For effective communication across a network, computers must be capable of transmitting data reliably and efficiently. Network protocols are designed to accomplish this goal, with some protocols emphasizing reliability and others efficiency. Network protocols often work together at different layers of the network communication process to provide both reliability and efficiency. Network administrators must understand the role and function of protocols, as much of their time is spent configuring and troubleshooting the protocols used by the network's clients and servers. This chapter discusses network protocols in general but focuses on the most common suite of protocols used in today's networks: TCP/IP.

TCP/IP's Layered Architecture

The term "protocol" isn't specific to the field of networking. In general, **protocols** are rules and procedures for communication and behavior or etiquette. Just as two people must share a common set of rules for verbal communication—a language—computers must also "speak" the same language and agree on the rules of communication. You use protocols in other ways. Texting, e-mail, and Facebook communication, for example, have their own rules of etiquette and, especially for texting, their own language.

Until fairly recently, you had a choice of network protocols you could install on your computer, depending on the computing environment. A small network in the 1990s running Windows 3.1 or Windows 95 probably ran the Windows-specific NetBEUI protocol. A network with Novell NetWare 4.x servers typically ran IPX/SPX. Both these protocols are obsolete now and are found only in networks that haven't been upgraded in more than a decade. Today, you can focus on the TCP/IP protocol suite, the protocol of the Internet and the default protocol all contemporary OSs run.

When a set of protocols works cooperatively, it's called a **protocol stack** or **protocol suite**. The most common protocol stack is **Transmission Control Protocol/Internet Protocol (TCP/IP)**, the Internet protocol suite. Although you can see by its name that TCP/IP consists of at least two protocols—TCP and IP—this protocol suite is actually composed of more than a dozen protocols operating at different layers of the communication process.

Recall the communication process explained in Chapter 1 and animated in Simulation 1. This discussion was an introduction to the idea of communication taking place in layers. The protocols in TCP/IP can also be divided into four layers, with similar names and functions. Figure 5-1 shows the layers of the TCP/IP protocol suite and which protocols operate at each layer. This layered architecture is usually referred to as the "TCP/IP model."

Many books and Web sites about TCP/IP call the Internetwork layer the "Internet layer," but the term "internetwork" describes the layer's function more accurately, especially because many people use the term Internet interchangeably with the term World Wide Web. Also, the Network access layer is often referred to as the "Network interface layer." Although both terms describe this layer's function, "Network access" has been used in this book.

The TCP/IP protocol suite includes more protocols than are shown in Figure 5-1, but the ones listed are some of the most common protocols used in networks. Before you examine each layer and protocol more closely, take a look at an example of how the layers work together.

Layer name	TCP/IP protocols			
Application	HTTP	FTP	DHCP	TFTP
	SMTP	POP3	DNS	SNMP
Transport	TCP		UDP	
Internetwork	ICMP	ARP	IPSec	
	IPv4 and IPv6			
Network access	Ethernet, token ring, FDDI, WAN technologies			

Figure 5-1 The TCP/IP layered architecture

Courtesy of Course Technology/Cengage Learning

Suppose you start your Web browser and have configured your home page as *http://www.course.com*. The Web browser formats a request for a page on the *www.course.com* Web server by using the Application-layer protocol HTTP. The request looks something like Figure 5-2.

Figure 5-2 The Application layer creates data

Courtesy of Course Technology/Cengage Learning

You've learned about packets and frames, but the unit of information the Application layer works with is simply called "data." The Application-layer protocol HTTP passes the request down to the Transport-layer protocol: in this case, TCP. Notice that the four Application-layer protocols on the left of Figure 5-1 use TCP as the Transport-layer protocol, and the Application-layer protocols on the right use UDP. (The difference between TCP and UDP is explained later in "Role of the Transport Layer.") TCP has its own job to do, so it adds a header to the request that looks like Figure 5-3.

Figure 5-3 The Transport layer adds its header to make a segment

Courtesy of Course Technology/Cengage Learning

The unit of information the Transport layer works with is called a **segment**. TCP passes the segment to the Internetwork layer. The Internetwork layer has a number of subprotocols, but most operate by following the basic rules and format of IP. IP then places its header on the segment, making it a packet (see Figure 5-4).

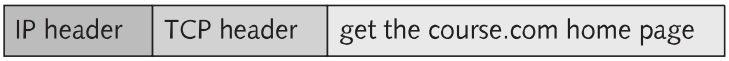

Figure 5-4 The Internetwork layer creates a packet

Courtesy of Course Technology/Cengage Learning

The packet is almost ready for delivery to the network medium, with one more stop at the Network access layer, where the NIC operates. As you know, NICs work with frames, so a frame header and trailer are added (see Figure 5-5).

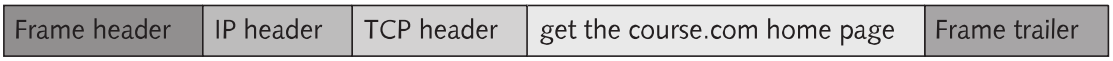

Figure 5-5 The frame is created and ready for delivery on the medium

Courtesy of Course Technology/Cengage Learning

The frame is then delivered to the network medium as bits on its way to the *www.course.com* server, where the Web server software processes it and returns a Web page to the computer that originated the request. Now that you have an idea of how these protocols work together, examine the roles of these four layers more closely, starting from the bottom: the Network access layer.

Hands-On Project 5-1: Viewing TCP/IP Layers in Windows and Configuring Your IP Address

Time Required: 10 minutes

Objective: View the properties of your computer's network connection, identify the TCP/IP layers, and configure your IP address.

Required Tools/Equipment: Your classroom computer

Description: In this project, you view the properties of your computer's local area connection and identify the TCP/IP layers. This project is similar to Hands-On Project 1-3, but you're viewing the TCP/IP protocol suite layers instead of the more general layers of the networking process. Next, you configure your IPv4 address.

1. Start your computer and log on as **NetAdmin**.
2. Open the Network and Sharing Center by clicking **Start, Control Panel**. Under Network and Internet, click **View network status and tasks**.
3. In the left pane of the Network and Sharing Center, click **Change adapter settings**. Right-click **Local Area Connection** and click **Properties** to open the Local Area Connection Properties dialog box.
4. The Connect using text box displays the network interface card. In the list box under it, you see several items. Client for Microsoft Networks, File and Printer Sharing for Microsoft Networks, Internet Protocol Version 4, and Internet Protocol Version 6 are the items you're interested in right now, as they're the most necessary software components for making network communication work.
5. For each component, write which TCP/IP layer or layers you think it operates in:
   - NIC displayed in the Connect using text box: _____

- Client for Microsoft Networks: _____
- File and Printer Sharing for Microsoft Networks: _____
- Internet Protocol Version 4: _____
- Internet Protocol Version 6: _____

6. Next, you configure your IP address settings. Click **Internet Protocol Version 4 (TCP/IPv4)** and click **Properties**.
7. If your IP settings have the "Obtain an IP address automatically" option enabled, click **Use the following IP address**. You use this option to set a static IP address. If your address is already static, make a note of it, and skip entering the information in Step 8. Click **OK**.
8. For the following IP address settings fields, enter the information shown unless your instructor tells you to use different values, and then click **OK** when you're finished:
   - IP address: **192.168.100.*XX*** (replacing *XX* with your student number)
   - Subnet mask: **255.255.255.0**
   - Default gateway: provided by your instructor
   - Preferred DNS server: provided by your instructor
9. Click **Close**. If you're prompted to set a network location, click **Work network**, and then click **Close**.
10. To test your configuration, open a command prompt window and try to ping your default gateway address and your preferred DNS server address. If either ping is unsuccessful, inform your instructor and troubleshoot your settings.
11. Close all open windows, but leave your computer running for the next project.

The following list recaps your IP address configuration and explains each item's purpose:

- The IP address provides your computer with a unique internetwork identity on a logical IP network.
- The subnet mask defines which part of the IP address is the network ID and which is the host ID.
- The default gateway is a router in your network that your computer sends packets to when the destination is a remote network.
- The preferred DNS server is the address of a DNS server that resolves computer names to IP addresses.

These items are explained in more detail in this chapter and throughout this book.

Hands-On Project 5-2: Identifying the TCP/IP Layers in a Frame

Time Required: 10 minutes

Objective: Capture packets and view the TCP/IP layers in the frame.

Required Tools/Equipment: Your classroom computer with Wireshark installed

Description: In this project, you capture some frames generated by your Web browser and examine the captured frames to identify the TCP/IP layers.

1. If necessary, log on to your computer as **NetAdmin**.
2. Start Wireshark and click **Capture Options**. In the Capture Filter text box, type **tcp port http**, and then click **Start**.
3. Start a Web browser, and after the home page loads, exit the browser.
4. In Wireshark, click the **Stop the running live capture** toolbar icon to stop the capture. Scroll up to the first packet summary line, if necessary.
5. Click a packet summary in the top pane with HTTP in the protocol field and an Info line beginning with GET. In the middle pane are summaries of each protocol header (see Figure 5-6). You can ignore the first line starting with Frame *X* (with *X* representing the frame number), as it gives information about the frame, such as the time it arrived, its length, protocols in the frame, and so forth.

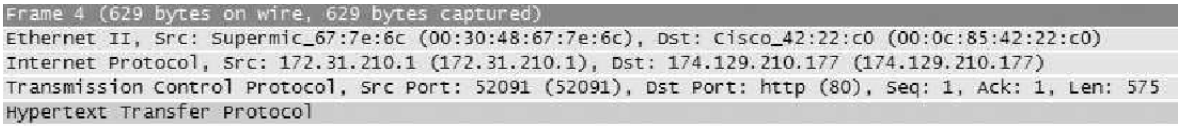

Figure 5-6 Summary of protocol headers in Wireshark

Courtesy of Course Technology/Cengage Learning

6. Click to expand the line beginning with **Ethernet II**. Examine the information in this header (discussed in more detail in the following sections). Write which layer of the TCP/IP model the Ethernet II header represents, and then click again to collapse this header:

   _____

7. Click to expand the line beginning with **Internet Protocol**. Examine the information in this header (discussed in more detail in the following sections). Write which layer of the TCP/IP model the Internet Protocol header represents, and then click again to collapse this header:

   _____

8. Click to expand the line beginning with **Transmission Control Protocol**. Examine the information in this header (discussed in more detail in the following sections). Write which layer of the TCP/IP model the Transmission Control Protocol header represents, and then click again to collapse this header:

   _____

9. Click to expand the line beginning with **Hypertext Transfer Protocol**, and examine the information. This data portion of the frame is what a Web server actually sees and responds to. In this case, the HTTP command is `GET`, which means HTTP is requesting a page (or part of a page) from the Web server. Write which layer of the TCP/IP model the HTTP protocol represents, and then click again to collapse this header:

   _____

10. Exit Wireshark and click **Quit without Saving** when prompted.
11. Close all open windows, but leave your computer running for the next project.

Role of the Network Access Layer

Strictly speaking, the Network access layer isn't composed of TCP/IP protocols. As you can see, network technologies, such as Ethernet and token ring, operate at this layer. So this layer is part of the TCP/IP architecture only to the extent that the layer above—the Internetwork layer—has the capability to communicate with any network technologies following the rules of the Network access layer. Some tasks the Network access layer performs have already been discussed but are worth repeating here:

- Provides a physical (MAC) address for the network interface
- Verifies that incoming frames have the correct destination MAC address
- Defines and follows media access rules
- Receives packets from the Internetwork layer and encapsulates them to create frames
- Deencapsulates received frames and sends the resulting packets to the Internetwork layer
- Often provides frame error detection in the form of a CRC code
- Transmits and receives bit signals
- Defines the signaling needed to transmit bits, whether electrical, light pulses, or radio waves
- Defines the media and connectors needed to make a physical network connection

As you learn in Chapter 6, the last three items in this list are tasks the Physical layer performs in the more detailed OSI networking model, which splits the Network access layer into two separate layers.

Role of the Internetwork Layer

The Internetwork layer is where administrators usually do the most network configuration. It's where the IP protocol operates, and it can be looked at as the heart of the TCP/IP protocol suite. IP addresses, of course, are defined here, and routing takes place in this layer, too. Without routing, the Internet and World Wide Web wouldn't exist. With all the complexity of configuring routing and managing IP addresses, this layer is also where most errors in network configuration occur. In a large internetwork, a lot of your time is typically spent unraveling the intricacies of the Internetwork layer.

The Internetwork layer is responsible for four main tasks, discussed in the following sections:

- Defines and verifies IP addresses
- Routes packets through an internetwork
- Resolves MAC addresses from IP addresses
- Delivers packets efficiently

Defines and Verifies IP Addresses An IP address is assigned to every computer and network device using TCP/IP for communication. IP addresses are used for two main purposes: to identify a network device at the Internetwork layer and to identify the network on which a device resides. When an IP address is assigned to a computer or network device (referred to as an "IP host" or just "host"), the host's Internetwork-layer identity is defined. When a host receives an IP packet, it compares the packet's destination IP address with its own address to verify that the packet was delivered correctly. If the destination address matches or is a broadcast or recognized multicast address, the packet is processed;

otherwise, it's discarded. When a host sends a packet, the IP protocol places its own IP address in the packet header's source field before sending the packet to the network interface.

The IP address is also used to identify the network on which a host resides. Every IP address contains two parts: a network ID and a host ID. This format is similar to a 10-digit phone number, with a three-digit area code identifying the region of the country where the number was assigned and a seven-digit number identifying the particular phone. IP addresses aren't as straightforward in their format, as you discover later in "IP Addressing," but there's always a portion of an IP address that identifies the network on which the host resides.

Routes Packets Through an Internetwork The next task of the Internetwork layer is determining the best way to get a packet from network to network until it reaches its destination. If there were only one way for a packet to get from here to there, this aspect of the Internetwork layer's job would be pretty ho-hum. However, much like the nation's road system, most large networks, such as the Internet, have multiple paths for getting from location A to location B. Which path to take isn't always a clear-cut decision. Some paths are heavily traveled, and some are lightly traveled; some paths have construction or accidents, and others are clear sailing.

As mentioned, routers work at the Internetwork layer, and their job is to select the best path to the destination. If a path becomes unavailable or congested, they select an alternative, if available. Routers use the network ID portion of IP addresses along with their routing tables to determine on which network a destination device can be found and the best way to get packets to their destination. Chapter 7 discusses routers in more detail.

Resolves MAC Addresses from IP Addresses As you've learned, every frame sent to the network medium contains both physical (MAC) and logical (IP) source and destination addresses. When a packet is ready to be sent to the Network access layer, the destination device's MAC address must be retrieved before the frame header can be constructed. TCP/IP uses Address Resolution Protocol (ARP) for this task. ARP is discussed in more detail later in "Address Resolution Protocol," but in a nutshell, it returns a computer's MAC address by asking the network which computer is assigned a particular IP address.

Delivers Packets Efficiently Internetwork-layer protocols focus mainly on efficient delivery of packets. The secret to achieving this efficiency is low processing overhead. Internetwork-layer protocols don't include features such as flow control, delivery confirmation, or message reassembly; these features require considerable overheard to ensure reliable delivery, at the cost of efficiency.

Internetwork protocols rely on protocols in the Transport and Application layers to provide advanced reliability features. Protocols at this layer are concerned with one packet at a time, with no concern for packets that came before or after it and with no confirmation that delivery was successful. This communication strategy is called connectionless communication, and protocols using it are called connectionless protocols.

When using a **connectionless protocol**, no lasting connection is made from source to destination. A connectionless protocol relies on an upper-layer protocol to ensure the packet's safe journey. This process is much like delivering a first-class letter via the U.S. mail. You drop the letter in the mailbox and hope it makes it to its destination. Usually it does, but when you want to be certain it

was received (or notified if it wasn't), you must add a layer of complexity by sending the letter certified mail, which requires an acknowledgement that the letter was received. In networking, protocols providing this acknowledgement are called connection-oriented protocols. In the TCP/IP model, these protocols are in the Transport layer, discussed later in "Role of the Transport Layer."

Protocols at the Internetwork Layer

IP is the underlying basis for most Internetwork-layer protocols, which means they just send specialized versions of IP packets. The protocols operating at this layer are too numerous to describe in this book, so the following sections focus on the most commonly used:

- IPv4
- IPv6
- ICMP
- ARP
- IPSec

Internet Protocol Version 4

Internet Protocol Version 4 (IPv4), or just IP, is an Internetwork-layer protocol that provides source and destination addressing and routing for the TCP/IP protocol suite. IP is a connectionless protocol, so it's efficient but unreliable. Note that "unreliable" doesn't mean it fails often. In this context, it simply means IP has no method for ensuring that data is delivered to the destination. IP assumes the Transport or Application layer provides reliable data delivery in applications that require it.

IPv4 is the most common version in networks and the first version that was in widespread use. Earlier versions never really made it out of the lab. One of IP's most important functions is the definition of logical addresses, naturally called **IP addresses**. IPv4 defines a 32-bit dotted decimal address: 172.31.149.10, for example. Each grouping of numbers separated by a dot (period) is an 8-bit value that can range from 0 to 255. Because an IP address has 32 bits, a total of 2^{32} addresses are possible, which is approximately 4 billion. That might seem like a lot of addresses, but as you learn later, many are wasted, and available addresses to assign to devices on the Internet are running out.

As mentioned, in the IP address format, part of the address specifies the network where the computer assigned the address is located, and the rest of the address specifies a unique host ID in the network. For example, using the address 172.31.149.10, 172.31 might be the network ID and 149.10 might be the host ID. This topic can be complex and is covered in more detail later in "IP Addressing."

IP works with packets, and when it receives a message from the layers above, it adds an IP header. So far in this chapter, only the destination (the intended recipient) and source (the sending machine) addresses of the IP header have been discussed. There's quite a bit more to an IP header, and the following list describes the more important fields in the order they appear in a packet:

- *Version*—This field simply indicates which version of IP is in use. Today, the possibilities are 4 and 6. Because computers can run both versions at the same time, the Version field tells the computer whether the packet should be processed by IPv4 or IPv6.
- *Time to live*—The TTL field is a safeguard that prevents a packet from wandering aimlessly through an internetwork, which can be caused by a network misconfiguration.

Before a packet is sent to the network, the TTL field is given a value, usually 64, 128, or 255. As the packet travels through the internetwork, the TTL value is decremented at each router. If the TTL value reaches 0, the packet is deemed to have expired, and the router that decremented the packet to 0 discards it. Most routers also send a message back to the source address as notification that the packet expired.

- *Protocol*—This field is a numeric code specifying the type of IP packet or the next layer protocol contained in the packet. For example, the Protocol value in an ICMP packet (used by the Ping program) is 1. If the packet contains a Transport-layer protocol, such as TCP, the value is 6. There are more than 140 different types of IP packets.
- *Checksum*—This field is a value produced by a mathematical calculation on data in the header that protects the IP header's contents. When a network device receives a packet, this value is recalculated and compared with the value in the Checksum field. If they match, the header hasn't been altered.
- *Source address*—This field is self-explanatory.
- *Destination address*—This field is self-explanatory, too, but it can be one of three types, as in a MAC address: unicast (intended for a single computer), broadcast (sent to all computers in the network), or multicast (sent to a group of computers).

Internet Protocol Version 6 **Internet Protocol version 6 (IPv6)** has many of the features of IPv4 but addresses IPv4's shortcomings, not the least of which is its inadequate 32-bit address space. The advantage of a layered approach to networking is that IPv6 can run on computers alongside IPv4 without needing to change the Transport layer or Network access layer. Most Application-layer protocols require no changes either, except those dealing directly with IP addresses, such as DHCP and DNS. IPv6 is discussed in more detail later in "Introduction to Internet Protocol Version 6."

Address Resolution Protocol **Address Resolution Protocol (ARP)** is used to resolve a logical (IP) address to a physical (MAC) address. When a system begins a conversation with a host and doesn't have its MAC address to create the frame header, it sends an ARP broadcast frame requesting the MAC address corresponding to the host's IP address. A network device configured with the specified IP address responds with an ARP reply message containing its MAC address. Then the packet is sent to the Network access layer, and the frame can be constructed.

This process requires additional explanation, as you might be wondering what happens when the two computers are on separate networks on a corporate internetwork or even miles apart on the Internet. As discussed in Chapter 2, routers are responsible for getting packets from one network to another, and they don't forward broadcast packets, which makes routers the delimiting device for broadcast domains. If routers did forward broadcasts, when any computer on the Internet sent a broadcast, the message would be forwarded to every LAN on the Internet, and the Internet would be overrun with broadcasts.

When a computer using TCP/IP wants to communicate with another computer, it must know the destination computer's IP address. Usually, the application sending the message knows the address, or it's resolved by using a name lookup. If the destination's IP address is on the same network as the source, the source computer sends an ARP request message in the form of a broadcast. All computers in the broadcast domain process the ARP request, and the computer with this IP address sends back an ARP reply containing its MAC

address. After the MAC address is received, the frame header can be constructed, and the frame is delivered to the destination.

If a computer has to send an ARP broadcast every time it wants to send an IP packet to a particular destination, the network would have one broadcast ARP frame for every frame carrying actual data, which is a big waste of bandwidth. To avoid sending an ARP request every time an IP packet is sent, PCs and other devices store learned IP address/MAC address pairs in an **ARP cache**, a temporary location in RAM. (You viewed your ARP cache in Hands-On Project 1-4 with the `arp -a` command.) So a computer or router has to send an ARP broadcast only once for each destination host it communicates with on its network.

ARP cache entries aren't kept indefinitely. Most computers keep each entry for only a few minutes after it's last used to avoid storing inaccurate information, which could result from a changed NIC or IP address.

If the destination computer is on another network, the computer uses ARP to retrieve the MAC address of the router configured as its default gateway. The packet is delivered to the router, and the router determines where the packet should go next to get to its destination. When the packet gets to the destination network, the router on the destination network uses ARP to get the destination computer's MAC address. Figure 5-7 illustrates this process, and it's animated in Simulation 9. Notice that the destination MAC address in the original

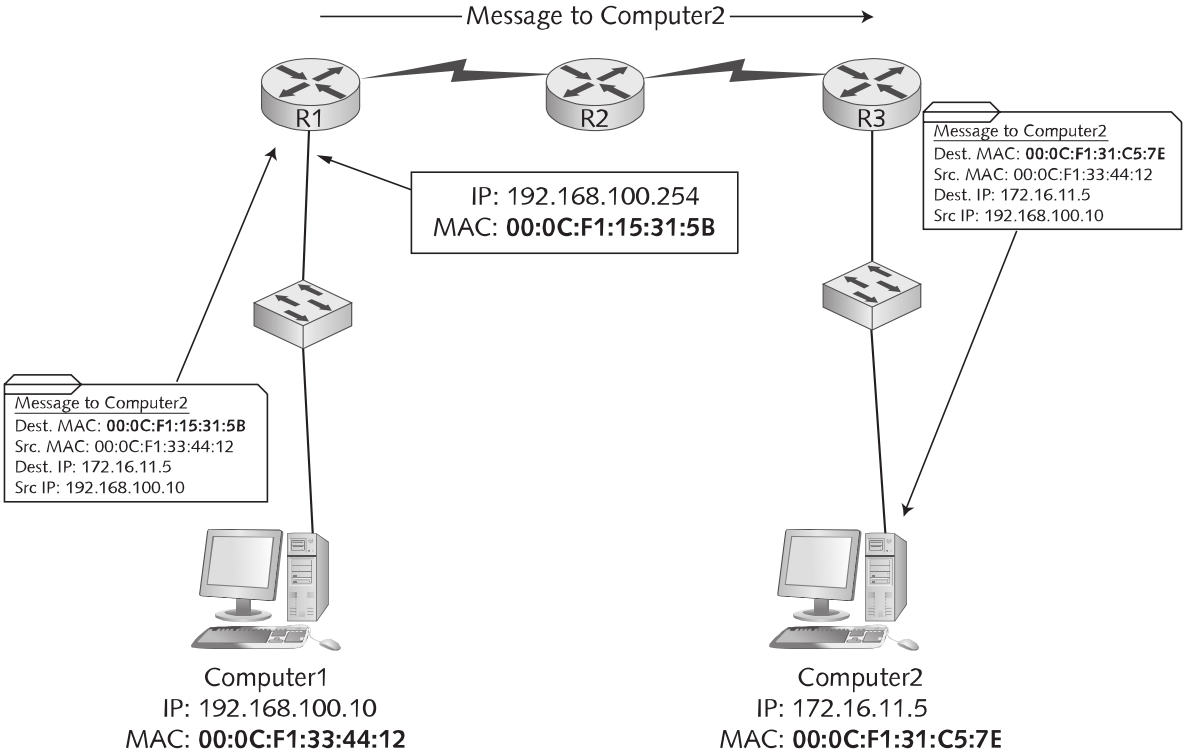

Figure 5-7 How MAC addresses are used in an internetwork

Courtesy of Course Technology/Cengage Learning

message is the MAC address of router R1, but the destination IP address remains the same throughout the journey. Only when the message gets to the destination network does the MAC address become Computer2's address. Notice also that the source MAC address in the frame going from router R3 to Computer2 has changed, showing that the frame is coming from router R3.

Simulation 9: The changing frame header

Internet Control Message Protocol

Internet Control Message Protocol (ICMP) is used to send error and control messages between systems or devices. It's an encapsulated IP protocol, meaning it's wrapped in an IP header. In essence, ICMP is just a specialized IP packet with its own header.

ICMP has many message types, but the two most people know are ICMP Echo (sent by the Ping program) and ICMP Reply (sent by the target of the ping). Ping uses ICMP Echo packets to request a response from another computer to verify whether it's available for communication. The response, if received, is an ICMP Reply packet indicating not only that the remote host is reachable, but also how long the message's round trip from sender to receiver took.

The Tracert program uses ICMP Echo packets to determine the route a packet takes through an internetwork. It sends an ICMP Echo packet with the TTL value in the IP header set to 1. When the packet reaches the first router on its way to the destination, the router decrements the TTL value to 0, discards the packet, and sends a TTL-Expired ICMP packet to the sending machine to notify it that the packet expired. Tracert receives the TTL-Expired message containing the router's IP address and then has the address of the first router in the path to the destination. Next, Tracert sends an ICMP Echo packet with a TTL of 2. When the packet gets to the second router, it again expires, and the second router sends a TTL-Expired message. In this way, Tracert discovers the IP address of every router between the source and destination computers.

These uses of ICMP are the most common, but ICMP has more than 20 message types, many of which are now obsolete. To learn more about ICMP message types, see *www.iana.org/assignments/icmp-parameters*. The Internet Assigned Numbers Authority (IANA) is in charge of everything related to numbers used in Internet protocols.

Internet Protocol Security

Internet Protocol Security (IPSec) works with IPv4 to ensure secure delivery of packets. Most OSs now support IPSec as a feature that can be enabled for certain types of communication between specific computers. In other words, IPSec can be used to secure sensitive network transmissions between computers needing the extra security.

IPSec provides security by using authentication and encryption. It authenticates the identity of computers transmitting data with a password or some other form of credentials, and it

encrypts data so that if packets are captured, the data will be unintelligible. IPSec requires additional network and computer resources, so it should be enabled only for highly sensitive communication and in environments where security risks are high.

Hands-On Project 5-3: Examining the Internetwork Layer

Time Required: 10 minutes

Objective: Capture packets and view the Internetwork layer.

Required Tools/Equipment: Your classroom computer with Wireshark installed

Description: In this project, you capture some ICMP packets and examine the IP header information.

1. If necessary, log on to your computer as **NetAdmin**.
2. Start Wireshark and click **Capture Options**. In the Capture Filter text box, type **icmp**, and then click **Start**.
3. Ping your default gateway or DNS server. If you don't remember these IP addresses, use the `ipconfig` command to display them.
4. In Wireshark, click the **Stop the running live capture** toolbar icon to stop the capture. Scroll up to the first packet summary line, if necessary.
5. Click a packet summary in the top pane with ICMP in the Protocol field and Echo (ping) request in the Info field.
6. Click to expand the line beginning with **Internet Protocol**. In the header, find the fields discussed previously in "Protocols at the Internetwork Layer." Write the values in the Version, Time to live, Protocol, and Checksum fields:

_____

_____

7. Click to expand the line beginning with **Internet Control Message Protocol**, and examine the information in the ICMP header. The Type field specifies the type of ICMP message. The data portion of the ICMP field is simply a string of letters. It doesn't matter what's in the data part of the Echo Request message, as long as the reply contains the same data.
8. Exit Wireshark and click **Quit without Saving** when prompted.
9. Close all open windows, but leave your computer running for the next project.

Hands-On Project 5-4: Capturing ARP and ICMP Packets

Time Required: 10 minutes

Objective: Use Wireshark to capture packets created by the Tracert program.

Required Tools/Equipment: Classroom computers with Wireshark installed

Description: In this project, you use Wireshark to capture ARP and ICMP packets generated by the Tracert program.

1. If necessary, log on to your computer as **NetAdmin** and open a command prompt window.
2. Type **arp -d** and press **Enter** to clear your ARP cache.
3. Start Wireshark and click **Capture Options**. In the Capture Filter text box, type **arp or icmp**, and then click **Start**.
4. At the command prompt, type **tracert books.tomsho.com** and press **Enter**. When Tracert is finished, click the **Stop the running live capture** toolbar icon in Wireshark to stop the capture. Scroll to the first packet summary line, if necessary.
5. Find the ARP packets your computer has generated by looking in the Info column for "Who has *A.B.C.D*, Tell 192.168.100.*XX*" (replacing *A.B.C.D* with the address of your default gateway and *XX* with your student number). Click this packet summary line.
6. Notice that the Dst (for destination) address is ff:ff:ff:ff:ff:ff, indicating a broadcast. In the middle pane, click to expand the **Ethernet II** line. Notice that the Type field is ARP (0x806), which tells the Network access layer which Internetwork-layer protocol should receive the packet. Click again to collapse this line.
7. Click to expand the **Address Resolution Protocol (request)** line. Examine the information in the ARP header. The ARP message has fields to indicate what technology is used in the Network access layer (Ethernet) and the protocol type that needs the MAC address (IP, in this case). Click again to collapse this line.
8. Next, in the top pane, click the ARP reply message immediately following the ARP request. The Info column should be similar to "*A.B.C.D* is at 0A:1B:2C:3D:4E:5F." The MAC address in the ARP reply is the MAC address of your default gateway. Explore the Network access and Internetwork headers for this frame. (*Note*: You might also find an ARP request and ARP reply for your DNS server if it's in the same network as your computer.)
9. In the top pane, click the first **ICMP Echo (ping) request** message from your computer to the destination computer at *books.tomsho.com*. The IP address should be 67.210.126.125, but IP addresses can change, so it might be different.
10. In the middle pane, click to expand the **Internet Protocol** line. Notice that the value in the Time to live line is 1.
11. In the top pane, click the **ICMP Time-to-live exceeded** message that follows the Ping request. This message was generated by the first router en route to *books.tomsho.com*. Notice that the source address is the address of your default gateway.
12. Find the next ICMP Echo (Ping) Request message and view the TTL value. Tracert sends three Echo Request messages for each TTL value, so the first three Echo Request messages have a TTL value of 1. Find the fourth ICMP Echo (Ping) Request message and view the TTL value, which should be 2. The Time-to-live exceeded message following it is from the next router down the line. Tracert follows this pattern until reaching the destination device (*books.tomsho.com*).
13. Exit Wireshark, but leave the command prompt window open if you're continuing to the next project.

Hands-On Project 5-5: Using the `arp` Command

Time Required: 10 minutes

Objective: Use the `arp` command to view and change the ARP cache.

Required Tools/Equipment: Your classroom computer.

Description: In this project, you use the `arp` command to view and then delete the ARP cache, and you use the `ping` command to generate ARP cache entries.

1. If necessary, log on to your computer as **NetAdmin**.
2. Some tasks require opening the command prompt window as an administrator. To do this, click **Start**, point to **All Programs**, **Accessories**, right-click **Command Prompt**, and click **Run as administrator**. In the User Account Control (UAC) message box, click **Yes**.
3. To display your current ARP cache, type **arp -a** and press **Enter**. A list of IP address/MAC address pairs is displayed. The Type field (third column) indicates whether the entry is static or dynamic. Windows 7 generates static entries automatically, but dynamic entries are generated by network communication. If you don't have any entries, the message "No ARP Entries Found" is displayed.
4. To delete the ARP cache, type **arp -d** and press **Enter**. To verify that the entries have been deleted, type **arp -a** and press **Enter** again.
5. Type **ping 192.168.100.*XX*** (replacing *XX* with the IP address of another computer in your network) and press **Enter**. Display your ARP cache again. You should see the IP address you pinged along with its MAC address.
6. Clear the ARP cache again. Type **ping www.course.com** and press **Enter**. Display the ARP cache again. You'll probably see two new entries in your ARP cache. On the following lines, list these two new entries and state why they were generated. Compare the entries in the ARP cache with the IP address of *www.course.com*. Do you see this IP address in the ARP cache? Write your answer along with an explanation.

_____

_____

_____

7. Leave the command prompt window open for the next project.

Hands-On Project 5-6: Using the Netstat Program

Time Required: 10 minutes

Objective: Use the Netstat program to view network interface and IP protocol status and statistics.

Required Tools/Equipment: Your classroom computer

Description: In this project, you use Netstat to view statistics about your network interface and the IP protocol. Then you generate traffic with Ping and Tracert to see the statistics of different packet types change.

1. If necessary, log on to your computer as **NetAdmin** and open a command prompt window.
2. To display statistics about your Ethernet interface, type **netstat -e** and press **Enter**. These statistics include the number of bytes and packets received and sent through the Ethernet interface. If any errors are indicated in the display, you might have problems with your network connection that are slowing the network down. If the error packets approach 1% of the total number of packets, something is probably wrong with your NIC or physical interface.
3. To see statistics for all protocols, type **netstat -s** and press **Enter**. To limit the display to just IP statistics, type **netstat -ps IP** and press **Enter**.
4. To see your network statistics updated every 5 seconds, type **netstat -ps IP 5** and press **Enter**. Press **Ctrl+C** to stop the program.
5. To display ICMP information, type **netstat -ps ICMP** and press **Enter**. A variety of ICMP message types are displayed along with how many of each type of message were received and sent. Most, if not all, will be Echo and Echo Reply messages.
6. Type **ping 5.5.5.5** and press **Enter**. This command should generate ICMP Destination Unreachable messages. To see whether the number of Destination Unreachable messages has increased, type **netstat -ps ICMP** and press **Enter**.
7. The ICMP TTL-Expired messages used in Tracert are called Time Exceeded messages in Netstat. Type **tracert books.tomsho.com** and press **Enter**. To see whether the number of Time Exceeded messages has increased, type **netstat -ps ICMP** and press **Enter**.
8. To display your computer's routing table, type **netstat -r** and press **Enter**. Every computer has a routing table it uses to decide which interface to send packets to. The first entry lists the network destination as 0.0.0.0, which is the entry for your default gateway.
9. Leave the command prompt window open for the next project.

Role of the Transport Layer

Without the Transport layer in the TCP/IP protocol suite, large internetworks would be in big trouble. So many things can go wrong with complex, constantly changing networks that without some reliability measures, successful transfers of large amounts of data would be the exception rather than the norm. In environments such as the Internet, using only connectionless protocols simply wouldn't work. The more robust protocols in the Transport layer provide the reliability needed to handle the unpredictable nature of the Internet (or any large internetwork, for that matter).

The Transport layer has two protocols. **Transmission Control Protocol (TCP)** is connection oriented and designed for reliable transfer of information in complex internetworks. **User Datagram Protocol (UDP)** is connectionless and designed for efficient communication of generally small amounts of data. Both protocols perform the following tasks:

- Work with segments.
- Provide a means to identify the source and destination applications involved in a communication.
- Protect data in the segment with a checksum.

Working with Segments Both Transport-layer protocols work with units of data called segments. For outgoing data, in which the Application-layer protocol requires the services of a Transport-layer protocol, the Application layer passes data to TCP or UDP, depending on which protocol it was designed to use. Both TCP and UDP add a header to the data to make it a segment. The Transport-layer protocol then passes the segment to the Internetwork-layer protocol, usually IP. With incoming data, the Internetwork-layer protocol deencapsulates the packet and forwards the resulting segment to the Transport-layer protocol. The Transport-layer protocol processes the segment, deencapsulates it, and sends the resulting data up to the Application layer.

Identifying Source and Destination Applications Have you ever wondered how your computer keeps track of the myriad network applications you run? At any time, you might be running a Web browser, an e-mail application, and a chat program and have a file open on a file server. When one of these applications receives data from the network, a frame is received by the NIC, which sends a packet up to the IP protocol, which then sends a segment to TCP or UDP. Now what? Eventually, data that's received must go to an application or a network service.

The Transport-layer header provides the information needed to determine the application the received data is sent to. TCP and UDP use **port numbers** to specify the source and destination Application-layer protocols. Using an envelope analogy, if the IP address is the zip code and the street number is the MAC address, the port number specifies the person in the house who should read the letter. In other words, the MAC address and IP address get the packet to the computer, and the port number gets the data to the application or service.

The IANA assigns a dedicated port number to every well-known network service. For example, Web servers are assigned port 80, so when your computer formats a message to a Web server, the destination port number in the TCP header is 80. Likewise, when your e-mail application requests messages from your mail server, it sends the request to port 110, the Post Office Protocol (POP3) port number. Most client applications are assigned a random port number when they make a request to a server. So when you start a Web browser, for example, the Web browser window is assigned a port number. When the request for a Web page goes out, the source port number in the TCP header contains the number assigned to that Web browser window so that the Web server knows to which port the reply should be sent. If you open another Web browser window or tab, another port number is assigned, and so forth. The port number is a 16-bit value, so you can open as many as 65,000 windows!

You can see the list of well-known port numbers at *www.iana.org/assignments/port-numbers*.

An IP application that doesn't use a Transport-layer protocol, such as the Ping program and routing protocols, can rely on the Internetwork layer to provide application information. As you've seen, the IP packet header includes the Protocol field for just this purpose.

Protecting Data with a Checksum To protect data integrity, TCP and UDP provide a checksum similar to the CRC in the Network access layer, but the CRC isn't always a perfect mechanism for ensuring that data wasn't corrupted on the way to its destination. Routers and switches have been known to corrupt data, recalculate the CRC code, and send the corrupted data on its way. In this situation, the receiver has no way of knowing the data was corrupted because the CRC was calculated after the corruption. Intermediate devices don't recalculate the checksum in the Transport layer, so if data corruption occurs along the way, the final receiving station detects the checksum error and discards the data. To ensure reliability, calculating a checksum is as far as UDP goes. All other reliability features at the Transport layer are the domain of TCP.

TCP: The Reliable Transport Layer

If an application requires reliable data transfer, it use TCP as the Transport-layer protocol. TCP provides reliability with the following features that aren't available in UDP:

- Establishing a connection
- Segmenting large chunks of data
- Ensuring flow control with acknowledgements

Each feature relies on TCP being a connection-oriented protocol. TCP establishes a connection with the destination, data is transferred, and the connection is broken.

Establishing a Connection: The TCP Handshake Establishing a connection with TCP is similar to making a phone call. You dial the number and wait for your party to answer, usually with a greeting. The caller then states his or her name and says who he or she wants to talk to. If everything is agreeable, a conversation begins.

A TCP session begins when a client sends a TCP synchronization (SYN) segment to the destination device, usually a server. A destination port number (typically a well-known port, such as 80) is specified, and a source port number is assigned dynamically. When the server receives the SYN segment, it usually responds by sending one of two segments: an acknowledgement-synchronization (ACK-SYN) segment or a reset connection (RST) segment. If an RST segment is returned, the server refused the request to open a session, possibly because the destination port is unknown. If an ACK-SYN segment is returned, the client completes the **three-way handshake** by sending an ACK segment back to the server. The client is then ready to begin sending or requesting data. You capture and examine a three-way handshake in Challenge Lab 5-1.

Segmenting Data One safeguard TCP provides is segmenting data before sending it to the Internetwork layer and reassembling the data at the destination before sending it up to the Application layer. When TCP receives data from the Application layer, the size of the data might be too large to send to the Internetwork layer in one piece. Remember that Ethernet can send only frames that are a maximum of 1518 bytes. It's TCP's job to break the data into smaller segments before handing each segment to the Internetwork layer. Each segment is labeled with a sequence number so that if segments arrive at the destination out of order, they can be reassembled in the correct order by using the sequence number. Programs that work with large amounts of data, such as Web browsers and file transfer programs, use Application-layer protocols that work with TCP for this reason. Applications

that work with small amounts of data can use UDP, which doesn't disassemble or reassemble data.

Ensuring Flow Control with Acknowledgements Another role of TCP is to provide **flow control**, which prevents a destination from becoming overwhelmed by data, resulting in dropped packets. TCP does this by establishing a maximum number of bytes, called the window size, that can be sent before the destination must acknowledge receipt of the data. If a sending machine hasn't received an acknowledgement before sending the number of bytes established by the window size, it stops sending data. If no acknowledgement is received in a specified timeout period, the sender retransmits the data from the point at which an acknowledgement was last received.

Role of the Application Layer

The Application layer provides network services to user applications that access network resources. For example, when you run Microsoft Word and need to open a file on a network server, Word contacts Client for Microsoft Networks, an Application-layer service, which provides the details of accessing files on the server. Client for Microsoft Networks implements an Application-layer protocol called Server Message Block (SMB), which is also known as Common Internet File System (CIFS). Linux uses NFS and Samba file-sharing Application-layer protocols.

In some cases, the Application-layer protocol or service is built into the user application, as with a Web browser or e-mail client. For example, a Web browser contains the software that implements Hypertext Transfer Protocol (HTTP). Whether the Application-layer protocol is implemented by the user application or by a network service, the process is the same: When data is ready to be sent, it's transferred from the Application-layer protocol to the Transport layer and down the protocol stack until a frame is transmitted as bits to the network medium.

Application-layer protocols also provide authentication and data-formatting services as needed. For example, if a client attempts to access a server that's password protected, the Application layer is responsible for handling the exchange of packets that allow user logon. If data needs to be formatted or translated in some way for the user application, as with some types of data encryption, the Application layer provides that service for user applications. For example, when you connect to a secure Web site with HTTPS, the authentication and encryption that occur with HTTPS are Application-layer functions.

Some functions of the TCP/IP model's Application layer are separated into additional layers in the OSI model discussed in Chapter 6. For example, the Session layer handles network logon, and the Presentation layer handles data encryption and decryption.

With most Application-layer protocols, both a client and a server version exist. For HTTP, the client is a Web browser and the server is a Web server, such as Microsoft Internet Information Services (IIS) or the popular Apache Web Server that's often used on Linux servers. Client for Microsoft Networks has File and Printer Sharing for Microsoft Networks as its server counterpart.

Most Application-layer protocols facilitate a client's access to data, such as an e-mail message or a document. However, the Application layer contains some specialized protocols for

making a network easier to use and configure. Examples include protocols for name resolution and dynamic IP address assignment. Several Application-layer protocols are discussed in more detail in the next sections, but to sum up, the Application layer provides these functions:

- Access by applications to network services
- Client/server data access
- Name resolution
- Dynamic address assignment
- Authentication/user logon
- Data formatting and translation

HTTP: Protocol of the World Wide Web HTTP is the protocol Web browsers use to access data on the World Wide Web. Originally, its main purpose was simply to transfer static Web pages written in HTML. Now HTTP is also used for general file transfer and downloading and displaying multimedia files. Because it's often used to transfer large amounts of data over the Internet, it uses TCP as its Transport-layer protocol, and the default TCP port number is 80. Figure 5-8 shows a typical HTTP message as it might look at the Internetwork layer before going to the Network access layer to be framed. The TCP header contains important source and destination port numbers.

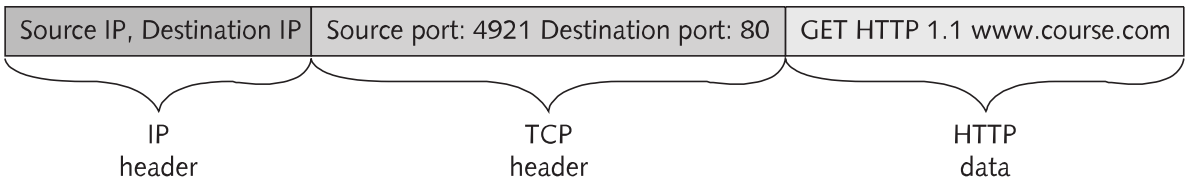

Figure 5-8 An HTTP message

Courtesy of Course Technology/Cengage Learning

POP3, IMAP, and SMTP: E-mail Protocols E-mail clients use the **Post Office Protocol version 3 (POP3)** protocol to download incoming messages from an e-mail server to their local desktops. POP3 clients download e-mail from the mail server running at the user's ISP, and these message are then deleted from the server. POP3 uses TCP port 110.

Internet Message Access Protocol (IMAP) has advanced message controls, including the capability to manage messages locally yet store them on a server, plus numerous fault-tolerance features. IMAP downloads only e-mail headers initially, and then downloads the message body and attachments when the message is selected. IMAP uses TCP port 143.

Simple Mail Transfer Protocol (SMTP) is the standard protocol for sending e-mail over the Internet. POP3 is used to retrieve e-mail, and SMTP is used to send it. SMTP uses TCP port 25. All three e-mail protocols use the TCP Transport-layer protocol to ensure reliable delivery of large messages.

Dynamic Host Configuration Protocol Some drawbacks of using TCP/IP in a large network include detailed configuration of devices and keeping track of assigned

addresses and to which machine they're assigned. To make these tasks easier, **Dynamic Host Configuration Protocol (DHCP)** was developed.

To use DHCP, a server must be configured with a block of available IP addresses and other IP address configuration information. To receive its IP address from the server, each computer must be configured to request its address configuration. A computer requests IP address information from the DHCP server in the form of a broadcast message. Each time a computer requests an address, the server assigns one until it has no more addresses to assign.

A computer just leases the address the server assigns to it. The network administrator defines the lease time when the DHCP server is configured. This time can be as little as a few minutes to an infinite period, in which case the lease never expires. A typical lease time is one day or a few days. When 50% of the lease time has elapsed, the computer attempts to renew the lease from the same DHCP server that responded to the initial DHCP request. If no response is received, the computer waits until 87.5% of the lease time has elapsed; at that point, a broadcast DHCP renewal request is sent. If no response has been received when the lease expires, the computer broadcasts a DHCP request for a new IP address. If no DHCP server responds, one of two things happens: TCP/IP stops functioning, or the computer assigns itself an address from a special range of addresses beginning with 169.254.

These special addresses are reserved for **Automatic Private IP Addressing (APIPA)**. An address in the APIPA range is assigned automatically to an APIPA-enabled computer when an IP address is requested via DHCP but no DHCP server responds to the request. Using APIPA rather than a DHCP server to assign addresses is recommended only for small networks that aren't attached to the Internet.

A major benefit of using DHCP is how easily computers can be moved. When a computer is moved to a new network segment and turned on, it requests its configuration from a DHCP server on that segment. This type of address assignment shouldn't be used for systems that require a static address, such as Web servers, DNS servers, and DHCP servers, because computers with these network services are usually expected to maintain the same IP address.

DHCP uses the UDP Transport-layer protocol because DHCP servers are usually located on the same network as the DHCP client, and DHCP messages are short in length. Recall that UDP is a connectionless protocol and provides few reliability features, so it works best when the amount of data in each transaction is small.

All major OSs include a DHCP client service, and most server OSs and routers include the DHCP server component.

Domain Name System **Domain Name System (DNS)** is a name-to-address resolution protocol that keeps a list of computer names and their IP addresses. Through a correctly configured workstation, a user can use a computer's name—for instance, Server1 or www.course.com—rather than a numerical address, such as 207.46.134.189, to communicate with the computer. For example, when you enter "www.course.com" in your Web browser's address box, the Web browser contacts the DNS client service on your computer. The

DNS client contacts the DNS server specified in your OS's IP configuration and requests that the name "www.course.com" be resolved to an IP address. The DNS server responds with the IP address assigned to the computer named www at the course.com domain. From there, using the IP address returned, your Web browser application can contact the Web server to request a Web page.

DNS uses the UDP Transport-layer protocol because DNS messages usually consist of a single packet of data, so there's no need for the reliability measures TCP offers. The DNS system used throughout the Internet is organized as a treelike hierarchy (see Figure 5-9). The tree consists of these domain levels: root, top, second, subdomain, and host. All levels below the root level have branches, each of which has a name. When you put all the names of a branch together, separated by a period, you have the **fully qualified domain name (FQDN)** of the network resource, such as *www.course.com*.

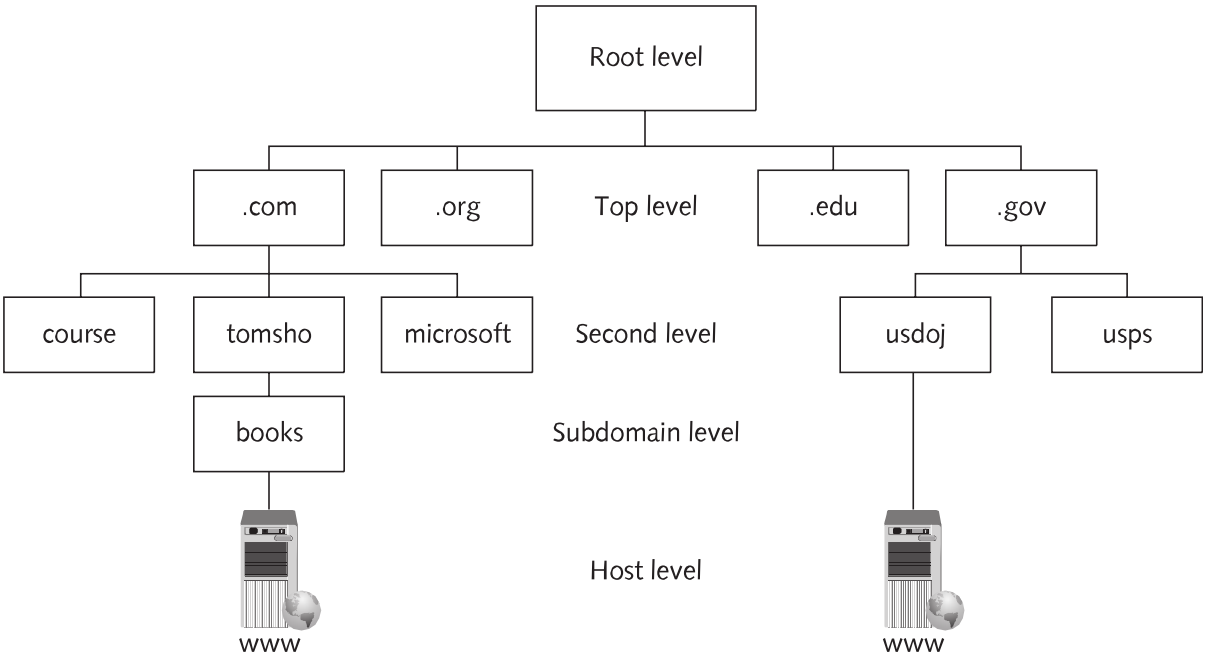

Figure 5-9 DNS hierarchical tree structure

Courtesy of Course Technology/Cengage Learning

The top-level domains are organized into categories—such as commercial (.com), nonprofit organizations (.org), government (.gov), and education (.edu)—or country of origin, indicated by a two-letter country code. The second-level domains are usually the name of a company or an institution. The subdomain level is optional and can consist of several names separated by a period. An example is a department or branch of an organization. Finally, the host level represents individual computers hosting network services. For example, in *www.books.tomsho.com*, com is the top-level domain name, tomsho is the second-level domain, books is the subdomain, and www is the hostname.

Because of the hierarchical nature of DNS, not every DNS server needs to maintain a database of all domain names, computer names, and IP addresses for the entire Internet. Most DNS servers maintain addresses for the domain in which they're installed. The domain might be a single secondary-level domain, such as xyzcorp.com, or if you own a business

hosting Web sites for other companies, you might maintain hundreds or thousands of domains, but that number is small compared with the entire Internet.

When a DNS server is installed, the administrator creates one or more domain names or zones. A zone is named by using the second-level and top-level domain names and the subdomain, if necessary. Most of the information a DNS zone contains consists of hostname/IP address pairs, called host records. An administrator can create host records manually, and they can be created dynamically with Dynamic DNS (DDNS), which enables a computer to create its own DNS record.

In addition to host records, a DNS server database is loaded with a list of IP addresses that point to root servers around the world. These servers supply the addresses of top-level domain servers, which are used to provide addresses of second-level domain servers. This hierarchical organization allows any DNS client anywhere in the world to access the DNS servers for any domain.

You can view a map of the root servers around the world at *http://public-root.com/root-server-locations.htm*.

To speed up communication, DNS clients in most OSs maintain a DNS cache, much as ARP maintains an ARP cache. The DNS cache stores name and IP address pairs along with other pertinent DNS data for names that have been resolved recently. This cache prevents your DNS client from having to request that a DNS server do a name lookup for a name that was resolved recently. Additionally, the cache contains a text file called Hosts, which stores name and IP address pairs. This file usually contains only the name "localhost" mapped to 127.0.0.1, but you can add entries manually by editing the file. If there are computers you access frequently by name and you don't expect their addresses to change, you can add an entry for these computers in the Hosts file, thereby preventing a network DNS lookup from occurring when you access them. You examine the Hosts file in Hands-On Project 5-7.

Many other Application-layer protocols work with TCP or UDP, but the protocols discussed in this chapter cover the most commonly used ones in most networks. For all these protocols to work, TCP/IP needs a correctly addressed network, so in the next section, you turn your attention to IP addressing.

Hands-On Project 5-7: Working with DNS Tools

Time Required: 10 minutes

Objective: Use Ipconfig and Nslookup to work with DNS.

Required Tools/Equipment: Your classroom computer

Description: In this project, you use Ipconfig to display and delete your DNS cache, and then view your Hosts file. You also use Nslookup to query your DNS server.

1. If necessary, log on to your computer as **NetAdmin** and open a command prompt window.

2. To see a list of recent DNS lookups, type **ipconfig /displaydns** and press **Enter**. To delete the entries, type **ipconfig /flushdns** and press **Enter**. Display the DNS cache again. Unless there are entries in your Hosts file, you should get the message "Could not display the DNS Resolver Cache."

At the command prompt, you can press the up and down arrow keys to access recent commands you have entered.

3. To perform a DNS lookup, type **ping www.course.com** and press **Enter**. Display the DNS cache again.
4. You should see a DNS record for *www.course.com* that includes the IP address and other information. Another field in the DNS cache is a TTL value, which is different from the TTL in an IP packet. This DNS TTL value is sent by the DNS server maintaining the *www.course.com* record. It's measured in seconds and tells your DNS client how long to cache the DNS record as a safeguard against clients holding on to DNS records whose IP addresses might have changed.
5. To open your computer's Hosts file, click **Start**, point to **All Programs**, **Accessories**, right-click **Notepad**, and click **Run as administrator**. In the UAC message box, click **Yes**. Click **File**, **Open** from the Notepad menu. In the Open dialog box, navigate to **C:\Windows\System32\Drivers\Etc**. In the File type drop-down list, click **All Files**. Double-click the **hosts** file to open it.
6. After the last line in the file, type **67.210.126.125 books**, and then save the file and exit Notepad.
7. At the command prompt, type **ipconfig /displaydns** and press **Enter** to see that the entry is in your DNS cache. Type **ping books** and press **Enter**. Delete the DNS cache (see Step 2), and then display the DNS cache again. Notice that the books entry remains in the cache because the Hosts file data always stays in the cache.
8. Type **nslookup www.course.com** and press **Enter**. Your DNS server's name and IP address are displayed, along with the name and IP address of *www.course.com*. You use Nslookup to look up a host's IP address without actually communicating with it.
9. Type **nslookup** and press **Enter**. You enter Nslookup's interactive mode. Type **www.google.com** and press **Enter**. Notice that more than one address is returned along with one or more aliases (other names that *www.google.com* goes by). Type **www.google.com** again (or press the up arrow to repeat the last line you typed) and press **Enter**. You should see the IP addresses returned in a different order. (If you don't, keep trying, and the order will change.) The *www.google.com* page can be reached by a number of different IP addresses, and the addresses are returned in a different order so that a different server is used each time, which is called load balancing.
10. Type **198.60.123.100** and press **Enter**. Nslookup is also used to do reverse lookups, in which the IP address is given and the hostname is returned.
11. To set the DNS server that Nslookup uses to a public DNS server run by Google, type **server 8.8.8.8** and press **Enter**. Type **www.microsoft.com** and press **Enter**. If you're ever concerned that your DNS server isn't working correctly, you can test it with Nslookup

and compare the results of your DNS server with the results from another server, such as Google's.

12. Leave the command prompt window open for the next project.

Hands-On Project 5-8: Working with the DHCP Client

Time Required: 10 minutes

Objective: Use Ipconfig to work with your DHCP client.

Required Tools/Equipment: Your classroom computer

Description: In this project, you change your IP settings to use DHCP and then see how to work with DHCP by using Ipconfig.

1. If necessary, log on to your computer as **NetAdmin**.
2. Open the Network and Sharing Center by clicking **Start, Control Panel**. Under Network and Internet, click **View network status and tasks**.
3. In the left pane of the Network and Sharing Center, click **Change adapter settings**. Right-click **Local Area Connection** and click **Properties** to open the Local Area Connection Properties dialog box.
4. Click **Internet Protocol Version 4 (TCP/IPv4)** and click **Properties**. If your IP address settings were set manually in Hands-On Project 5-1, write these settings on the following lines. At the end of this project, you set your IP address again by using these settings.

_____

_____

5. Click **Obtain an IP address automatically.** Click **OK** and then **Close**.
6. Open a command prompt window, if necessary. Type **ipconfig /all** and press **Enter** to view detailed IP configuration information. Under Ethernet adapter Local Area Connection, you see information about DHCP, including its status (enabled or not), the DHCP server's IP address, and lease information.
7. Occasionally, you might need to force your computer to renew its lease (for example, if changes are made on the DHCP server and you need to get the newest IP configuration). To renew a DHCP lease, type **ipconfig /renew** and press **Enter**. Display your detailed IP configuration again to see that the lease information has changed.
8. To release your IP address configuration, type **ipconfig /release** and press **Enter**. This command's output shows that your IP configuration has been deleted. To request a new IP address configuration, type **ipconfig /renew** and press **Enter**. (Note that you might not get the same IP address you had before.) Using these commands can help you troubleshoot DHCP-related problems.
9. Close the command prompt window, and set your IP configuration to the values you wrote down in Step 4. Stay logged on for the challenge labs at the end of the chapter.

IP Addressing

As you've learned, IP is responsible for addressing and routing in the TCP/IP environment. IP addresses are 32-bit (4-byte) logical addresses. The 32 bits are grouped into four 8-bit **octets**, and each octet is represented by a decimal number from 0 to 255. The four decimal numbers are separated by periods in a format called **dotted decimal notation**, as in 172.24.208.192.

As discussed, an IP address is divided into two distinct parts. One part designates which logical network the computer is a part of; the remainder of the address represents the host ID for that computer. For example, a computer with the address 172.24.208.192 resides on the 172.24 network, and its host ID is 208.192. In this case, the complete network address is expressed as 172.24.0.0, with the trailing zeros indicating a network address because a host ID can't be 0. The computer next to it might have the address 172.24.18.26, but even though their host IDs are quite different, both computers are on the same network because they share the same network address (172.24).

You can determine how many host addresses are in a network (the **address space**) by looking at the host ID's size. In the address 172.24.208.192, for example, the host ID occupies the third and fourth octets of the address, which allows 16 bits for the address space. With 16 bits of address space, you can use the formula 2^{16}, which yields 65,536. An address with a network number of 201.55.66 leaves only one octet (or 8 bits) for the host ID, or 2^{8}, which yields 256 possible addresses.

IP Address Classes

IP addresses are categorized in ranges referred to as Classes A, B, C, D, or E. Only IP addresses in the A, B, and C classes are available for host assignment. Although the IP address class system has been somewhat superseded by a more flexible way to manage IP addresses, called Classless Interdomain Routing (CIDR, discussed later in this chapter in "Classless Interdomain Routing"), the class system is a basis for determining which part of an IP address is the network ID and which part is the host ID. The first octet of an address denotes its class. Note the following facts about IP address classes:

- The value of the first octet for Class A addresses is between 1 and 127. Class A addresses are intended for use by large corporations and governments. An IP address registry assigns the first octet, leaving the last three octets for network administrators to assign to hosts. This allows 24 bits of address space or 16,777,214 hosts per network address. In a Class A IP address such as 21.155.49.211, for example, the network ID is 21.0.0. So the first address in the 21.0.0.0 network is 21.0.0.1, and the last address is 21.255.255.254.
- Class B addresses begin with network IDs between 128 and 191 and are intended for use in medium to large networks. An IP address registry assigns the first two octets, leaving the third and fourth octets available for administrators to assign as host addresses. In the Class B address 172.17.11.4, for example, the network ID is 172.17.0. Having two octets in the host ID allows 65,534 hosts per network address.
- Class C addresses are intended for small networks. An IP address registry assigns the first three octets, ranging from 192 to 223. In the Class C address 211.255.49.254, for example, the network ID is 211.255.49. These networks are limited to 254 hosts per network.

- Class D addresses are reserved for multicasting, in which a packet is addressed so that more than one destination can receive it. Applications using this feature include videoconferencing and streaming media. In a Class D address, the first octet is in the range 224 to 239. Class D addresses can't be used to assign IP addresses to host computers.
- Class E addresses have a value from 240 to 255 in the first octet. This range of addresses is reserved for experimental use and can't be used for address assignment.

A couple of notes about this list: First, if you did your math, you would see that a Class C address provides 2^8 bits of address space, which yields 256 addresses, not 254. Note, too, that the number of addresses specified for Classes A, B, and C are two fewer than the address space suggests. This discrepancy happens because each network has two reserved addresses: the address in which all host ID bits are binary 0s and the address in which all host ID bits are binary 1s. For example, all the host bits in address 198.44.19.0 are binary 0s, and this address represents the network number and can't be assigned to a computer. The host bits in address 198.44.19.255 are binary 1s; this address is the broadcast address for the 198.44.19.0 network and can't be assigned to a computer.

The other note concerns the 127.0.0.0 network. Although technically a Class A address, it's reserved for the **loopback address**, which always refers to the local computer and is used to test the functioning of TCP/IP. A packet with a destination address starting with 127 is sent to the local device without reaching the network medium. Likewise, the reserved name **localhost** always corresponds to the IP address 127.0.0.1 so that a local machine can always be referenced by this name.

Even though localhost and the loopback address are usually associated with the address 127.0.0.1, any address in the 127.0.0.0 network (except 127.0.0.0 and 127.255.255.255 in most OSs) references the local machine.

Private IP Addresses

Because of the popularity of TCP/IP and the Internet, unique IP addresses to assign to Internet-accessible devices are almost exhausted. To help alleviate this problem, TCP/IP's technical governing body reserved a series of addresses for private networks—that is, networks whose hosts can't be accessed directly through the Internet. This nonprofit governing body, the Internet Engineering Task Force (IETF; *www.ietf.org*), is responsible for TCP/IP standards and characteristics. The reserved addresses are as follows:

- Class A addresses beginning with 10 (one Class A network address)
- Class B addresses from 172.16 to 172.31 (16 Class B network addresses)
- Class C addresses from 192.168.0 to 192.168.255 (256 Class C network addresses)

The addresses in these ranges can't be routed across the Internet, which is why any organization can use them to assign IP addresses to their internal hosts. If access to the Internet is necessary, a process called Network Address Translation (NAT) is used, explained next.

IPv6 eliminates the need for private addressing because it provides a 128-bit address space, compared with IPv4's mere 32 bits. You learn more about IPv6 later in this chapter in "Introduction to Internet Protocol Version 6."

Several IP address registries around the world cooperatively manage the total collection of valid IP addresses. Their activities are overseen by the Internet Assigned Numbers Authority (IANA), a nonprofit agency responsible for Internet addressing and address management.

Network Address Translation

Although subnetting can alleviate the IP address shortage problem, as you learn later in "Subnet Masks," it simply makes more efficient use of existing addresses. **Network Address Translation (NAT)** helps more by allowing an organization to use private IP addresses while connected to the Internet. Recall that there are three ranges of private IP addresses (one range for each class), and these addresses can't be used as the source or destination address in a packet on the Internet.

Anyone can use private IP addresses for address assignment to internal computers and devices, and because the addresses aren't sent to the Internet, there's no address conflict. What if you want your computers to have access to the Internet, however? That's where NAT comes in. An organization can, for example, assign all its workstations' addresses in the 10.x.x.x private network. Say an organization has 1000 workstations. Although these addresses can't be used on the Internet, the NAT process translates a workstation address (as a packet leaves the corporate network) into a valid public Internet address. When data returns to the workstation, the address is translated back to the original 10.x.x.x address. NAT is usually handled by a network device that connects the organization to the Internet, such as a router. As shown in Figure 5-10, when station 10.0.0.1 sends a packet to the Internet, the NAT router intercepts the packet and replaces its source address with 198.60.123.101 (a public Internet address). When a reply comes back addressed to 198.60.123.101, the NAT router replaces the destination address with 10.0.0.1.

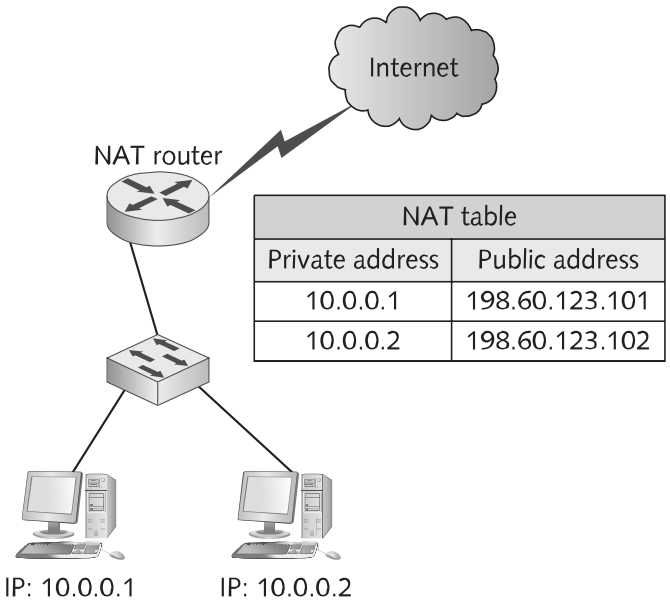

Figure 5-10 Private addresses are translated to public addresses with NAT

Courtesy of Course Technology/Cengage Learning

This process allows any number of companies to use private IP addresses in their own network but requires a public IP address only when a workstation attempts to access the Internet. NAT reduces the number of public IP addresses needed because a public address is required only if a computer accesses the Internet. NAT does have a drawback, in that one public address is required for every private address. However, it's usually used only for Web servers and other devices that must be accessed through the Internet.

An extension of NAT, called **Port Address Translation (PAT)**, allows several hundred workstations to access the Internet with a single public Internet address. This process relies on each packet containing not only source and destination IP addresses, but also source and destination TCP or UDP port numbers. With PAT, the address is translated into a single public IP address for all workstations, but a different source port number (which can be any value from 1024 to 65,535) is used for each communication session, allowing a NAT device to differentiate between workstations. The typical router used in home and small business networks is already configured to use PAT.

Figure 5-11 shows an example of how PAT is used. Notice that the public address is the same for both entries; only the port number differs. When an Internet server responds to 198.60.123.100 on port 3105, however, the router knows to translate the destination address in the packet to 10.0.0.2 port 12441. Notice also that the public address in the NAT/PAT table is the same as the router's Internet-connected interface. Although this configuration isn't necessary, it's common in home/small office routers. In Simulation 10, you see an animation of how PAT works.

NAT/PAT table	
Private address: Port	Public address: Port
10.0.0.1:2562	198.60.123.100:5311
10.0.0.2:12441	198.60.123.100:3105

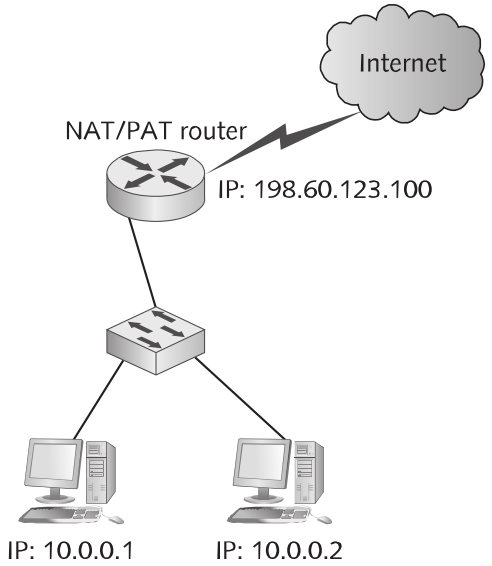

Figure 5-11 PAT uses the port number to allow using a single public IP address

Courtesy of Course Technology/Cengage Learning

Simulation 10: Demonstrating NAT/PAT

For an excellent tutorial on NAT, see *www.howstuffworks.com/nat.htm*.

Classless Interdomain Routing

As mentioned, addressing by class has been superseded by a more flexible addressing method. To use all available addresses more efficiently, a different addressing scheme called **Classless Interdomain Routing (CIDR)** is now used on the Internet. With this scheme, when an address is assigned, the network and host IDs don't always begin and end on octet boundaries according to the IP address class; instead, the network ID and host ID can be almost any number of bits (with the combined number of bits equaling 32, of course).

For example, a Class C address's network ID is 24 bits, and its host ID is 8 bits. Using CIDR, an address registry can assign an address with a network ID of 26 bits, leaving 6 bits for the host ID. This technique involves reallocating bits from the host portion of the address to create two or more network numbers. Put another way, one larger network is broken into two or more smaller networks, or **subnets**. **Subnetting** is the process of dividing a single network address into two or more subnetwork addresses, each with fewer available host IDs than the original network address. Subnetting allows fewer hosts on each network but results in more networks overall.

When CIDR addresses are assigned, a slash denotes the number of bits in the network ID; this format is called "CIDR notation." For example, if your company requires only 30 host addresses to assign to computers attached to the Internet, your ISP might give you the network address 192.203.187.0/27. In a 32-bit IP address, the /27 means the first 27 bits of the IP address designate the network ID, and the last 5 bits designate the host ID. The /27 following the IP address is also referred to as the **IP prefix** (or just "prefix"). CIDR notation is used in documentation and by some systems to specify IP addresses. However, the IP address configuration in most OSs uses a second dotted decimal number called a subnet mask, discussed next.

Subnet Masks

In the early days of IP, a host used the address class to determine which part of the address was its network ID. However, this method doesn't offer the flexibility that CIDR addressing requires. When an IP address is assigned to a computer or other IP device, it's always accompanied by a subnet mask. IP uses an address's **subnet mask** to determine which part of the address denotes the network portion and which part denotes the host. It's a 32-bit number in dotted decimal format consisting of a string of eight or more binary 1s followed by a string of 0s. A binary 1 in the subnet mask signifies that the corresponding bit in the IP address belongs to the network address, and a binary 0 signifies that the corresponding bit in the IP address belongs to the host ID.

Each of the three main address classes has a default subnet mask that uses the decimal number 255 for each octet corresponding to the network ID. (The number 255 is 11111111 in binary and fills all eight bit positions in an IP address octet.) Therefore, the default Class A subnet mask is 255.0.0.0, the default Class B subnet mask is 255.255.0.0, and the default Class C subnet mask is 255.255.255.0.

For example, if a computer has the IP address 153.92.100.10 and the subnet mask 255.255.0.0 (a Class B mask), the network portion is 153.92 and the host portion is 100.10. Using the same address of 153.92.100.10 but with the subnet mask 255.255.255.0, the network portion is now 153.92.100 and the host portion is 10. By altering the subnet mask, the network has been changed from one with 65,534 host addresses to one with only 254 host addresses. At first glance, diminishing the size of the address space might not seem like a good thing. However, if you consider how IANA assigns IP addresses to companies and ISPs, it makes more sense.

For example, a new business expects to have a large presence on the Internet with Web servers, DNS servers, application servers, routers, and so forth. This business contracts with an ISP for its Internet access, and the ISP allocates IP addresses the business can use. The business calculates that eventually about 400 addresses will be needed for its Web servers and other devices. The ISP has no Class C addresses to allocate but does have the Class B network 153.92.0.0; however, a Class B address provides 65,534 addresses! If the ISP allocated the company the IP address 153.92.0.0 with the 255.255.0.0 Class B subnet mask, more than 65,000 addresses would be wasted. After a network address is used in one network, no other network can use any addresses in that network. One solution is for the ISP to allocate the company two network addresses: 153.92.0.0 and 153.92.1.0, both with subnet mask 255.255.255.0. That gives the company two network addresses, each with 254 possible host addresses; more than enough for its needs. This solution leaves network addresses 153.92.2.0/24 through 153.92.255.0/24 available for assignment to other networks. (Remember that in CIDR notation, /24 means a 255.255.255.0 subnet mask.)

All devices on a single logical network (meaning each device can communicate with another device without going through a router—sometimes called a network segment) must share the same network address and, therefore, use the same subnet mask.

Table 5-1 summarizes the three classes of IP address used for IP host assignment.

Table 5-1 IP address class summary

Value of first octet	Class	Default subnet mask	Number of hosts/network
1–127	A	255.0.0.0	16,777,214
128–191	B	255.255.0.0	65,534
192–223	C	255.255.255.0	254

How Is the Subnet Mask Used? The IP address configuration of a computer that's part of an internetwork or has access to the Internet requires at least three elements: IP address, subnet mask, and default gateway. As discussed in Chapter 2, the default gateway is the IP address of the router to which the computer sends packets destined for a different

network. The default gateway address must be in the same network as the computer's IP address.

So how does a computer know from the address whether the destination computer is on a different network? Is the address 172.19.44.211 on a different network from 172.19.46.188? The only way to know is by consulting the subnet mask. Using these two addresses, take a look at the sample network shown in Figure 5-12.

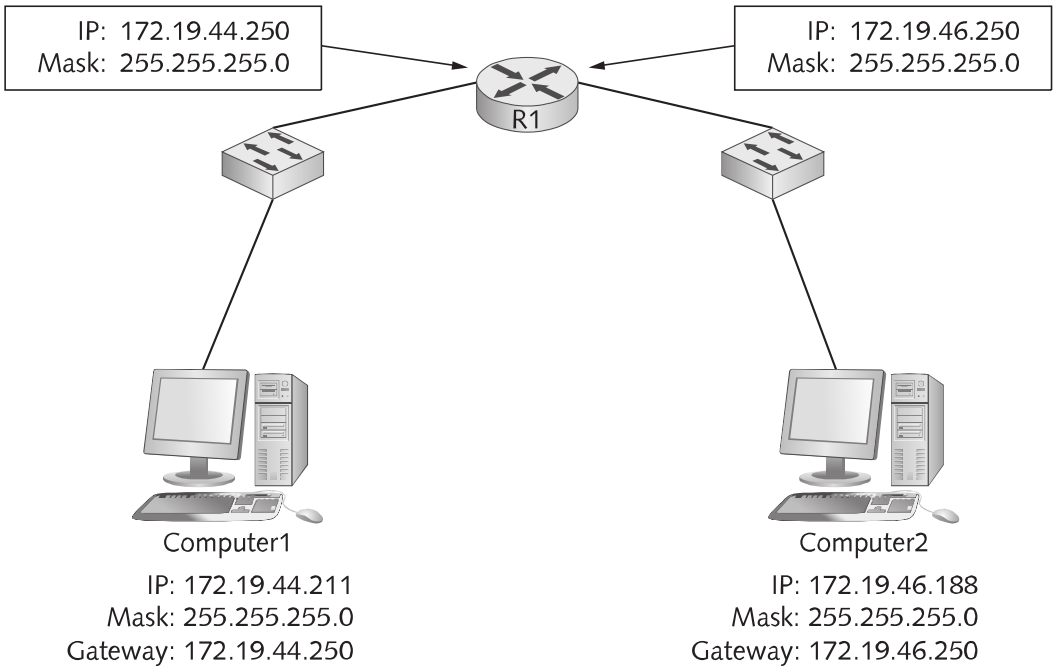

Figure 5-12 Determine the destination computer's network address with the subnet mask

Courtesy of Course Technology/Cengage Learning

Here's what happens when Computer1 has a packet to send to Computer2:

1. Computer1 must first know its network address. It determines this by doing a logical AND operation between its IP address and subnet mask. A logical AND is an operation between two binary values. AND operations can have the following results:

```
0 AND 0 = 0
1 AND 0 = 0
0 AND 1 = 0
1 AND 1 = 1
```

2. The logical AND operation between Computer1's IP address and subnet mask looks like this:

```
10101100.00010011.00101110.10111100 (binary for 172.19.44.211)
                  AND
11111111.11111111.11111111.00000000 (binary for 255.255.255.0)
-----------------------------------
10101100.00010011.00101110.00000000 (binary for 172.19.44.0)
```

   The resulting network address is 172.19.44.0.

3. The next step is to determine whether Computer2's address is on the same network or a different network. The same AND calculation is done between Computer2's IP address and Computer1's subnet mask. (Computer1 has no way of knowing Computer2's subnet mask.) The resulting network address is 172.19.46.0.
4. Because Computer2 is on a different network, Computer1 knows that the packet must be sent to the router, which forwards it to Computer2's network.

Why Subnet? An IP network or subnetwork can be defined as a group of computers and devices that share the same network portion of their assigned IP addresses and don't have to go through a router to communicate with one another. Another term for an IP network is broadcast domain, explained in Chapter 2. Dividing IP networks into smaller subnetworks is done for a number of reasons:

- As discussed, subnetting usually makes more efficient use of available IP addresses. To look at another example, in the older IP address class system of determining network and host IDs, the smallest network, using a Class C address, consists of 254 hosts. If an organization requires only 30 host addresses, it can be assigned a portion of a Class C address, leaving most of the remaining addresses available for assignment on other networks.
- With subnetting, a company can divide its network into logical groups. When one large network is divided into two or more smaller subnetworks, a router is needed to allow hosts on one subnetwork to communicate with hosts on another subnetwork. A router serves as a natural security barrier between the two subnetworks because access control lists can be configured on a router to restrict the type of network traffic traveling from one subnet to another. Being able to restrict access enables network administrators to, for example, place the Payroll Department computers and servers on their own subnet and disallow computers from other subnets to access any resources in the Payroll Department.
- Subnetting can make network communication more efficient. Because routers don't forward broadcast traffic from one subnet to another, broadcast traffic generated by one subnet of 125 computers is heard and processed only by those 125 computers. If you have a single network of 500 computers, the amount of broadcast traffic is often detrimental to overall network performance. Subnetting into four networks of 125 computers each results in much smaller broadcast domains, which usually improves network performance.

Binary Arithmetic

IP address subnetting and supernetting (covered later in this chapter) is a lot easier if you understand the basics of binary arithmetic. For the purposes of this book, you need to be able to convert decimal numbers to binary numbers and vice versa.

Before you tackle these calculations, you need to review how the decimal numbering system works. It's based on powers of 10 (which is where the word "decimal" comes from, with "dec" meaning "ten"). Ten different symbols, 0 through 9, are used to represent any possible number. Each place in a decimal number can have one of 10 possible values: again, 0 through 9. Furthermore, each place in a decimal number can be expressed as a power of 10. The ones place can be expressed as a number, 0 thru 9, multiplied by 10 raised to the 0 power, or 10^0. (Any number raised to the 0 power equals 1.) The tens place can be

expressed as a number multiplied by 10 to the 1 power, or 10^1. The hundreds place can be expressed as a number multiplied by 10^2, and so on. For example, the decimal number 249 can be expressed as either of the following:

$$\mathbf{2} * 10^2 + \mathbf{4} * 10^1 + \mathbf{9} * 10^0 = 249$$
$$\mathbf{2} * 100 + \mathbf{4} * 10 + \mathbf{9} * 1 = 249$$

When you see the number 249, you don't think of it in these terms because you grew up using the decimal numbering system, and recognizing the hundreds place, tens place, and ones place happens without conscious effort, as does the multiplication and addition that occurs. However, take a look at this number:

```
379420841249
```

A little more thought has to go into recognizing that the 3 represents 300 billion, the 7 represents 70 billion, and so forth. The binary number system works the same way, except everything is governed by twos. Two digits, 0 and 1, represent every possible number, and each place in a binary number is 0 or 1 multiplied by a power of 2. So instead of having the ones place, the tens place, the hundreds place, and so on, you have the ones place, the twos place, the fours place, and so on, based on 2^0, 2^1, 2^2, and so forth. For example, using the same method you used to solve the decimal example, you can express the binary number 101 as either of the following. The numbers in bold are the binary digits.

$$\mathbf{1} * 2^2 + \mathbf{0} * 2^1 + \mathbf{1} * 2^0 = 5$$
$$\mathbf{1} * 4 + \mathbf{0} * 2 + \mathbf{1} * 1 = 5$$

Converting Decimal to Binary One method of converting decimal to binary is with a process called "successive divisions." With this method, you divide the decimal number by 2, write down the remainder (which must be 0 or 1), write down the dividend, and repeat until the dividend is 0.

The decimal number 125 is converted to binary in the following example:

```
125 divided by 2 equals 62, remainder 1
62 divided by 2 equals 31, remainder 0
31 divided by 2 equals 15, remainder 1
15 divided by 2 equals 7, remainder 1
7 divided by 2 equals 3, remainder 1
3 divided by 2 equals 1, remainder 1
1 divided by 2 equals 0, remainder 1
```

To produce the binary number corresponding to 125, you must then write the digits starting from the bottom of the remainder column and work your way up: 1111101. Now check the work involved. Because you have only seven binary digits and want to have eight total, you pad with zeros on the left, leaving you with 01111101. The exponential expansion of 01111101 is $0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$.

Another way to convert from decimal to binary is shown in Table 5-2. The first two rows are the binary and exponent values of each bit position of an 8-bit number. You use 8 bits because in subnetting, most work can be done 8 bits at time. The third row is what you complete to determine the decimal number's binary representation.

Table 5-2 Decimal-to-binary conversion table

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	**1**	**1**	**1**	**1**	**1**	**0**	**1**

To use this method, start with the number you're trying to convert to binary: in this case, 125, which is referred to as the "test number." You compare the test number with the leftmost number in the preceding table (128). If it's equal to or greater than this number, you place a 1 in the column and subtract the number in the column from your test number; otherwise, place a 0 in the column. Remember: Eight binary places or 8 bits can represent only a value up to 255. If you're converting a number greater than 255, simply extend the table to the left (256, 512, and so on). Here's the sequence of steps:

1. 125 is less than 128, so you place a **0** in the column under the 128. The test number remains 125.
2. 125 is greater than 64, so you place a **1** in the column under the 64 and subtract 64 from 125, leaving your new test number as 61.
3. 61 is greater than 32, so you place a **1** in the column under the 32 and subtract 32 from 61, leaving your new test number as 29.
4. 29 is greater than 16, so you place a **1** in the column under the 16 and subtract 16 from 29, leaving your new test number as 13.
5. 13 is greater than 8, so you place a **1** in the column under the 8 and subtract 8 from 13, leaving your new test number as 5.
6. 5 is greater than 4, so you place a **1** in the column under the 4 and subtract 4 from 5, leaving your new test number as 1.
7. 1 is less than 2, so you place a **0** in the column under the 2.
8. 1 is equal to 1, so you place a **1** in the column under the 1 and subtract 1 from 1, leaving your new test number as 0. When your test number is 0, you're done.

Now try this with 199, 221, and 24. You should get the following results:

```
199 = 11000111
221 = 11011101
24 = 00011000
```

Converting Binary to Decimal Using 11010011 as the example, here are the steps:

1. Count the total number of digits in the number (eight digits in 11010011).
2. Subtract one from the total (8 - 1 = 7).
3. The result of the subtraction (7) is the power of 2 to associate with the highest exponent for 1 in the number, so in this case, the first 1 in your binary number is multiplied by 2^7.
4. Convert to exponential notation, using all the digits as multipliers.
5. 11010011, therefore, converts to the following:

$$11010011 = 1 * 2^7 + 1 * 2^6 + 0 * 2^5 + 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$$
$$= 128 + 64 + 0 + 16 + 0 + 0 + 2 + 1 = 211$$

Another way to do this conversion is to use Table 5-2, as you did for the decimal-to-binary conversion. Of course, if your binary number is more than 8 bits, you can simply extend the table to the left as many places as necessary. Here's how to do it: Write your binary number in the third row of the table. For every column with a 1 bit, write down the corresponding decimal number from the first row. For columns with a 0 bit, you can simply skip them or write down a 0. Using the binary number 11010011, you have 1 bits in the 128, 64, 16, 2, and 1 columns. Add these values together, and you get 211. Choose some numbers and practice to make sure you understand how to do this conversion.

Recognizing Bit Patterns Many of the numbers you work with when subnetting have telltale patterns. For instance, subnet masks always consist of consecutive 1s followed by consecutive 0s. You never have, for example, a subnet mask that looks like 10110001 in binary (177 in decimal). A subnet mask always consists of a series of zero or more 1s, followed by a series of zero or more 0s, as Table 5-3 shows. Work on memorizing these correlations so that you can deal with subnet masking problems when you see them later in this chapter.

Table 5-3 Subnet mask bit patterns

Binary	Decimal
00000000	0
10000000	128
11000000	192
11100000	224
11110000	240
11111000	248
11111100	252
11111110	254
11111111	255

Table 5-4 lists the binary place values from the 1s place to the 128s place. It's important to know these values so that you can calculate lists of subnets quickly after you reallocate bits for a new subnet mask.

Table 5-4 Binary place values

Binary	Place value	Calculation
00000001	1s place	2^0
00000010	2s place	2^1
00000100	4s place	2^2
00001000	8s place	2^3
00010000	16s place	2^4
00100000	32s place	2^5
01000000	64s place	2^6
10000000	128s place	2^7

Calculating a Subnet Mask

Before you start creating subnet masks, you need to know under what situations you might need to do so. If you work for an ISP, you might need to subnet the address space your company allocates to its customers, as discussed earlier. You don't want to allocate hundreds of addresses to a company that needs only a few, for example. If you're the network administrator for a large corporate internetwork, you might need to subnet the address space your ISP has assigned to you or subnet the private network addresses you're using so that you can create smaller broadcast domains or logical groupings.

There are usually two approaches to subnetting, and they depend on the answer to these questions: Am I subnetting to provide a network with a certain number of host addresses? Or am I subnetting to provide a network with a certain number of logical subnets? If you're working for an ISP, the answer is usually yes to the first question, and if you're a network administrator for a corporate network, the answer is more likely to be yes to the second question. Sometimes the answer is a combination of both.

Say you have a large internetwork and need to break an IP address space into several subnets. Follow this process:

1. First, decide how many subnets you need. You can figure out the number of subnets needed by seeing how many network cable segments are or will be connected to router interfaces. Each router interface connection indicates a required subnet.
2. Next, decide how many bits you need to meet or exceed the number of required subnets. To calculate this value, use the formula 2^n, with n representing the number of bits you must reallocate from the host ID to the network ID. For example, if your starting network number is the Class B address 172.20.0.0, its default subnet mask is 255.255.0.0, which is your starting point. The number of subnets you create is always a power of 2, so if you need 20 subnets, you must reallocate 5 bits ($2^5 = 32$) because reallocating 4 bits gives you only 24 or 16 subnets.
3. Reallocate bits from the host ID, starting from the most significant host bit (that is, from the left side of the host ID).
4. You must also ensure that you have enough host bits available to assign to computers on each subnet. To determine the number of host addresses available, use the formula $2^n - 2$, with n representing the number of host (0) bits in the subnet mask.

Here's an example to help you put this formula to work: CNT Books Inc. wants 60 subnets for its Class B address: 172.20.0.0/16. The nearest power of 2 to 60 is 64, which equals 2^6. This means you must reallocate 6 bits from the host portion of the original subnet mask (255.255.0.0) and make them subnet bits.

Reallocating 6 bits, starting from the leftmost bit of the third octet, creates a subnet mask with the bit pattern 11111100. The decimal value for this number is 128 + 64 + 32 + 16 + 8 + 4, or 252. This reallocating of bits changes the subnet mask from 255.255.0.0 to 255.255.**252**.0.

To calculate the number of host addresses for each subnet, count the number of 0 bits remaining in the subnet mask to determine the number of bits left for the host ID. In this case, that number is 10 (2 in the third octet and 8 in the fourth). Again, the formula for determining the number of host addresses is $2^n - 2$, so you have $2^{10} - 2 = 1022$. 1022 addresses per subnet should be more than enough for most networks.

Now that you have a correct subnet mask, you need to determine what network numbers can be derived from using this subnet mask. To do this, take the reallocated 6 bits, place them in the network number, and cycle the 6 bits through the possible combinations of values they represent. Table 5-5 shows the first 16 subnetwork numbers resulting from the preceding steps, with the third octet written in binary on the left and the resulting subnetwork address written in decimal on the right. The bits shown in bold are the 6 bits used to create the subnets. If you convert the third octet on the left side from binary to decimal, you'll see that it equals the third octet on the right.

Table 5-5 Subnetwork numbers and addresses

Subnetwork number in binary	Subnetwork address
172.20.**00000**000.0	172.20.0.0
172.20.**00000**100.0	172.20.4.0
172.20.**00001**000.0	172.20.8.0
172.20.**00001**100.0	172.20.12.0
172.20.**00010**000.0	172.20.16.0
172.20.**00010**100.0	172.20.20.0
172.20.**00011**000.0	172.20.24.0
172.20.**00011**100.0	172.20.28.0
172.20.**00100**000.0	172.20.32.0
172.20.**00100**100.0	172.20.36.0
172.20.**00101**000.0	172.20.40.0
172.20.**00101**100.0	172.20.44.0
172.20.**00110**000.0	172.20.48.0
172.20.**00110**100.0	172.20.52.0
172.20.**00111**000.0	172.20.56.0
172.20.**00111**100.0	172.20.60.0
.....	.....
172.20.**11111**100.0	172.20.252.0

A Pattern Emerges Table 5-5 shows the first 16 of the possible 64 subnets and the last subnet created for network 172.20.0.0. As you can see, there's a pattern to the subnetwork numbers—they go in increments of 4. You can derive this pattern without having to list the subnets, however. Look at the octet where the subnet bits are reallocated, and then look at the rightmost reallocated bit. The subnet increment is determined by the binary place value of this bit (refer to Table 5-4): in this case, the 4s place.

You know when to stop counting subnets when all the subnet bits are binary 1s, as in the last entry in the table. You also know to stop counting when the subnet number equals the value of the changed octet in the subnet mask. In this case, the subnet mask 255.255.0.0

was changed to 255.255.**252**.0 after the bit reallocation. The 252 in the third octet of the subnet mask is the same value as the last subnet number.

Determining Host Addresses Similarly, the host addresses in each subnet can be determined by cycling through the host bits. Therefore, the subnetwork 172.20.32.0 would have host addresses from 172.20.32.0 through 172.20.35.255. However, you can't use the IP address in which all host bits are 1s because it's the broadcast address for that network, so your actual range is 172.20.32.1 through 172.20.35.254, giving you 1022 host addresses. Table 5-6 shows this for the first five subnets and the last subnet.

Table 5-6 Host addresses per subnet

Subnetwork number	Beginning and ending host addresses in binary	Beginning and ending host addresses in decimal
172.20.0.0	172.20.00000000.00000001–172.20.00000011.11111110	172.20.0.1–172.20.3.254
172.20.4.0	172.20.00000100.00000001–172.20.00000111.11111110	172.20.4.1–172.20.7.254
172.20.8.0	172.20.00001000.00000001–172.20.00001011.11111110	172.20.8.1–172.20.11.254
172.20.12.0	172.20.00001100.00000001–172.20.00010011.11111110	172.20.12.1–172.20.15.254
172.20.16.0	172.20.00010000.00000001–172.20.00010011.11111110	172.20.16.1–172.20.19.254
.....		.....
172.20.252.0	172.20.11111100.00000001–172.20.11111111.11111110	172.20.252.1–172.20.255.254

Another Subnet Mask Example In Figure 5-13, the network number is 192.168.100.0, which is a Class C network address with a default subnet mask of 255.255.255.0. The following steps show how to calculate a new subnet mask:

1. In this example, you can see that four cable segments are connected to router interfaces. The WAN cable segment between the two routers counts as a single cable segment and, therefore, a single subnet. You have to account for the WAN subnet even if the network has no hosts because the router interfaces require an IP address. As you can see, there are four subnetworks: Subnet A requires 43 IP addresses (40 for the Windows 7 hosts, 2 for the servers, and 1 for the router interface). Subnet B requires 53 IP addresses, subnet C requires 43 IP addresses, and subnet D requires only 2 IP addresses.
2. To accommodate the required number of subnets (4), you need a power of 2 that's equal to or greater than 4. Because $2^2 = 4$, you need to reallocate 2 bits from the host ID to the network ID.
3. Reallocating 2 bits from the leftmost part of the host portion of the original subnet mask (255.255.255.0) gives the last octet of your new subnet mask: the bit pattern 11000000. Converting to decimal and putting the entire subnet mask together yields 255.255.255.192. The 192 in the last octet is derived from adding the 128s place and the 64s place because they're the only 2 bits that are 1 in the third octet.

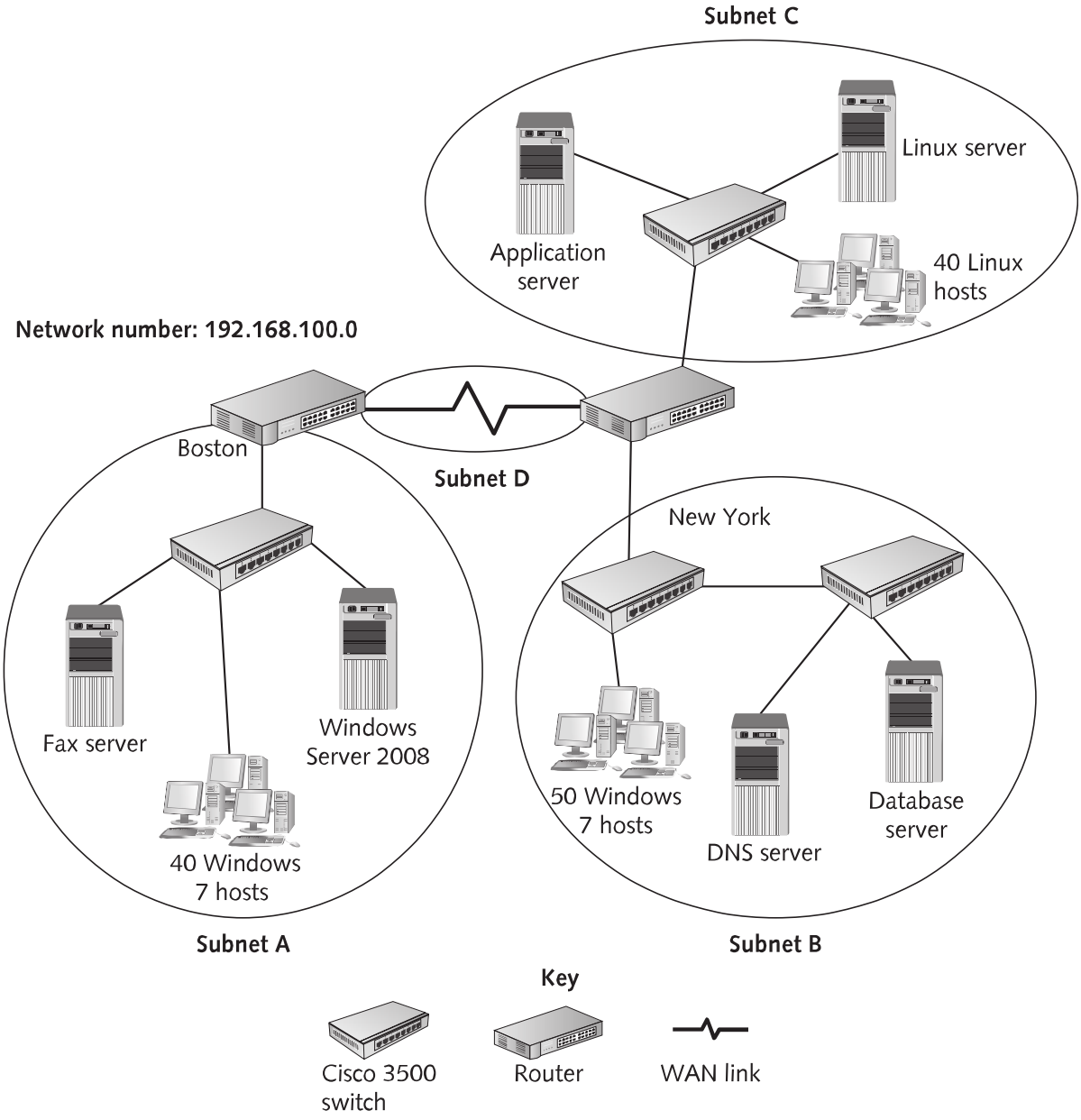

Figure 5-13 A sample network for calculating subnet mask requirements

Courtesy of Course Technology/Cengage Learning

4. To be sure you have enough host bits per subnet, use the formula $2^n - 2$, where n is the number of 0 bits in the new subnet mask. The result is $2^6 - 2 = 62$. This number of host addresses satisfies your requirement of a maximum of 53 hosts per subnet.

To learn more about this topic and try an outstanding tutorial to help you calculate subnets (and supernets, for that matter), go to *www.learntosubnet.com*.

Supernetting

Although not as commonly practiced as subnetting, **supernetting** is sometimes necessary to solve certain network configuration problems and to make routing tables more streamlined. With routing tables, supernetting is usually referred to as "route aggregation" or "route summarization."

Supernetting reallocates bits from the network portion of an IP address to the host portion, effectively making two or more smaller subnets a larger supernet. Supernets allow combining two or more consecutive IP network addresses and make them function as a single logical network. Here's how it works:

1. Say you have four Class C network addresses—192.168.0.0, 192.168.1.0, 192.168.2.0, and 192.168.3.0—available for your network design. You have a total of 900 hosts on your proposed network. You don't have four router interfaces that can use the four different network numbers, however. You can combine the four networks into one by reallocating 2 bits ($2^2 = 4$) from the network portion of the address and adding them to the host portion. You then have a network address of 192.168.0.0 with the subnet mask 255.255.252.0. The 252 in the third octet is derived from setting the last 2 bits of the original Class C subnet mask (255.255.255.0) to 0, thereby making them part of the host portion.
2. Instead of supporting only 8 bits for the host address portion, the supernet now supports 10 bits (8 + 2) for host addresses. This number of bits provides 2^{10} - 2 host addresses on this supernet, or 1022, which satisfies your requirement for 900 hosts and allows you to assign all host addresses in a single network.

As mentioned, combining two or more small networks into one larger network is only one reason to supernet. Routers on the Internet can have enormous routing tables. The larger the routing table, the more work the router must do to determine where to send a packet. Route aggregation or summarization can combine multiple routing table entries into a single entry, which can drastically decrease the table's size on Internet routers. This reduction in routing table size increases routers' speed and efficiency. The procedure is similar to supernetting, except you configure routers.

Routing tables grow partly because routers communicate with one another by sending their routing tables to one another. If several networks can be represented by a single routing table entry, the routing tables are more efficient. Taking the previous example, suppose RouterA in a corporate network has the network addresses 192.168.0.0, 192.168.1.0, 192.168.2.0, and 192.168.3.0 in its routing table, and it communicates with RouterB (see Figure 5-14). Without supernetting/route summarization, RouterA sends all four network addresses to RouterB, each with its 255.255.255.0 subnet mask. Consequently, RouterB's routing table expands with these four additional routes. However, because all four routes lead to the same place (RouterA), these routes can be represented by a single entry. RouterA can summarize these routes by simply sending RouterB the address 192.168.0.0 with subnet mask 255.255.252.0, which tells RouterB that the addresses 192.168.0.1 through 192.168.3.254 can be reached through RouterA.

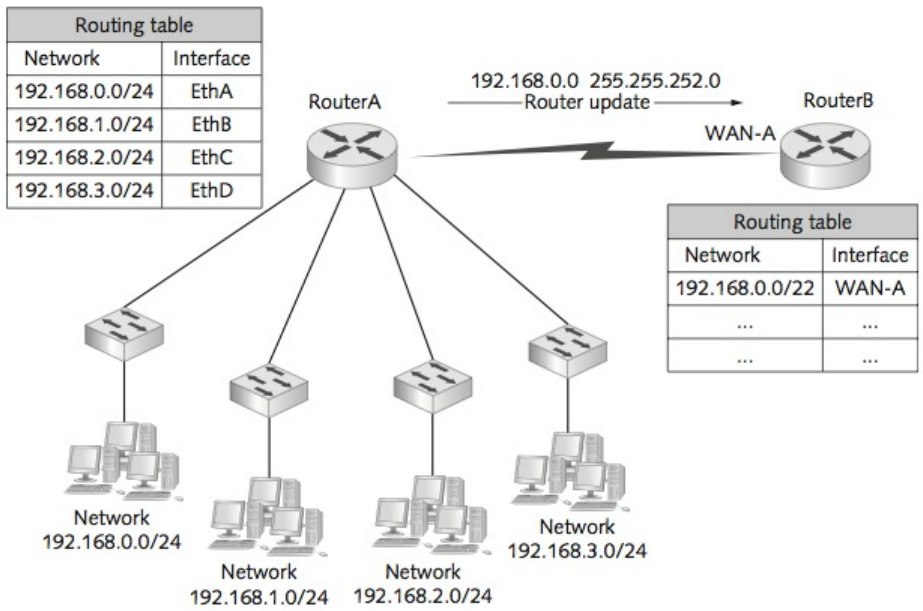

Figure 5-14 RouterA sends a summary of its routing table to RouterB

Courtesy of Course Technology/Cengage Learning

Introduction to Internet Protocol Version 6

IPv6 is the network community's answer to resolving some problems in IPv4, including limits of the 32-bit address space, a lack of built-in security, a sometimes complicated setup, and a lack of built-in **quality of service (QoS)**. QoS describes a network's capability to prioritize data packets based on the type of information they contain (for example, voice, video, or file data) or urgency of the information. QoS headers in IPv6 packets can identify packets that require special or priority handling, making applications such as streaming audio and video much easier to implement.

An IPv6 address is 128 bits rather than the 32 bits in an IPv4 address. This length increases the number of possible addresses from about 4 billion in IPv4 to 3.4×10^{38} addresses (that's 34 followed by 37 zeros!) in IPv6. Unless IP addresses are assigned to every star in the universe, it's safe to say enough IPv6 addresses will be available.

The IPSec protocol provides authentication and encryption. In IPv4, it must be added to a network, but it's already incorporated in IPv6. Authentication ensures that the sender and receiver of data packets are known to each other and have permission to send and receive data. Encryption makes the data in packets unreadable except to the computers involved in the transmission.

IPv6 is autoconfiguring, which means there's no IP address to assign and no subnet mask to determine. Two types of autoconfiguration are available in IPv6:

- Stateless autoconfiguration is the simplest. When a workstation starts, it listens for information broadcast by a local router and assigns itself an address based on the network configuration broadcast by the router and the station's MAC address.

- Stateful autoconfiguration relies on a DHCP server, as with IPv4. This configuration type requires setting up and configuring a DHCPv6 server. Of the two autoconfiguration types, stateless autoconfiguration is usually the most common.

IPv6 Addresses Unlike IPv4 addresses, which are specified in dotted decimal notation in 8-bit sections, IPv6 addresses are specified in hexadecimal format in 16-bit sections separated by a colon, as in this example: 2001:1b20:302:442a:110:2fea:ac4:2b.

If one of the 16-bit numbers doesn't require four hexadecimal digits, the leading 0s are omitted. Furthermore, some IPv6 addresses contain consecutive 0s in two or more 16-bit sections, so a shorthand notation is used to eliminate consecutive 0 values. Two colons replace two or more consecutive 0 values, as the following example shows:

- Longhand notation: 2001:DB8:0:0:0:2ed3:340:ab
- Shorthand notation: 2001:DB8::2ed3:340:ab

There's actually some order to these seemingly cumbersome IPv6 addresses. In the IPv6 address space, an addressing hierarchy of three parts is used: a public topology, a site topology, and an interface identifier. In short, the first three 16-bit sections (totaling 48 bits) represent the public topology, which could be an Internet backbone service provider, for example. The next 16 bits represent the site topology, such as a business or a local ISP, and the last 64 bits (four 16-bit sections) represent the interface identifier, which is derived from the MAC address on the host's NIC. The interface identifier is the unique host address.

When IPv6 is commonplace, this hierarchical address scheme will make locating Internet resources faster and more efficient and eliminate the need for problem-prone IPv4 processes, such as NAT and complicated subnetting and supernetting.

Hexadecimal Notation Hexadecimal notation, which you have seen in MAC addresses and now IPv6 addresses, is a numbering system like decimal and binary. Hexadecimal, or just hex, is based on powers of 16 and uses 16 symbols to represent all possible numbers. Rather than invent new symbols, however, the numbers 0 to 9 are used for the first 10 symbols and the letters A to F for the remaining 6 symbols, which have the values 10 to 15 in decimal. A hex number, therefore, is expressed by using the symbols 0 to F, and each place value is based on a power of 16. For example, the hex number 4C can be converted to decimal by $4 \times 16^1 + C \times 16^0$. The symbol C represents decimal 12, so 4C in decimal is 64 + 12 = 76.

Hexadecimal notation is often used to represent numbers in the computer world because it can be converted easily to binary, as it's based on powers of 2. For example, $2^4 = 16$, so every hex digit can be expressed as exactly 4 bits. Converting from hex to binary is just a matter of converting each digit to its 4-bit binary equivalent. For instance, AC4F in hexadecimal is expressed as 1010 1100 0100 1111 in binary.

Networking doesn't require a lot of hex-to-binary conversion until you start working with IPv6 addresses. If you're an aspiring programmer, however, understanding the hex numbering system is sure to be beneficial.

The IPv6 Host ID The host ID of an IPv6 address is typically 64 bits and uses the interface's MAC address to make up the bulk of the address. Because a MAC address is only 48 bits, the other 16 bits come from the value FF-FE inserted after the first 24 bits of the MAC address. In addition, the first two zeros composing most MAC addresses are replaced

with 02. For example, given the MAC address 00-0C-29-7C-F9-C4, the host ID of an IPv6 address is 02-0C-29-FF-FE-7C-F9-C4. This autoconfigured 64-bit host ID is referred to as an Extended Unique Identifier (EUI)-64 interface ID.

By default, Windows Vista and later don't use EUI-64 interface IDs when configuring the link-local address. Instead, they create random interface IDs when autoconfiguring an interface address. However, you can configure Windows to use the EUI-64 interface address with the following command.

```
netsh interface ipv6 set global randomizeidentifiers=disabled
```

You might wonder whether there's an equivalent for the IPv4 loopback address 127.0.0.1. There is, and it's ::1. If you expand this address, it's 0:0:0:0:0:0:0:1.

In contrast to Windows, Linux does use EUI-64 IPv6 host addressing.

Subnetting with IPv6 Although subnetting as done in IPv4 will be a thing of the past, it doesn't mean subnetting won't be used at all in IPv6 networks. Typically, ISPs allocated IPv4 addresses to businesses in groups specified by a network address and an IP prefix. ISPs try to give a business only the number of addresses it requires. However, with IPv6 having such a large address space, most address allocations will have a /48 prefix, even for small home networks. This means the network ID is 48 bits, and the network administrator has 80 bits for assigning subnets and host IDs. Because the host ID is 64 bits, 16 bits are left for creating subnets. This number of bits allows 65,536 subnets, more than enough for all but the largest organizations. Large conglomerates can get multiple /48 prefix addresses or /47 prefix addresses, which provide more than 130,000 subnets. A typical IPv6 address, then, as assigned by an ISP looks like Figure 5-15.

Figure 5-15 Structure of a typical IPv6 address

Courtesy of Course Technology/Cengage Learning

With 16 bits available to subnet, there are many strategies you can use. A small network that doesn't have multiple subnets can simply leave the subnet ID as all 0s, for example, and an address in this situation might look like this:

```
2001:DB8:A00:0000:020C:29FF:FE7C:F9C4/64
```

This address begins with 2001:DB8, which is not random. The IPv6 developers realized that people will be writing about how to work with IPv6, so instead of authors using random values for examples, they reserved 2001:DB8 for use in documentation. The A00 in the address is the last 16 bits of the network prefix and was chosen randomly for this example. The 0s following the A00 are the subnet ID, and the last 64 bits are the computer's interface ID. The /64 just indicates that the network portion of the address is the first 64 bits (network prefix plus subnet ID).

A network that does need to subnet could just take the 16 bits for the subnet ID and start counting. For example, a company could make the first three subnets as follows; the bold part of the address is the subnet ID, and the 64-bit interface ID has been omitted.

- 2001:DB8:A00:0000
- 2001:DB8:A00:0001
- 2001:DB8:A00:0002

Large organizations with multiple locations could take a more structured approach and assign each location a bank of subnets, as in the following example:

- 2001:DB8:A00:0000—Assigned to New York location
- 2001:DB8:A00:4000—Assigned to London location
- 2001:DB8:A00:8000—Assigned to Shanghai location

With this strategy, each location has 4000 hexadecimal subnet IDs to work with. For example, New York can make subnets 2001:DB8:A00:0000, 2001:DB8:A00:0001, 2001:DB8:A00:0002, and so forth, up to 2001:DB8:A00:3FFF. Put another way, each location can configure up to 16,384 subnets. As you can see, subnetting does still exist in IPv6, but it's a more straightforward process than in IPv4.

For more on IPv6, see *http://technet.microsoft.com/en-us/library/bb878121.aspx*.

This chapter has focused on network protocols, specifically those in the TCP/IP suite. Other protocol suites exist, such as IPX/SPX, NetBEUI, and AppleTalk, but they're considered legacy protocols that are no longer used in new network installations. Previous editions of this book covered these protocols, but for this edition, discussion of them has been moved to Appendix B to allow room to focus more on TCP/IP.

Chapter Summary

- TCP/IP is the main protocol suite used in networks. Like most facets of networking, TCP/IP takes a layered approach and is organized in these four layers: Application, Transport, Internetwork, and Network access.
- The Network access layer is composed of network technologies such as Ethernet and token ring. The Internetwork layer is where most network configuration occurs and is composed of IP, ICMP, and ARP. The Transport layer provides reliability and works with segments. The Application layer consists of protocols such as HTTP and DNS and provides an interface for applications to access network services.
- IP addresses are divided into a network ID and host ID. There are three primary address classes: A, B, and C. Address classes determine the default network ID and host ID portions of an IP address. Each class has a range of private IP addresses that can't be used on the Internet; they're used to address private networks. NAT must be used to access the Internet when a host is assigned a private address.
- CIDR largely replaces the IP address class system addressing; it uses a prefix number or subnet mask to determine the network and host IDs of an IP address.

- Subnetting enables an administrator to break a large network into two or more smaller networks that require a router for communication. It also allows an ISP to allocate only the number of public IP addresses a company requires instead of assigning an entire address class.
- IPv6 will eventually replace IPv4 because of its larger 128-bit address space and built-in security and QoS features. IPv6 addresses are expressed as eight four-digit hexadecimal values.

Key Terms

Address Resolution Protocol (ARP) An Internetwork-layer protocol used to resolve a host's IP address to its MAC address. ARP uses a broadcast frame containing the target host's IP address, and the host that's assigned the address responds with its MAC address.

address space The number of addresses available in an IP network number that can be assigned to hosts.

ARP cache A temporary storage location in an IP host's RAM that keeps recently learned IP address/MAC address pairs so that the ARP protocol isn't necessary for each packet sent to a host.

Automatic Private IP Addressing (APIPA) A private range of IP addresses assigned to an APIPA-enabled computer automatically when an IP address is requested via DHCP but no DHCP server responds to the request. *See also* Dynamic Host Configuration Protocol (DHCP).

Classless Interdomain Routing (CIDR) A method of IP addressing in which the network and host IDs are determined by a prefix number that specifies how many bits of the IP address are network bits; the remaining bits are host bits.

connectionless protocol A type of network communication in which data is transferred without making a connection between communicating devices first, and the receiving station gives no acknowledgement that the data was received.

Domain Name System (DNS) An Application-layer protocol that resolves computer and domain names to their IP addresses. DNS uses the UDP Transport-layer protocol. *See also* User Datagram Protocol (UDP).

dotted decimal notation The format used to express an IPv4 address; four decimal numbers separated by periods.

Dynamic Host Configuration Protocol (DHCP) An Application-layer protocol used to configure a host's IP address settings dynamically. It uses the UDP Transport-layer protocol because DHCP messages consist of a single packet and are used on the local LAN.

flow control A mechanism network protocols use to prevent a destination device from becoming overwhelmed by data from a transmitting computer, resulting in dropped packets.

fully qualified domain name (FQDN) A name that includes the hostname, subdomain names (if applicable), second-level domain name, and top-level domain name, separated by periods.

Internet Control Message Protocol (ICMP) An Internetwork-layer protocol used to send error and control messages between systems or devices. It's an encapsulated IP protocol, meaning it's wrapped in an IP header.

Internet Message Access Protocol (IMAP) An Application-layer protocol used by an e-mail client to download messages from an e-mail server; operates on TCP port 143. IMAP also provides fault-tolerance features. It downloads only message headers from the server initially, and then downloads the message body and attachments after the message is selected.

Internet Protocol Security (IPSec) An extension to IP that provides security by using authentication and encryption. It authenticates the identity of computers transmitting data with a password or some other form of credentials, and it encrypts data so that if packets are captured, the data will be unintelligible.

Internet Protocol Version 4 (IPv4) A connectionless Internetwork-layer protocol that provides source and destination addressing and routing for the TCP/IP protocol suite. Uses 32-bit dotted decimal addresses.

Internet Protocol version 6 (IPv6) A connectionless Internetwork-layer protocol that provides source and destination addressing and routing for the TCP/IP protocol suite. Uses 128-bit hexadecimal addresses and has built-in security and QoS features.

IP address A 32-bit dotted-decimal address used by IP to determine the network a host resides on and to identify hosts on the network at the Internetwork layer.

IP prefix A value used to express how many bits of an IP address are network ID bits. Usually expressed as */PrefixNumber*—for example, 192.168.1.24/27, with 27 as the IP prefix.

localhost The name used to refer to the loopback address in an IP network. *See also* loopback address.

loopback address An address that always refers to the local computer; in IPv4, 127.0.0.1 is the loopback address.

Network Address Translation (NAT) A service that translates a private IP address to a public IP address in packets destined for the Internet, and then translates the public IP address in the reply to the private address.

octet A grouping of 8 bits, often used to identify the four 8-bit decimal numbers that compose an IP address (as in "first octet," "second octet," and so forth).

Port Address Translation (PAT) An extension of NAT, a service that allows several hundred workstations to access the Internet with a single public Internet address by using Transport-layer port numbers to differentiate each host conversation. *See also* Network Address Translation (NAT).

port number A field in the Transport-layer protocol header that specifies the source and destination Application-layer protocols that are used to request data and are the target of the request, respectively.

Post Office Protocol version 3 (POP3) An Application-layer protocol used by a client e-mail application to download messages from an e-mail server; operates on TCP port 110.

protocol Rules and procedures for communication and behavior. Computers must use a common protocol and agree on the rules of communication.

protocol stack *See* protocol suite.

protocol suite A set of protocols working cooperatively to provide network communication. Protocols are "stacked" in layers in which each layer performs a unique function required for successful communication. Also called a protocol stack.

quality of service (QoS) A term that describes a network's capability to prioritize data packets based on the type of information they contain (for example, voice, video, or file data) or urgency of the information.