, or use a separate piece of paper.

Figure 6-45 *Scenario 6-3 Network Diagram*

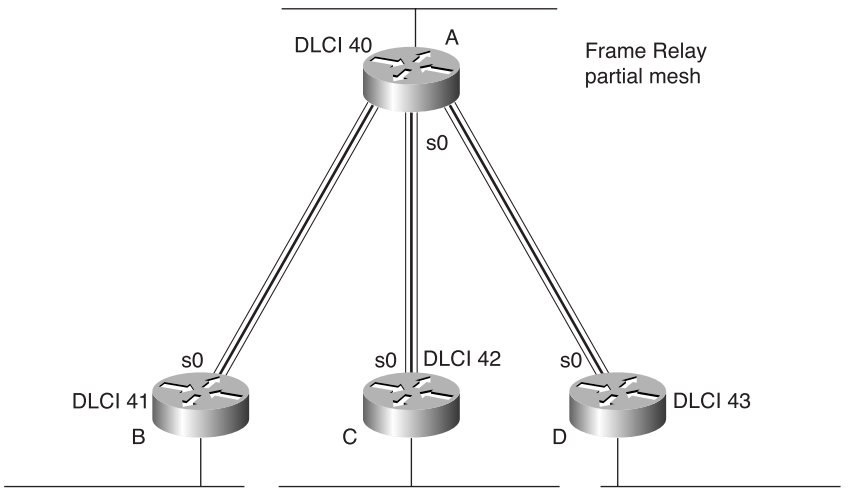


Table 6-62 *Scenario 6-3 Planning Chart*

Location of Subnet	Subnet Mask	Subnet Number	Router's IP Address
Ethernet off Router A			
Ethernet off Router B			
Ethernet off Router C			
Ethernet off Router D			
VC between A and B			
VC between A and C			
VC between A and D			

Given the network setup in Figure 6-45, perform the following tasks:

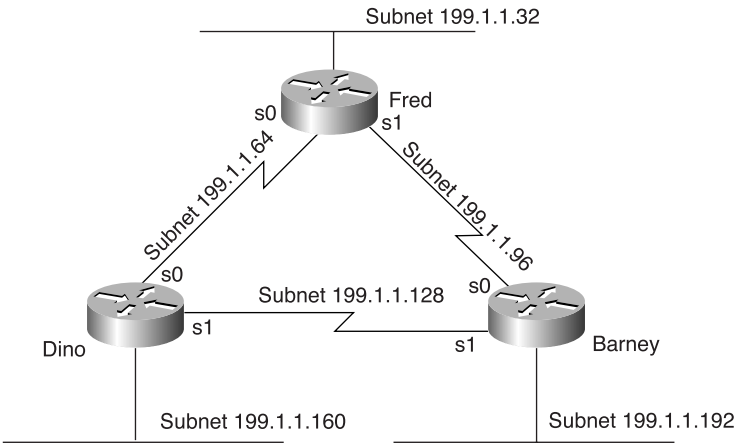
1. Choose the best subnet mask that meets the criteria.
2. Use Table 6-62 to plan which subnet numbers will be used.
3. Create IP-related configuration commands for each router. Use the DLCIs from Figure 6-45.

Scenario Answers

Answers to Scenario 6-1: IP Addressing and Subnet Calculation

Assuming that you had issued the commands in Example 6-20, the most specific network diagram would look like Figure 6-46.

Figure 6-46 *Scenario 6-1 Answer—Network with Router Fred*



The clues that you should have found in the **show** commands are as follows:

- The types and IP addresses of the interfaces on Fred were in the **show interface** and **show ip interface brief** command output.
- The subnets could be learned from the **show ip route** command or derived from the IP addresses and masks shown in the **show interface** command output.
- The neighboring routers' IP addresses could be learned from the **show ip protocol** command.
- The neighboring routers' IP addresses and host names could be learned from the **show cdp neighbor detail** command.
- The metric for subnet 199.1.1.128/27 in RIP updates implies that both neighbors have an equal-cost route to the same subnet. Because two separate but duplicate networks would be a bad design, you can assume that the neighboring routers are attached to the same medium.

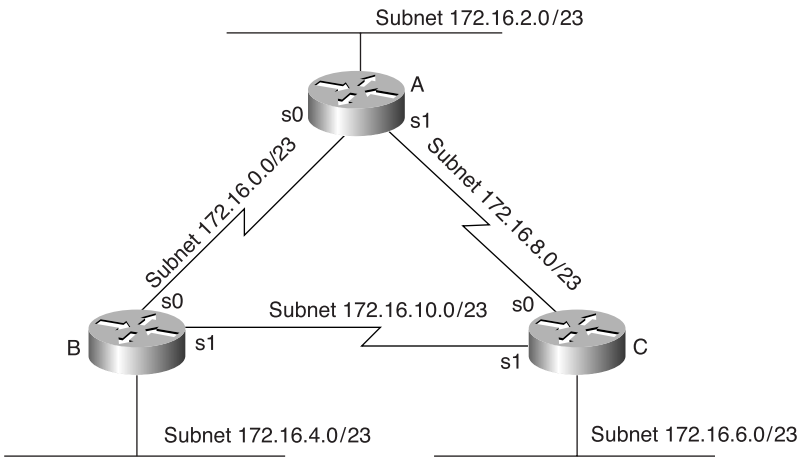
- If you are completely bored, the **telnet 199.1.1.x** command could have been issued for all IP addresses in subnets not directly connected to Fred, hoping to get a router login prompt. That would identify the IP addresses of other router interfaces.

There is no way to know what physical media are beyond the neighboring routers. However, because CDP claims that both routers are 2500 series routers, the possible interfaces on these neighboring routers are limited. Figure 6-46 shows the other subnets as Ethernet segments. Similarly, the figure shows the two neighboring routers attached to the same medium, which is shown as a serial link in Figure 6-46.

Answers to Scenario 6-2: IP Subnet Design with a Class B Network

Figure 6-47 shows one correct answer for the network skeleton presented in Figure 6-44.

Figure 6-47 *Scenario 6-2 Diagram Scratch Pad*



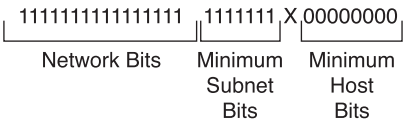
Answers to Task 1 for Scenario 6-2

Given the details in Figure 6-44 and Table 6-61 for Scenario 6-2, the subnet mask criteria are as follows:

- 200 hosts in a subnet, maximum
- 100 subnets, maximum
- Static size masks used all over this network

The mask must have at least eight host bits because $2^7 \times 128$ is not enough and $2^8 \times 256$ is more than enough for numbering 200 hosts in a subnet. The mask must have at least seven subnet bits, likewise, because 2^7 is the smallest power of 2 that is larger than 100, which is the required number of subnets. The first 16 bits in the mask must be binary 1 because a Class B network (172.16.0.0) is used. Figure 6-48 diagrams the possibilities.

Figure 6-48 *Subnet Mask Options for Scenario 6-2*



The only bit position in which a decision can be made is the 24th bit, shown with an X in Figure 6-48. That leaves two mask possibilities: 255.255.254.0 and 255.255.255.0. This sample shows the 255.255.254.0 mask just so you can have a little more practice with harder masks. Given the choice in a real network, choose the easy mask!

Answers to Task 2 for Scenario 6-2

To choose a mask and pick enough subnets to use for the original topology illustrated in Figure 6-44, a review of the longer binary algorithm and shortcut algorithm for deriving subnet numbers is required. To review, subnet numbers have the network number value in the network portion of the subnet numbers and have all binary 0s in the host bits. The bits that vary from subnet to subnet are the subnet bits—in other words, you are numbering different subnets in the subnet field.

Valid subnets with mask 255.255.254.0 are as follows:

172.16.0.0 (zero subnet)
172.16.2.0
172.16.4.0
172.16.6.0
.
.
.
172.16.252.0
172.16.254.0 (broadcast subnet)

The first six subnets, including the zero subnet, were chosen for this example, as listed in Table 6-63.

Table 6-63 *Scenario 6-2 Subnets and Addresses*

Location of Subnet Geographically	Subnet Mask	Subnet Number	Router's IP Address
Ethernet off Router A	255.255.254.0	172.16.2.0	172.16.2.1
Ethernet off Router B	255.255.254.0	172.16.4.0	172.16.4.2
Ethernet off Router C	255.255.254.0	172.16.6.0	172.16.6.3
Serial between A and B	255.255.254.0	172.16.0.0	172.16.0.1 (A) and .2 (B)
Serial between A and C	255.255.254.0	172.16.8.0	172.16.8.1 (A) and .3 (C)
Serial between B and C	255.255.254.0	172.16.10.0	172.16.10.2 (B) and .3 (C)

Answers to Task 3 for Scenario 6-2

Given the details in Figure 6-44 and Table 6-61 for Scenario 6-2, the configurations in Examples 6-21 through 6-23 satisfy the exercise of creating IP-related configuration commands for each router. These examples include only the IP-related commands.

Example 6-21 *Router A Configuration, Scenario 6-2*

```
ip subnet-zero
no ip domain-lookup
!
interface serial0
ip address 172.16.0.1 255.255.254.0
interface serial 1
ip address 172.16.8.1 255.255.254.0
interface ethernet 0
ip address 172.16.2.1 255.255.254.0
!
router igrp 6
network 172.16.0.0
```

Example 6-22 *Router B Configuration, Scenario 6-2*

```
ip subnet-zero
no ip domain-lookup
!
interface serial0
ip address 172.16.0.2 255.255.254.0
interface serial 1
ip address 172.16.10.2 255.255.254.0
interface ethernet 0
ip address 172.16.4.2 255.255.254.0
!
router igrp 6
network 172.16.0.0
```

Example 6-23 *Router C Configuration, Scenario 6-2*

```
ip subnet-zero
no ip domain-lookup
!
interface serial0
ip address 172.16.8.3 255.255.254.0
interface serial 1
ip address 172.16.10.3  255.255.254.0
interface ethernet 0
ip address 172.16.6.3 255.255.254.0
!
router igrp 6
network 172.16.0.0
```

Answers to Scenario 6-3: IP Subnet Design with a Class C Network

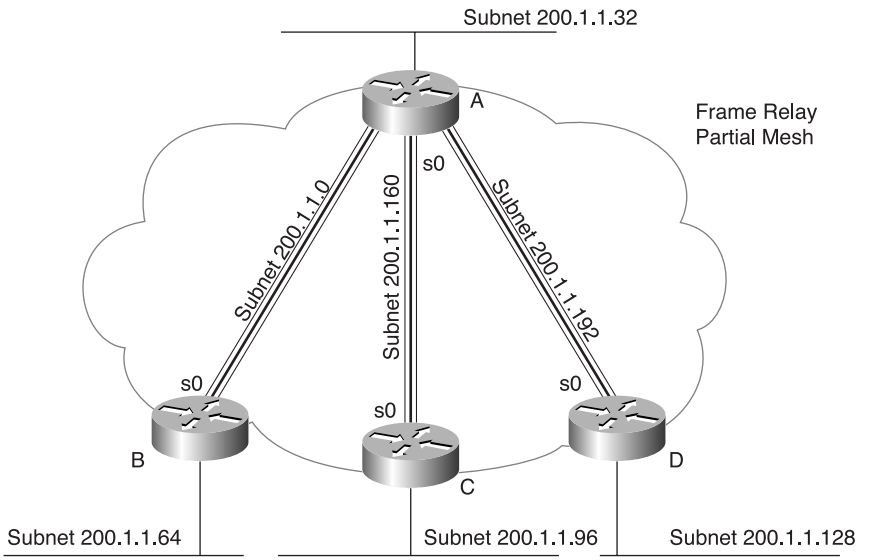
Planning the network in this scenario requires a topology that includes four sites, one Ethernet at each site, and partially meshed Frame Relay for connectivity, as shown previously in Figure 6-45. The number of subnets will never grow. You must choose a mask that will maximize the number of hosts per subnet, and you must use network 200.1.1.0.

Answers to Task 1 for Scenario 6-3

Given the design criteria and the network setup illustrated in Figure 6-45, this scenario requires tricky subnet masks because a Class C network is used and because subnetting is needed. Using Frame Relay subinterfaces, there will be a need for seven different subnets—one for each Ethernet and one for each Frame Relay VC.

If three subnet bits are used, eight mathematical possibilities exist for subnet numbers. However, one is the zero subnet and the other is the broadcast subnet. In this case, use of one of these is desired because the design called for maximizing the number of hosts per subnet. Deciding against use of the zero and broadcast subnets then would require four subnet bits, leaving only four host bits, implying 14 hosts per subnet. So, three subnet bits and five host bits will be used in this solution (mask of 255.255.255.224). Figure 6-49 summarizes the subnets on the network diagram.

Figure 6-49 *Scenario 6-3 Network, with Subnets Written on Diagram*



Answers to Task 2 for Scenario 6-3

Given the design criteria and the network setup illustrated in Figure 6-45 for Scenario 6-3, Table 6-64 shows the choices of subnets and addresses in this example. Only one subnet, 200.1.1.224, which is the broadcast subnet, is not used. Of course, you could have chosen a different set of subnets and used them on different links, but the mask you used should have been 255.255.255.224, based on the criteria to maximize the number of hosts per subnet.

Table 6-64 *Scenario 6-3 Subnets and Addresses*

Location of Subnet	Subnet Mask	Subnet Number	Router's IP Address
Ethernet off Router A	255.255.255.224	200.1.1.32	200.1.1.33
Ethernet off Router B	255.255.255.224	200.1.1.64	200.1.1.65
Ethernet off Router C	255.255.255.224	200.1.1.96	200.1.1.97

continues

Table 6-64 *Scenario 6-3 Subnets and Addresses (Continued)*

Location of Subnet	Subnet Mask	Subnet Number	Router's IP Address
Ethernet off Router D	255.255.255.224	200.1.1.128	200.1.1.129
VC between A and B	255.255.255.224	200.1.1.0	200.1.1.1 (A) and .2 (B)
VC between A and C	255.255.255.224	200.1.1.160	200.1.1.161 (A) and .162 (B)
VC between A and D	255.255.255.224	200.1.1.192	200.1.1.193 (A) and .194 (B)

Answers to Task 3 for Scenario 6-3

Using the DLCIs from Figure 6-45, you can find the IP-related configuration commands for each router in Examples 6-24 through 6-27.

Example 6-24 *Router A Configuration, Scenario 6-3*

```
ip subnet-zero
no ip domain-lookup
!
interface serial0
encapsulation frame-relay
interface serial 0.1
ip address 200.1.1.1 255.255.255.224
frame-relay interface-dlci 41
!
interface serial 0.2
ip address 200.1.1.161 255.255.255.224
frame-relay interface-dlci 42
!
interface serial 0.3
ip address 200.1.1.193 255.255.255.224
frame-relay interface-dlci 43
!
interface ethernet 0
ip address 200.1.1.33 255.255.255.224
!
router igrp 6
network 200.1.1.0
```

Example 6-25 *Router B Configuration, Scenario 6-3*

```
ip subnet-zero
no ip domain-lookup
!
interface serial0
```

Example 6-25 *Router B Configuration, Scenario 6-3 (Continued)*

```
encapsulation frame-relay
interface serial 0.1
ip address 200.1.1.2 255.255.255.224
frame-relay interface-dlci 40
!
interface ethernet 0
ip address 200.1.1.65 255.255.255.224
!
router igrp 6
network 200.1.1.0
```

Example 6-26 *Router C Configuration, Scenario 6-3*

```
ip subnet-zero
no ip domain-lookup
!
interface serial0
encapsulation frame-relay
interface serial 0.1
ip address 200.1.1.162 255.255.255.224
frame-relay interface-dlci 40
!
interface ethernet 0
ip address 200.1.1.97 255.255.255.224
!
router igrp 6
network 200.1.1.0
```

Example 6-27 *Router D Configuration, Scenario 6-3*

```
ip subnet-zero
no ip domain-lookup
!
interface serial0
encapsulation frame-relay
interface serial 0.1
ip address 200.1.1.194 255.255.255.224
frame-relay interface-dlci 40
!
interface ethernet 0
ip address 200.1.1.129 255.255.255.224
!
router igrp 6
network 200.1.1.0
```