

Introduction to Networking

Objectives

Upon completion of this chapter, you should be able to answer the following questions:

- What are the requirements for an Internet connection?
- What are the major components of a personal computer (PC)?
- What procedures are used to install and troubleshoot network interface cards (NICs) and modems?
- What basic testing procedures are used to test the Internet connection?
- What are the features of web browsers and plug-ins?
- What are the Base 2, Base 10, and Base 16 number systems?
- How do you perform 8-bit-binary-to-decimal and decimal-to-8-bit-binary conversions?
- How do you perform simple conversions between decimal, binary, and hexadecimal numbers?
- How are IP addresses and network masks represented in binary form?
- How are IP addresses and network masks represented in decimal form?

Key Terms

This chapter uses the following key terms. You can find the definitions in the Glossary:

Internet page 4

enterprise network page 4

Internet service provider (ISP) page 6

personal computers (PCs) page 7

central processing unit (CPU) page 8

random-access memory (RAM) page 9

disk drive page 9

hard disk page 9

input/output devices (I/O) page 9

motherboard page 9

memory chip page 9

parallel port page 10

serial port page 10

mouse port page 10

keyboard port page 10

Universal Serial Bus (USB) port page 10

expansion slots page 10

network interface card (NIC) page 11

video card page 11

sound card page 11

jack page 14

local-area network (LAN) page 14

continued

Ethernet page 14

plug-and-play page 16

bits per second (bps) page 17

networking devices page 17

Media Access Control (MAC) address page 19

servers page 20

media page 21

modems page 22

digital subscriber line (DSL) page 23

standards page 25

protocols page 25

Hypertext Transfer Protocol (HTTP) page 25

Internet Protocol (IP) page 25

protocol suite page 25

Transmission Control Protocol/Internet Protocol (TCP/IP)
page 25

web browser page 26

web servers page 26

binary digit (bit) page 26

byte page 26

plug-ins page 28

Transmission Control Protocol (TCP) page 28

IP address page 31

dotted decimal page 31

Universal Resource Locator (URL) page 35

default gateway page 37

ping page 39

tracert page 42

ASCII page 47

decimal numbering (Base 10) page 48

binary numbering (Base 2) page 49

hexadecimal numbering (Base 16) page 59

This chapter introduces the basic concepts and components of modern computer networks, including the basics of the TCP/IP protocol suite, upon which most modern networks are built. This chapter also covers some of the related binary, decimal, and hexadecimal math that is required to examine the details of how computer networks work. This chapter, along with Chapter 2, “Networking Fundamentals,” provides an overview of many of the topics related to computer networking, introduces many terms, and provides a solid foundation before you get into more detailed subjects in later chapters.

Connecting to Networks and the Internet

The Networking Academy course that you are (likely) taking when using this book may be your first formal introduction to the world of computer networking. However, today, most people have grown up with networks and networking as part of the overall culture of the developed world. As a result, most people start this course and book with some opinions about what a network really is and what the Internet is. This section formally defines a network. It also defines the basic concepts and terms behind one special and important network: the Internet.

What’s a Network?

To formally begin your networking journey, you need to start forming a more detailed and specific answer to the question “What’s a network?” Assuming that you took the time and effort to register for the Cisco Networking Academy Program CCNA 1 course, which is a basic networking course, you probably already have some opinions about the answer to this question. This section begins to answer the question.

First, consider the following formal, but general, definition of a computer network:

A combination of computer hardware, cabling, network devices, and computer software used together to allow computers to communicate with each other.

The goal of any computer network is to allow multiple computers to communicate. The type of communication can be as varied as the type of conversations you might have throughout the course of a day. For example, the communication might be a download of an MP3 audio file for your MP3 player; using a web browser to check your instructor’s web page to see what assignments and tests might be coming up; checking the latest sports scores; using an instant-messaging service, such as AOL Instant Messenger (AIM), to send text messages to a friend; or writing an e-mail and sending it to a business associate.

This chapter starts the process of closely looking at the four networking components mentioned in the formal definition: computer hardware, computer software, cabling, and networking devices. Before you look at each component, however, it is helpful to think about some examples of networks.

A Small Network: Two PCs and One Cable

You can create a simple network with two computers and a cable. Although it's not a terribly impressive network, such a network does occasionally serve a good purpose in real life, as well as being useful for discussing networking and learning some basic skills in classroom labs.

Figure 1-1 shows such a network.

Figure 1-1 A Two-PC, One-Cable Network

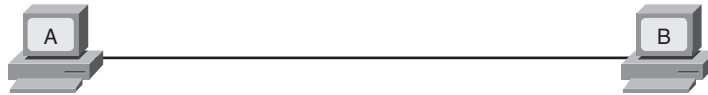


Figure 1-1 shows two computers, A and B, and a line that represents a networking cable. Implemented properly, this small network allows computers A and B to communicate. (That “implemented properly” phrase is simply a way to ignore the details you will learn over the coming months. More on that is covered in upcoming chapters.) This network certainly meets the formal definition for a computer network because multiple computers can communicate.

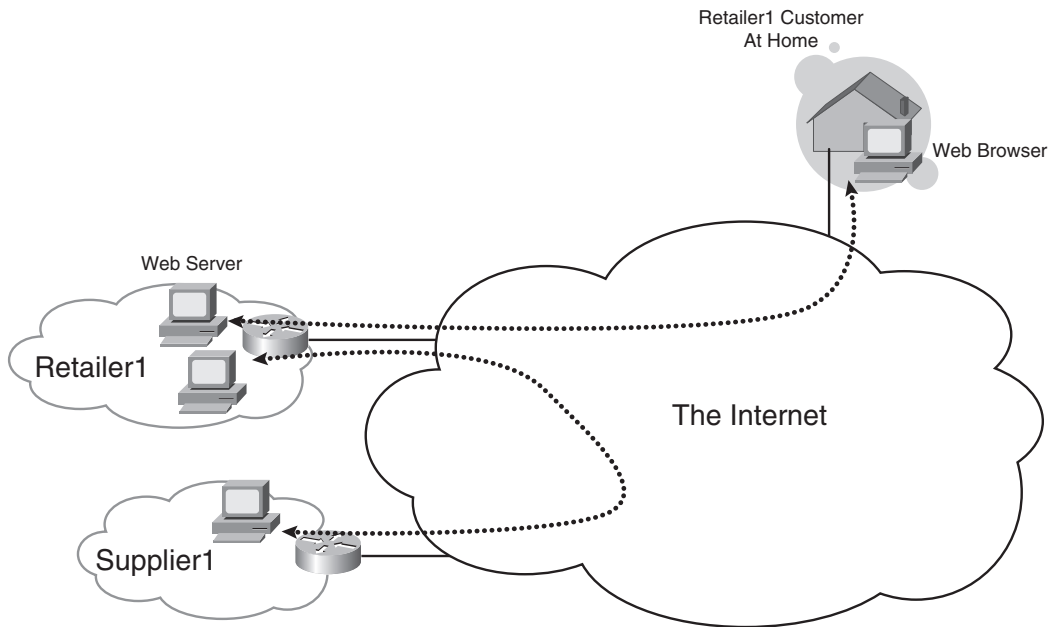
Although this network might seem small, small networks do have some useful purposes. For example, when you download a song to your PC and copy the song to an MP3 player over a cable, you have effectively created a small network. Another example of a small network is when two people with laptops attend the same meeting and use wireless to exchange files while sitting in the meeting.

A Very Large Network: The Internet

Consider a network that is the opposite of the simple network shown in Figure 1-1: the Internet. The Internet is somewhat challenging to define because it means many different things to so many people. From one perspective, the *Internet* is a very large, global network that allows almost every computer on the planet to communicate with the other computers on the planet. Not only is it a network in the formal sense, the communication it enables worldwide, across cultures and political boundaries, has fundamentally changed the world as we know it.

Under close examination, however, the Internet isn't a network at all. It's really a bunch of interconnected networks. In fact, that's how it got its name: Internet is short for *interconnected networks*. Figure 1-2 depicts part of the Internet.

All the pieces of Figure 1-2 create the Internet. First, on the left, two enterprise networks are shown: Retailer1 and Supplier1. The term *enterprise network* refers to a network built by one company, one government institution, one school system, or any other entity. In this case, these two companies hired network engineers to plan and implement a network that these companies' employees can use. At that point, the companies can carry on business communications between computers inside their respective companies.

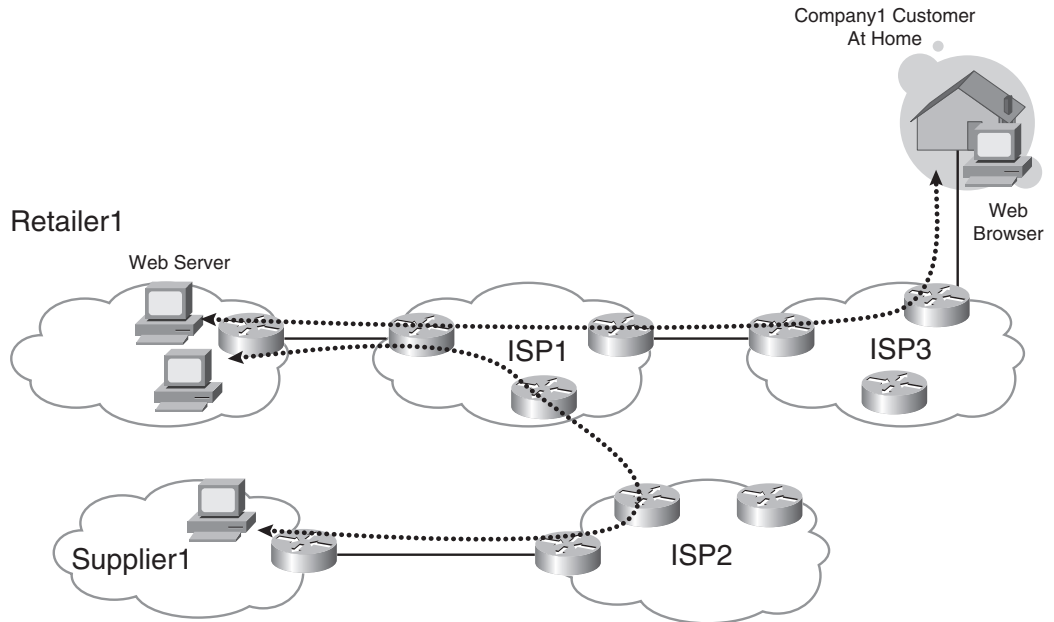
Figure 1-2 Internet

Besides communicating inside their respective companies, these two companies need to communicate with each other. Retailer1 needs to exchange information with its supplier, Supplier1. (For example, the retailer might simply need to order additional stock to fill its stores' shelves.) So, both Retailer1 and Supplier1 connect to the Internet, which allows the computers in the two companies to exchange information, such as orders and invoices, check on shipping and product availability, and the like.

Retailer1 also needs to communicate with its customers. Because Retailer1 sells consumer products, these consumers need to be able to get to Retailer1's website, which is located inside Retailer1's enterprise network. Therefore, Retailer1 has a second reason to connect to the Internet.

Next, potential customers also need to connect to the Internet. In Figure 1-2, the Retailer1 customer sits at home and uses a home computer and an Internet connection. After she's connected to the Internet, the customer can browse Retailer1's website, find products, order the products, pay via a credit card, and so on.

The Internet includes literally hundreds of thousands of enterprise networks, hundreds of millions of home users, and a mysterious cloud in the middle of Figure 1-2. When drawing figures of computer networks, if a portion of the network contains details that are not important to a particular discussion, that part of the network is typically represented as a cloud. Figure 1-2 is no exception. It shows the "Internet" as a big cloud without any details. Figure 1-3 removes the cloud, shows some details, and shows some other clouds.

Figure 1-3 Internet: A Closer Look

The core of the Internet is not one entity, but many. To create the Internet, a company called an *Internet service provider (ISP)* creates a network. An ISP then sells its services to businesses and individuals, with the most basic service being the ability for the customers' computers to send and receive data to and from any other computer on the Internet. To provide this basic overall service, an ISP must provide a customer with two things:

- A connection between an enterprise network, or a home user, and the ISP's network
- Connections between the ISP's network and every other part of the Internet

In Figure 1-3, three different ISPs supply a network connection to their respective customers. The home user, Retailer1, and Supplier1 each pay a fee, typically monthly, to their respective ISPs for the right to connect to that ISP. However, they do not pay money to the other two ISPs shown in the figure. For example, Retailer1 uses ISP1, so Retailer1 pays only ISP1 for its Internet service. Such agreements allow any company or individual to connect to an ISP, and it provides competition to keep prices more reasonable.

The ISPs must connect to each other so that they can forward traffic to all parts of the Internet. Figure 1-3 shows a direct line, which represents some networking cables, between two pairs of ISPs. The ISPs must have some path to each other so they can forward traffic between their respective customers, fulfilling their promise to connect their customers to the rest of the Internet.

ISPs do not need a direct connection to all other ISPs to meet the requirement of being able to reach all parts of the Internet. For example, ISP2 and ISP3 might need to send data between

each other for some of their customers. To do so, they send it through ISP1. As long as some path exists so all ISPs can reach all other ISPs in the world by using one or more different ISPs, the requirement for complete connectivity to the Internet is accomplished.

Perspectives on Networks of Different Sizes

Comparing the simple network of Figure 1-1 with the Internet in Figure 1-3 shows how different networks can be, particularly in size. In fact, many individual enterprise networks connected to the Internet have more than 10,000 computers connected to them, in hundreds of locations. These types of enterprise networks are complex in and of themselves. Also, home users might have multiple computers connected to a home network that's connected to the Internet.

Interestingly, as you dig deeper into how networks work, you can see that many of the networking concepts covered in this class are used in small, medium, and large networks—even the Internet. Certainly, the larger the network, the more work and effort it takes to successfully implement the network. However, that complexity—and the requirement for more effort and work to successfully implement networks—is actually a good thing because it means more jobs, more variety in those jobs, and more opportunity.

Next, you closely look at some network components and begin to understand how network engineers can construct a network.

Network Components

The people who create a computer network, referred to as network engineers, create networks by combining the four things mentioned in the formal definition of a network:

- Computer hardware (including NICs)
- Cables
- Networking devices
- Computer software

This section closely looks at the first three of these networking components. Networking software is covered later in this chapter in the section “TCP/IP Protocol Suite and TCP/IP Software.”

Computer Hardware

Computers come in many shapes, sizes, and types. However, the vast majority of people use computers that are best categorized as personal computers. *Personal computers (PCs)* are computers that are specifically designed to be used by a single person at a time.

Although some knowledge of the basics of PCs is important for this course, you do not need detailed knowledge of PCs to do well in this course. If you are new to computers or if you want further background on PCs, take the HP IT Essentials I: PC Hardware and Software course or

Note

Figure 1-3 shows several cylindrical icons that resemble hockey pucks. They represent a networking device called a router. Later chapters of this book, and major portions of the other three courses of the Networking Academy CCNA curriculum, expound upon the purpose and inner workings of routers.

read the book *HP IT Essentials I: PC Hardware and Software Companion Guide* (published by Cisco Press).

The next several subsections cover the most commonly discussed PC components.

General Types of PC Components

From a basic perspective, a PC has the following components:

- **Processor (also called a *central processing unit [CPU]*)**—A computer processor, or CPU, acts as a computer's brain. A CPU's job is to process, or think about, what the computer is trying to do. (Figure 1-4 shows a picture of a CPU.) The CPU's job includes many things, such as the following:
 - Creating the image that is displayed on the computer's screen
 - Taking input from the keyboard or mouse
 - Sending data over a network
 - Processing data for software running on the computer

Figure 1-4 CPU



- **Microprocessor**—A silicon chip that contains a CPU. A typical PC has several microprocessors, including the main CPU.
- **Temporary memory (also called *random-access memory [RAM]*)**—The processor needs memory in which to work on things. RAM is the computer equivalent of the papers and notes you might keep on your desk when studying. The CPU can quickly and easily access the data stored in RAM, and that data typically pertains to something the PC is actively processing. Note that the contents of RAM are lost when the computer is powered off.
- **Read-only memory (ROM)**—ROM is a type of computer memory in which data has been prerecorded. After data has been written onto a ROM chip, it cannot be removed; it can only be read. (PCs can re-record information into another type of ROM, called electronically

erasable programmable read-only memory [EEPROM]. The basic input/output system [BIOS] in most PCs is stored in EEPROM.)

- **Permanent memory (such as disks)**—Computers need long-term memory to store data that might be needed later, even after a computer is powered off. Permanent memory typically consists of a type of device called a *disk drive* or *hard disk*.
- **Input/output devices (I/O)**—To interact with humans, the computer must be able to know what the human wants it to do and provide the information to the human. Humans tell a computer what to do by manipulating an input device. For example, this occurs when the human types on the PC keyboard or moves/clicks with a mouse. For output, the computer uses a video display, audio speakers, and printers.

With these components, a PC can take input from the human, possibly gather data from the disk drives into RAM, process the data with the CPU, and provide the results through one of the output devices.

Motherboard

The PC *motherboard* holds many of the PC's most important components. The motherboard is a flat piece of plastic called a circuit board. Circuit board material is designed to be a good place to physically attach microprocessor chips, such as the CPU and RAM, and connect the components with wires and other hardware. The following list details some of the motherboard's individual components:

- **Printed circuit board (PCB)**—A thin plate on which chips (integrated circuits) and other electronic components are placed. Examples include the motherboard and various expansion adapters.
- **Transistor**—A device that amplifies a signal or opens and closes a circuit. Microprocessors can have millions of transistors.
- **Integrated circuit (IC)**—A device made of semiconductor material. It contains many transistors and performs a specific task. The primary IC on the motherboard is the CPU. ICs are often called chips.
- **Memory chips**—Another name for RAM, memory chips are Integrated circuits whose primary purpose is to be used to temporarily store information that is processed by the CPU.
- **Resistor**—An electrical component that is made of material that opposes the flow of electric current.
- **Capacitor**—An electronic component that stores energy in the form of an electrostatic field. It consists of two conducting metal plates separated by insulating material.
- **Connector**—A port or interface that a cable plugs into. Examples include serial, parallel, USB, and disk drive interfaces.
- **Light emitting diode (LED)**—A semiconductor device that emits light when a current passes through it. LEDs are commonly used as indicator lights.

- **Parallel port**—An interface that can transfer more than 1 bit at a time. It connects external devices, such as printers.
- **Serial port**—An interface used for serial communication in which only 1 bit is transmitted at a time. The serial port can connect to an external modem, plotter, or serial printer. It can also connect to networking devices, such as routers and switches, as a console connection.
- **Mouse port**—A port designed for connecting a mouse to a PC.
- **Keyboard port**—A port designed for connecting a keyboard to a PC.
- **Power connector**—A connector that allows a power cord to be connected to the computer to give electrical power to the motherboard and other computer components.
- **Universal Serial Bus (USB) port**—This interface lets peripheral devices, such as mice, modems, keyboards, scanners, and printers, be plugged and unplugged without resetting the system. PC manufacturers may one day quit building PCs with the older parallel and serial ports completely, instead using USB ports.
- **Firewire**—A serial bus interface standard that offers high-speed communications and real-time data services.

Expansion Slots and the PC Backplane

For various reasons, some parts of a computer cannot be easily attached to the motherboard. For example, disk drives are too large to attach directly to the motherboard. However, these devices still need to be accessible to the motherboard. So, the motherboard includes connectors that allow other parts, such as disk drives, to connect to the motherboard through a cable.

Other necessary computer components might be connected to the motherboard, or might not be connected, at the discretion of the PC's manufacturer. For example, the function provided by a NIC, which is important to networking, might be included on the motherboard of a PC, or it might not. (The upcoming section "Network Interface Cards" discusses NICs in more detail.)

To allow for additional functions besides what is provided on a particular PC's motherboard, PCs typically have the physical capability to accept expansion cards. These cards are built with the same general types of components as the motherboard: a circuit board, microprocessor chips, capacitors, and the like. However, these expansion cards typically fulfill a specific purpose. For example, if the motherboard does not include the same function as a NIC, a NIC can be added to the PC as an expansion card.

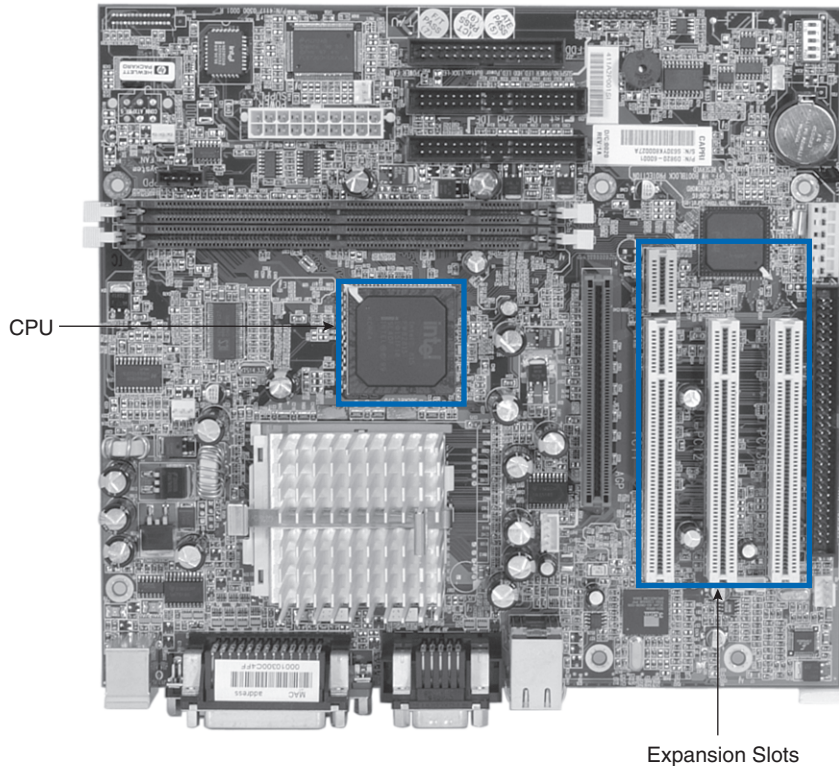
For expansion cards to be useful, they must connect to the motherboard through the PC backplane. The backplane is part of the motherboard that is designed as a place to allow the connection of the expansion cards. The backplane also provides several standardized plastic connectors, called **expansion slots**, into which expansion cards can be inserted. By connecting expansion cards into the backplane, the cards and the motherboard can communicate. Figure 1-5 shows a motherboard with the expansion slots in the lower-right part of the figure (the white vertically oriented rectangles).

Note

Although a NIC is an optional component of a PC, because most consumers want network access, most every new PC sold today has an integrated NIC.

Note

Expansion cards are sometimes called expansion boards, or sometimes simply boards or cards.

Figure 1-5 PC Motherboard and Expansion Slots**Note**

The term bus is used to refer to a PC's backplane.

The following list identifies some of the more popular cards found in these expansion slots:

- **Network interface card (NIC)**—An expansion board that provides a network communication connection to and from a PC. Many newer desktop and laptop computers have an Ethernet NIC built into the motherboard. (Ethernet is the most popular type of local-area network [LAN] in use today.)
- **Video card**—A board that plugs into a PC to give it display capabilities. Video cards typically include onboard microprocessors and additional memory to speed up and enhance graphics display.
- **Sound card**—An expansion board that handles all sound functions.

Miscellaneous PC Components

This section completes this chapter's list of some of the PC's components. Specifically, it defines a few different types of permanent storage options and the system unit and power supply:

- **CD-ROM drive**—An optical drive that can read information from a CD-ROM. This can also be a compact disk read-write (CD-RW) drive, a digital video disk (DVD) drive, or a combination of all three in one drive.
- **Floppy disk drive**—A device that can read and write to floppy disks (see Figure 1-6).

Figure 1-6 Floppy Disk Drive

- **Hard disk drive**—A device that reads and writes data on a hard disk. This is the primary storage device in the computer.
- **System unit**—The main component of the PC system. It includes the case, chassis, power supply, microprocessor, main memory, bus, expansion cards, disk drives (floppy, CD hard disk, and so on), and ports. The system unit does not include the keyboard, the monitor, or any other external devices connected to the computer.
- **Power supply**—The component that supplies power to a computer by taking alternating current (AC) and converting it to 5 to 12 volts direct current (DC) to power the computer.

Desktop Versus Laptop

Laptop computers differ from desktop computers in that they can be easily transported and used. Laptops are generally smaller than desktops, with built-in video display and keyboard so that transporting it is convenient. In most cases, laptops weigh less than 10 pounds and have a battery that lasts several hours, so they can be used while traveling.

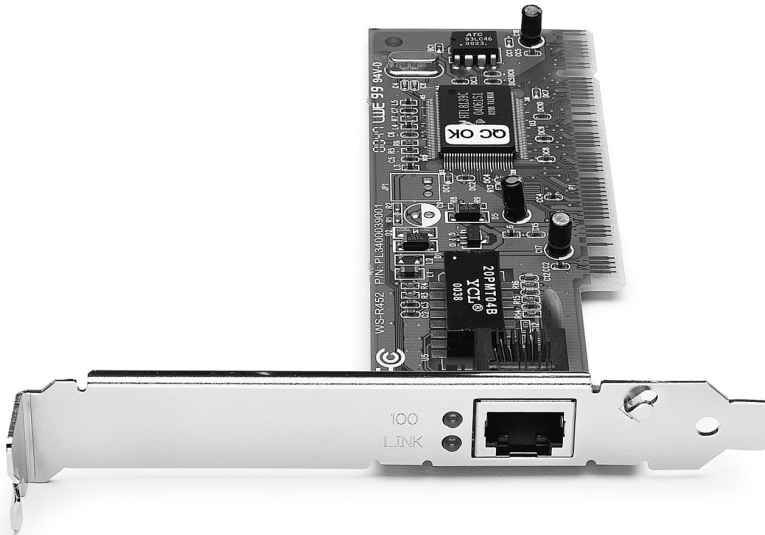
Laptop expansion slots are called *Personal Computer Memory Card International Association (PCMCIA)* slots or PC card slots. Devices such as NICs, modems, hard drives, and other useful devices, usually the size of a thick credit card, insert into the PC card slot. Figure 1-7 shows a PC card for a wireless LAN (WLAN).

Figure 1-7 PCMCIA Card**Lab 1.1.2 PC Hardware**

In this lab, you become familiar with the basic peripheral components of a PC system and their connections, including network attachments. You examine the internal PC configuration and identify major components. You also observe the boot process for the Windows operating system (OS) and use the Control Panel to find information about the PC hardware.

Network Interface Cards

For a PC to use a network, it must have some interface to the network cabling. PCs use network interface cards (NICs) to provide that interface. In fact, the name is somewhat self-descriptive: NICs are expansion cards that give a PC an interface to a network. Figure 1-8 shows a NIC.

Figure 1-8 Ethernet NIC

When installing the card in a PC, the part of the card on the right side of Figure 1-8, which slightly sticks out, is the part that is inserted into the expansion slots shown in Figure 1-5. The silver part at the bottom of the figure is the part that can be seen from the outside of the PC. This side has an opening, typically called either a socket or a *jack*, into which the networking cable can be inserted. (A jack is simply a hole, with a standard shape, into which a cable can be easily inserted.)

Figure 1-8 shows an Ethernet LAN NIC. The term *LAN* refers to a general type of network in which the distances between computers are relatively short—hence the phrase “local-area” in its name. Several types of LANs have been defined over the years, including Token Ring, Fiber Distributed Data Interface (FDDI), and *Ethernet*. Today, however, you are likely to use only Ethernet, in fact, the other two types of LANs have become so unpopular that it is difficult to find and purchase Token Ring or FDDI NICs.

When purchasing a NIC, consider the following features:

- **LAN protocol**—Ethernet, Token Ring, or FDDI. Today, Ethernet is used almost exclusively, and Token Ring and FDDI are seldom used. (Token Ring and FDDI are introduced briefly in Chapter 6, “Ethernet Fundamentals.”)
- **Media supported**—Twisted pair, coaxial, wireless, or fiber optic.
- **Bus support on the computer**—PCI or Industry-Standard Architecture (ISA).

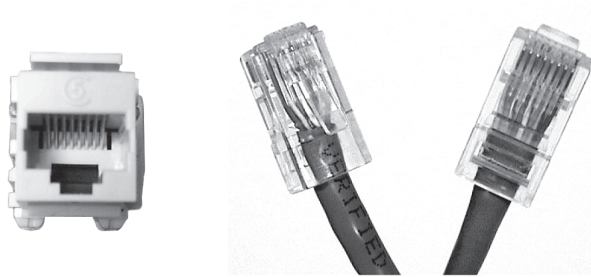
Finding an Ethernet NIC on Your PC

It might be an interesting exercise, at home or in the classroom, to look for an Ethernet NIC in your PC. To find one, look at the back of the PC where most of the connectors sit. Look for a jack, which is roughly rectangular in shape, like the dark rectangle on the side of the NIC shown in Figure 1-8. That’s the socket into which the cable is inserted. The cable attached to the NIC typically has a connector called an RJ-45 connector (RJ means registered jack). (The cable’s connector is simply the shaped plastic end of the cable.) The RJ-45 connector is shaped like the connector used on telephone cable for your home telephone, but it’s just a little wider. Figure 1-9 shows the shape of the RJ-45 jack and connectors to help you in your search.

Note

The connector on your home telephone cable is called an RJ-11 connector.

Note that although a NIC can be inserted into a PC’s expansion slot, many of the newer PCs sold today integrate the NIC’s functions onto the motherboard. So, when you look on your PC’s NIC, if you do not see the RJ-45 socket on any of the cards in the expansion slots, look at the other parts on the back of the computer; you might see the RJ-45 socket that’s connected to the motherboard.

Figure 1-9 RJ-45 Jack and Connectors

Installing Ethernet NICs

These days, the physical installation of a NIC is relatively easy. The IT Essentials course covers the details in more depth, but the process is extremely similar for any expansion card:



Step 1 Shut down or power off the PC.

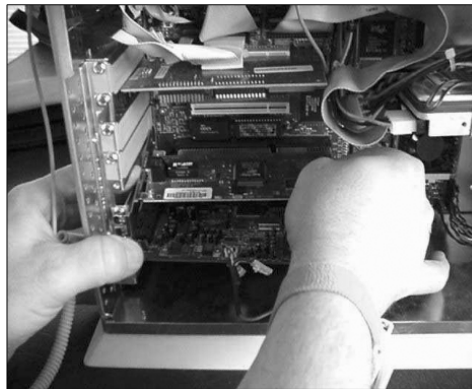
Step 2 Disconnect the power cord from the PC.

Step 3 Connect an antistatic strap to your wrist to protect the computer and NIC from your body's static electricity.

Step 4 Insert the NIC into the expansion slot.

Step 5 Reassemble the PC and turn it on.

After practicing a time or three, you should be able to easily insert the card. Figure 1-10 shows the physical installation.

Figure 1-10 Installing an Ethernet NIC

Besides the physical installation, the installer might or might not need skills relating to how to add hardware to the computer's OS, which is the software that controls the computer's actions. (For example, Windows XP and Linux are both computer OSs used on PCs.) Oftentimes today,

you can physically install the NIC and turn the computer on, and the OS automatically adds the hardware. This process is called *plug-and-play*.

If the plug-and-play process does not work, a successful NIC installation might require you to examine and change software settings using the software that comes with the NIC. This might require knowledge of how the NIC is configured in the OS and how to use the NIC diagnostics. It might also require that you have the skills to resolve hardware resource conflicts. The IT Essentials course covers this knowledge and skill.

Network Cabling Basics

For networks to work, the computers must have the capability to take the bits sitting in RAM on one computer and somehow send a copy of those bits into RAM on the other computer. For the process to work, the computer typically asks the NIC to send the bits over the cable that is connected to the NIC. So, the NIC must be connected to some form of transmission medium over which it can send the bits to the other computer. This section introduces the concept of transmission medium by using a specific example: the small Ethernet network shown in Figure 1-1, which uses an electrical Ethernet cable.

As mentioned in this book's Introduction, the chapters generally cover a slightly broader range of topics than the online curriculum. The small amounts of extra coverage provide you with better context or different ways to think about the same concepts.

Occasionally, and particularly in this chapter, this book provides several pages of additional information on a few important topics. This section and the upcoming section "Networking Devices" are the first two sections in this chapter that take the discussions much deeper than what's given in the CCNA course. You might appreciate the additional depth at various points in your reading. However, if you or your instructor prefer to skip over these deeper bits of additional coverage, feel free to skip forward to the section "Enterprise Networks and Home Internet Access."

The topics in this section are covered in more depth in Chapters 3, 4, 5, and 7, and the topics in the section "Networking Devices" are covered to some degree in Chapters 2, 5, 8, 9, and 10.

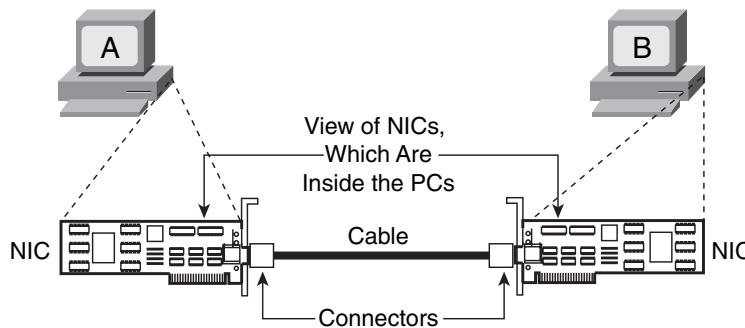
Creating a Transmission Medium Using Cables

To send data—bits—to another computer, the computers can use some physical medium over which electricity can be sent. In this case, the physical medium is typically a set of copper wires because copper easily conducts electricity. Because copper wires are brittle, the wires are usually wrapped in a colored plastic coating to help prevent them from breaking. The plastic coating also helps provide some electrical insulation, which is a property discussed in detail in Chapter 3, "Networking Media."

The LAN cable connected to the NIC contains multiple copper wires. NICs tend to use more than one wire, with a typical Ethernet NIC using four wires. Additionally, the outer part of the cable, which is made of flexible plastic and other insulating materials, adds strength and protection to the combined wires. The cable keeps the set of wires together for convenient use and physically protects the wires.

For the simple network in Figure 1-1 to work (for example, for PC B to send bits to PC A), PC B needs to be able to send electricity to PC A. To do so, the copper wire inside the cable needs to be physically touching something inside the NICs on each end of the cable to have a path from PC B to PC A over which electricity can pass. Figure 1-11 shows the idea, with the NICs inside the PCs removed from the figure to more clearly show the details.

Figure 1-11 Connecting the Copper Conductors in a Cable to the NICs



By connecting the NICs in the two PCs with the correct cable, the NICs now have a physical path between the two PCs over which electricity can flow. Now, they can use this electrical path to transmit bits.

The term *bits per second (bps)* often refers to the speed of network connections. Note that the unit is bits, not bytes. In real life, LANs typically run at much higher speeds, with a slow LAN transmitting at 10 million bits per second (megabits per second, or Mbps). The section “Names for the Rate at Which a Network Sends Data” describes the speeds at which bits can be sent over a networking cable.

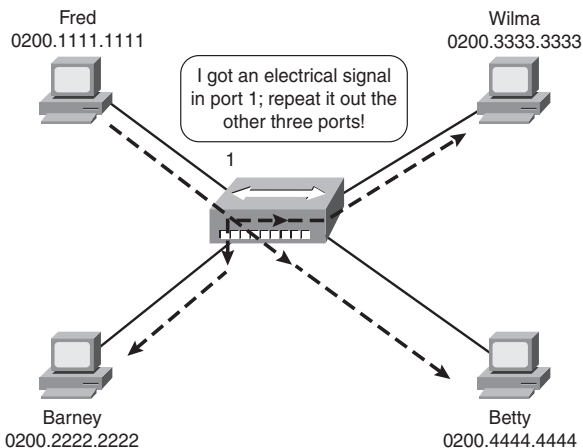
Networking Devices

This chapter began with a formal definition of a network, which included components such as computer hardware and cabling. This chapter so far has described, in general terms, hardware and cabling. This section introduces the idea behind what the online curriculum calls *networking devices* and the role they play in creating networks. In particular, this section shows an example of a simple networking device called a LAN hub.

Companies create their LANs by connecting a cable between each computer and some networking device, oftentimes either a LAN hub or LAN switch. In turn, these devices connect to each other. Designed and engineered properly, each computer can use one cable to connect to only one networking device but still can communicate with many other computers.

The term networking device refers to a class of computer hardware devices that is designed for the specific purpose of building networks. Many types of networking devices exist, including two that are discussed throughout all four courses of the Networking Academy CCNA curriculum: switches and routers. For example, LANs can use one type of network device called a LAN hub (or simply hub). (The term hub comes from the idea of a wheel, in which the hub is in the middle and several spokes radiate from it.) A LAN hub has a large number of jacks (oftentimes, RJ-45 sockets) that connect the LAN cables attached to various PCs. By connecting each PC to the hub with a single cable, the network has an electrical transmission medium between each PC and the hub. When the hub receives electrical signals, it simply repeats those signals out all the other interfaces. Figure 1-12 shows an example of the cabling topology.

Figure 1-12 Small LAN with a Hub Network Device



With a hub, the PCs need only a single NIC and a single cable that connects them to the hub. Figure 1-12 shows PC Fred transmitting data (shown as a dashed line), with the hub repeating the transmitted bits to each device connected to the hub. If another PC joins the network, the new PC simply needs a NIC with a cable that connects it to the hub; the existing devices require no changes.

A hub is just one example of a networking device that can be used to reach the main objective of networks: to allow multiple computers to communicate. LAN switches are another type of networking device similar to a LAN hub. A switch does the same work as a hub, but more efficiently, so today, switches are used more often than hubs. Also, routers (as shown in Figures 1-2 and 1-3) perform an important role. Routers can connect to wide-area networks (WANs), which provide a transmission medium across long distances. Chapter 2 closely looks at these networking devices, and later chapters further detail hubs, switches, and routers.

The behavior of a LAN using a hub also provides a good backdrop from which to cover a topic related to NICs—namely, the concept of an address for a NIC. The company that makes the NIC gives it a unique permanent address. The address is 48 bits long and is typically written in hexadecimal as 12 hexadecimal digits. (Each hexadecimal digit represents 4 bits.) When writing down these addresses, Cisco Systems tends to put two periods into the address to make it more readable. For example, in Figure 1-12, Fred has a NIC address of 020011111111, which is listed as 0200.1111.1111 on a Cisco product. The NIC address has many names besides NIC address. The most common name is *Media Access Control (MAC) address*.

When sending data, Fred adds a prefix to the data, including the MAC address of the intended recipient (for example, Barney). After the other three PCs receive the data, they use the destination MAC address to decide whether to process the data (Barney) or not (Wilma and Betty). (This is just one example of how NICs use their MAC addresses; you will learn about many other uses for MAC addresses before the end of this course.)



You can view a simulation of the network operation in Figure 1-12 by using Packet Tracer. Download the sample Packet Tracer scenarios from your login at <http://www.ciscopress.com/title/1587131641>, and load scenario NA01-0112. For more information, refer to this book's Introduction.

Enterprise Networks and Home Internet Access

So far, this chapter has introduced three of the four main components of a computer network: computer hardware (including NICs), cables, and network devices. Today, most every enterprise has a network installed, with the devices using many of the concepts covered thus far. This section closely looks at enterprise networks and then examines a few of the technologies used to access the Internet.

Enterprise Network Basics

Enterprises vary in many ways. The network created and owned by the college, school, or training company at which you are taking this course mostly likely has a network. In fact, the PCs in the classroom might be connected to that network. If so, the PCs are actually part of the enterprise network that the school uses. An enterprise network is nothing more than a network created to support the activities of that enterprise, whether that activity is to make money, govern, or educate.

Enterprise networks can be large or small. Regardless of size, enterprise networks have many common requirements and features:

- They might have several physical locations that are too far away to use a LAN.
- The need to support many PCs for the people who use the enterprise network.
- The need to connect servers to the network. *Servers* have information that is useful and important for the enterprise's functions.
- Typically, the computer-support engineers work near one or a few of the network's main sites.
- The need for Internet access for most or all of the PCs in the enterprise.

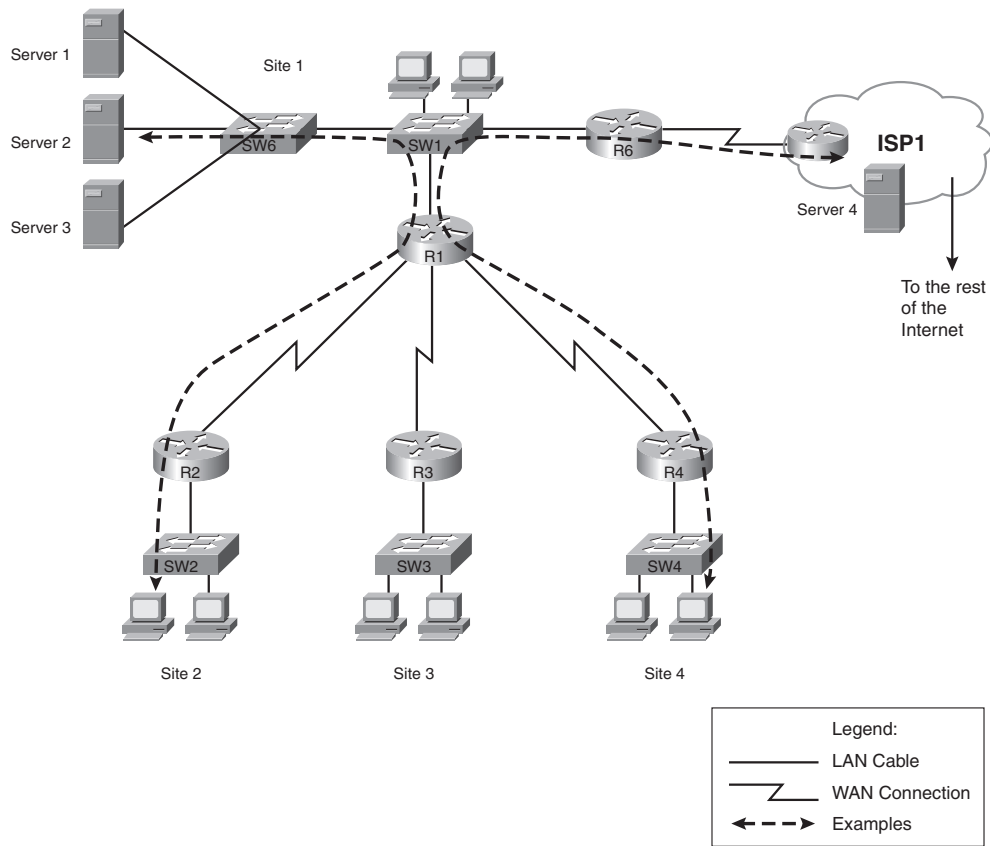
To support such goals and requirements, an enterprise network might look like what's shown in Figure 1-13. The figure shows an enterprise with four sites: one main site and three other sites. This enterprise could be a small business, local government, or even a school with four campuses.

In Figure 1-13, Site 1 houses a server farm, which is a collection of servers located in the same location. The PCs throughout the enterprise network can access these servers through the enterprise network. For example, a PC at Site 2 is shown communicating with one of the servers (as noted with the dark dashed line). Note that each PC is connected to a networking device called a LAN switch; the switch allows local communications over a LAN. (Switches achieve the same goal as hubs, but more efficiently, as covered briefly in Chapter 2 and in more depth in Chapter 7, "Ethernet Technologies," and Chapter 8, "Ethernet Switching.") The routers (hockey-puck icons) connect to the LAN and the WAN, forwarding traffic to and from the WAN connection. (WAN connections are often represented by a crooked line, sometimes called a lightning bolt.)

Only the main site has a connection to the Internet through ISP1. Although only Site 1 has the connection, the PCs throughout the enterprise use the enterprise network to reach router R6, which then forwards traffic into the Internet through ISP1.

This design allows the users inside the enterprise to access the Internet, as well as users inside the Internet to access devices inside the enterprise. This enterprise-network design takes care of Internet connectivity for the entire enterprise. The next section covers the basics of Internet connectivity from the home.

Figure 1-13 Typical Enterprise Network



Accessing an ISP Through a Phone Line and an Analog Modem

As previously mentioned, many ISPs exist, with most vying for a share of the home Internet access market. As shown in Figure 1-2, to use the Internet from home, a PC must somehow connect to an ISP. These ISPs try to get your attention through marketing and advertising and want you to sign up with them to gain access to the Internet.

To access the Internet, a home PC needs to use some transmission medium. With LANs, the transmission medium is a cable that is relatively short, typically less than 100 meters in length. However, the distance between your house and the offices where the ISP keeps its networking devices might be long. Rather than install an expensive miles-long cable between your house and an ISP, only to have you change to a new ISP in two months, ISPs use *media* (plural of medium) that are already installed in your house—namely, a phone line or a cable TV cable. This section describes how telephone lines work with analog modems. You learn more about how networks use cable TV lines in the section “Accessing an ISP Through a Cable TV Cable and a Cable Modem,” found later in this chapter.

When a phone call is placed, telephone companies essentially create an electrical circuit between two phones. When used to send and receive voice, the phones convert the sounds into an analog electrical signal by using a microphone built into the mouthpiece of the phone, sending that signal to the other phone. The receiving phone converts it back to sounds and plays it out a speaker built in to the phone's earpiece.

The analog electrical signal used by phones differs from digital electrical signals in that analog varies continuously, as shown in Figure 1-14.

Figure 1-14 Analog Electrical Signal

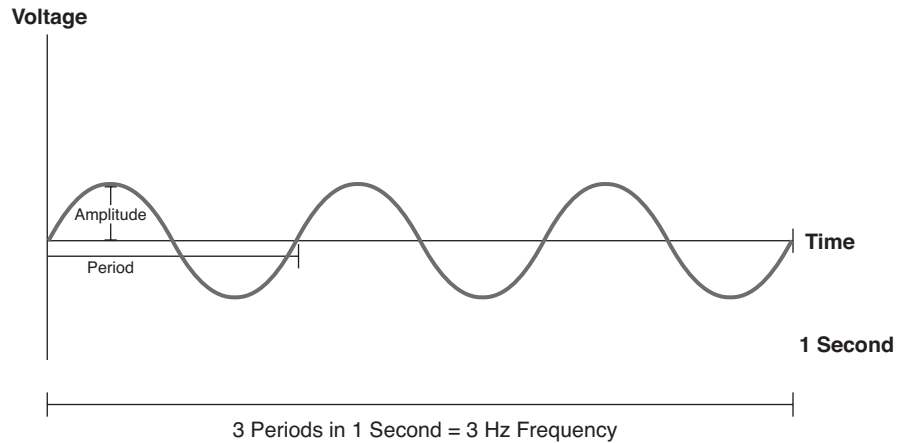


Figure 1-14 shows the voltage level on the y axis over time, with time on the x axis. In short, the voltage continuously varies from some positive to negative voltage (in other words, alternating current). Note that Figure 1-14 shows three complete cycles of the curve, beginning at the x axis, going up, down again, and back up to the x axis. The length of time it takes for one complete cycle to occur is called the *period*. The number of times that cycle occurs in 1 second is called the *frequency*. For example, the frequency shown in Figure 1-14 is 3 Hertz (Hz). (Hertz is a unit of measure that means “number of cycles per second.”) Finally, the maximum (positive) and minimum (negative) voltage is called the *amplitude*.

Phone companies originally used analog electrical signals for voice traffic because analog signals look like voice waves as they travel through the air. Both electrical energy and sound waves vary in terms of frequency, amplitude, and so on. For example, the higher the frequency, the higher the pitch of the sound; the higher the amplitude, the louder the sound. An electrical signal sent by a phone is *analogous* to the sound waves; hence, the term *analog* describes this type of signal.

Computers can use *modems* to send data to each other by sending analog signals, such as the signal shown in Figure 1-14. Two computers can essentially place a phone call to one another by using their modems. The modems then send binary 0s and 1s to each other by varying, or

modulating, the analog signal. For example, the modems might use a higher-frequency signal to mean 1 and a lower frequency to mean 0. From the phone company's perspective, however, it looks just like any other phone call because all the modems do is send the same kinds of analog electrical signals sent by a telephone.

Modems might exist as a built-in feature on the computer motherboard or as a separate internal expansion card. Modems might also be external to the PC, connecting to the PC via the PC's serial port. Such a modem is aptly named an external modem.

A Brief History of Remote Access

This section briefly reviews the history of modems and Internet access, even before the advent of the Internet. The idea of remote access to the Internet from home did not become a mainstream offering until the early 1990s. Before then, however, devices like modems were used for other purposes. The following list outlines the major events in the history of remote access:

1960s—Modems were used by dumb computer terminals to access large computers called mainframes.

1970s—Bulletin Board Systems (BBSs) allowed original PCs to access a server, let you post a message for others, and look at messages posted by others.

1980s—File transfers between computers became popular, as well as rudimentary interactive graphics.

1990s—Internet access through a modem became popular, with modem speeds rapidly increasing to 56 Kbps.

2000s—High-speed Internet access became more affordable and more popular.

The next two sections examine the high-speed Internet access methods that developed in the late 1990s, which are now mainstream technologies.

Accessing an ISP Through a Phone Line and a DSL Modem

Digital subscriber line (DSL) defines a much higher-speed method of using the same home phone lines that modems use. DSL differs from how modems work in many ways, such as the following:

- DSL uses digital electrical signals, not analog.
- Unlike modems, DSL allows a voice call to use the phone line simultaneously as DSL passes data.
- To not interfere with a concurrent voice call on the same phone line, DSL uses frequencies outside the range typically used for voice.
- DSL Internet service is “always on” in that no phone call or other effort must be made to access the Internet.

To use DSL, the home PC needs either an internal or external DSL modem or DSL router. Each of these devices understands, and uses DSL, but with slightly different features that are beyond the scope of the Networking Academy CCNA curriculum. Figure 1-15 shows a typical home installation with an external DSL router/modem.

Figure 1-15 Basic Operation of Modems

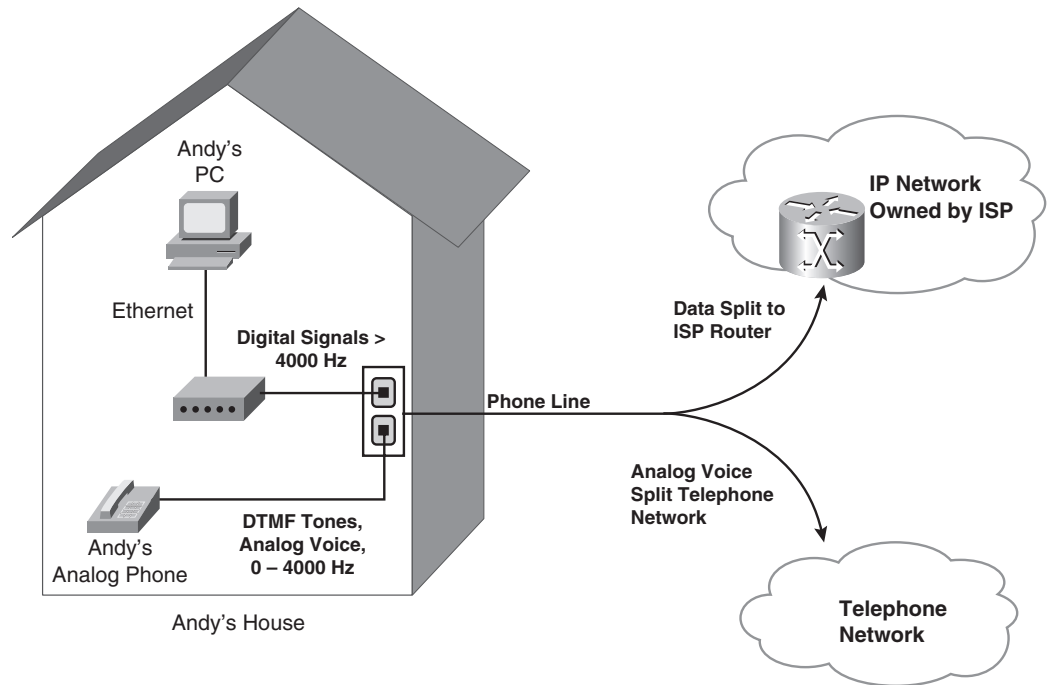


Figure 1-15 shows how both the normal phone and the DSL router connect, through a typical telephone line with an RJ-11 connector, to a phone socket. The local telephone company then splits out the voice to the telephone network and splits out the data to give to an ISP. In effect, the PC now has an electrical path between itself and the router, with the ability to send bits to and from that router. In turn, this access provides the PC with Internet connectivity.

Accessing an ISP Through a Cable TV Cable and a Cable Modem

Cable TV companies provide Internet access that, from a nontechnical perspective, is similar to DSL. The PC needs either an internal or external cable modem or a cable router. The cable modem/router connects to the CATV cable instead of the phone line. The cable modem sends and receives data to and from a router inside the cable TV company; at the same time, the CATV cable is still available for its primary purpose: TV. In other words, you can watch TV and switch channels all you want, while someone else in the house uses the Internet over the same CATV cable. Like DSL, the service is always on and doesn't require the user to do anything before beginning to use the Internet. Similar to DSL, it is fast, with download speeds well over 1 Mbps.

TCP/IP Protocol Suite and TCP/IP Software

The one network component that has not yet been covered in this chapter is software. Software provides the motivation and the reasons why a computer tries to communicate in the first place. You might build a network with computer hardware, NICs, modems, cables, and networking devices, but if no software exists, the computers do not attempt to communicate. Software provides that logic and that motivation for a computer to communicate.

You might have used computer software that, in turn, caused the computer to use the network. If you have ever opened a web browser to look at web pages or surf the web, you have used computer software that drives traffic across the network. In fact, because web browsers are so commonly used today, this section uses web browsers as examples.

Networking Standards, Protocols, and the TCP/IP Protocol Suite

For computer communications to be useful, the communication must follow a set of rules. Networking rules are formally defined by many different networking *standards* and networking *protocols*. Individually, a single networking standard or networking protocol defines the rules for a small part of what a network does. For example, an encoding scheme used by an Ethernet NIC would be a single standard. This section looks at a few networking protocols as examples, namely *Hypertext Transfer Protocol (HTTP)* and *Internet Protocol (IP)*.

Computers and network devices implement protocols mainly through computer software. For example, when you download a web page, the web browser uses HTTP, which the web-browser software implements. To deliver the data to and from the web server, the PC might use other protocols, such as the aforementioned IP. Today, most computer OSs implement IP, so IP is already built in to the OS and available for use.

For a network to work, all network components must use the same set of standards and protocols. Many options exist for standards and protocols to do the same (or similar) functions, so to help make sense of all that, a concept called a *protocol suite* or networking model was created. A protocol suite is a set of protocols through which, when a computer or networking device implements many of the protocols in the suite, the computers can communicate easily and effectively.

The *Transmission Control Protocol/Internet Protocol (TCP/IP)* protocol suite defines and collects a large set of networking standards and protocols that are used on most computers today. The concepts and protocols covered in this section, including HTTP and IP, are part of the TCP/IP protocol suite. Chapter 9, “TCP/IP Protocol Suite and IP Addressing,” covers the TCP/IP protocol suite (also called the TCP/IP networking model) in depth.

Note

A computer OS is software that controls the computer hardware, providing a human interface to the computer. Windows XP and Linux are two examples of PC OSs.

Using HTTP to Download a Web Page

This section covers what might be a familiar topic: the use of a *web browser*. What is probably new is how web-browser software implements one of the many protocols inside the TCP/IP protocol suite (specifically, the HTTP protocol) to get the contents of a website from a web server. In case you're less familiar with web browsers and *web servers*, the next section briefly introduces these concepts.

Web Browsers and Web Servers

Web-browser software shows information in a window of a PC's video display. That information might be simple text, graphics, video, or animation. The browser can also play audio. Today, the most popular web-browser software comes from Microsoft, called Internet Explorer (IE). Netscape and Mozilla Firefox are two other popular web browsers.

A web server is software that distributes information from the web server to web browsers. For example, Cisco Systems has a website (www.cisco.com) that lists tons of information about its products and services. Cisco creates the website by placing the web pages on a server, installing web server software on that server, and telling the web server software to supply the web pages to any browsers that request the web pages.

Note that many people use the terms *website* and *web page* to refer to web-based content. The term *website* refers to a bunch of related content. For example, in the case of Cisco, if you spend time clicking different links in the browser after starting at www.cisco.com, everything you look at is part of the Cisco website. At any one point in time, however, you look at an individual web page.

Web browsers can display a large variety of content, but, in some cases, they also need help. For example, browsers can easily display text and graphics. For some functions, however, the browser relies on other software, with the browser placing the other software's display window inside its window. For example, a browser might not directly support the ability to show a video, but it might instead use software that plays a video. The Microsoft IE browser, for example, might use the Microsoft Windows Media Player (WMP) to show a video.

Downloading a Web Page

Computers store long-term data on disk drives in an object called a file. Computers, in their most basic form, work with *binary digits*, which people commonly abbreviate to *bits*. However, when describing computers, it is cumbersome to discuss and work with every little bit. So, computers combine 8 bits into a *byte* for a small amount of added convenience. The next step is to combine a set of bytes into a file. Computer files hold a set of related information. For example, a single computer file might hold a homework assignment you typed, a graphical image, a song that an MP3 player can play, a video, or any other single entity that computer software might want to manipulate or use.

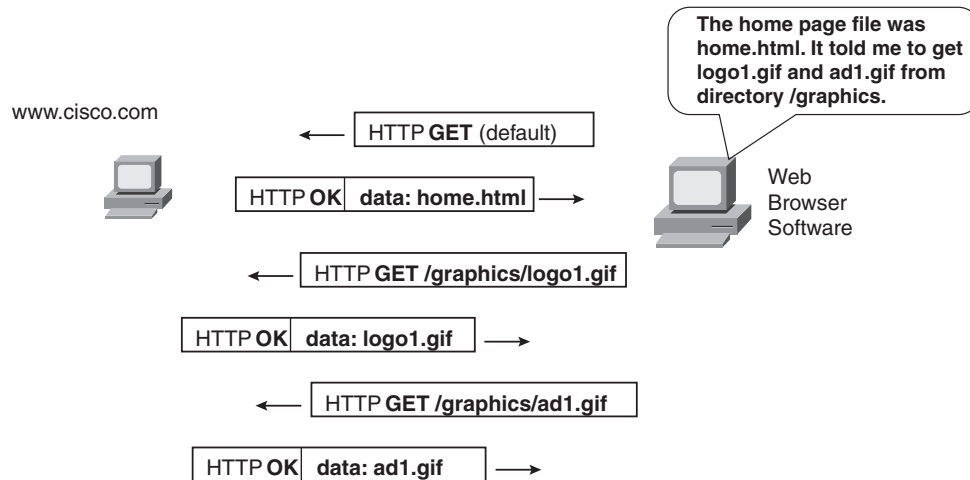
When a web developer creates a web page, the result is a set of files stored on the web server. One or more files might hold the web page's text. Other files hold graphics (typically, one file per graphical image). Other files might hold audio that plays when you load the web page.

When a browser loads a new web page, the following sequence of events occurs:

1. The browser asks the server to send one file that has both instructions and displayable content.
2. The browser displays the contents of the file.
3. The browser also looks at the instructions inside the file, which might tell it to get more files from the web server.
4. The browser asks the server for the additional files.
5. The browser displays the additional content, as well as looking for additional instructions to download other files.
6. The browser continues to look for instructions to download other files that are part of the web page until all files are downloaded and displayed.

To actually request the files and to cause the files to be transferred from the server to the web browser, the browser and server both use HTTP. Figure 1-16 depicts the flow and logic of how the web browser transfers files.

Figure 1-16 HTTP Transfers Three Files



HTTP uses the concept of a *get request*. The HTTP get request identifies the file that the browser needs from the server. The server obliges and sends the file. In this case, the first file, called home.html, holds instructions telling the browser to ask for two more files.

Note

The term HTTP is derived from the first type of file supported by a web browser: a file with Hypertext Markup Language (HTML) text and instructions. Original web browsers needed to download HTML files exclusively and, to do so, they needed a protocol to transfer the HTML files. So, HTTP is derived from the idea of a protocol to transport hypertext.

Note

Flash Player from Macromedia is required before you can see all the content of the online course. It can be downloaded from www.macromedia.com.

Browser Plug-Ins

Browsers know how to display the information in HTML and other types of files. However, rather than having to understand how to display any and every type of content, oftentimes the browser uses another program to display some types of content. For example, browsers often use a program called Flash Player (www.macromedia.com) to display animations like what's shown in the online curriculum. For video or audio, browsers often use products such as QuickTime (www.quicktime.com), RealPlayer (www.real.com), or WMP (www.microsoft.com).

When a browser needs to use one of these programs, it typically displays the related information inside the browser window on the PC display screen. In effect, this additional program is “plugged in” to the browser window. As a result, many people refer to these programs as *plug-ins*.



Lab 1.1.8 Web Browser Basics

In this lab, you learn how to use a web browser to access Internet sites, become familiar with the concept of a Universal Resource Locator (URL), and use a search engine to locate information on the Internet. You access selected websites to learn the definitions of networking terms and use hyperlinks to jump from a current website to other websites.

TCP/IP Networking Model

As previously mentioned, TCP/IP consists of a large number of protocols. In fact, the name TCP/IP refers to two of the more popular protocols inside TCP/IP: namely, *Transmission Control Protocol (TCP)* and Internet Protocol (IP). Because TCP/IP contains such a large volume of protocols, it is useful to think about the TCP/IP protocol suite by grouping its member protocols into categories called layers. Figure 1-17 shows the TCP/IP networking model and its component layers.

Figure 1-17 TCP/IP Networking Model and Protocols

TCP/IP Model	TCP/IP Protocols
Application	HTTP, SMTP, POP3
Transport	TCP, UDP
Internet	IP
Network Interface	Ethernet, Frame Relay, PPP

The TCP/IP networking model, like other networking models, shows several layers on top of each other. Each layer implies a general category of service or function that the protocols at that layer perform. For example, the application layer provides services to applications. Web browsers are concerned with displaying information on the screen, but to do so, they ask for help from a TCP/IP application layer protocol (namely, HTTP).

Chapter 2 covers the details about networking models in general, referencing another popular networking model called the OSI reference model. Chapter 9 covers the details about the TCP/IP networking model.

This chapter covers the most basic features of IP, TCP, IP routing, and IP subnetting; however, the online curriculum does not cover these topics until Modules 9 and 10. The book has included this additional coverage of these very practical parts of networking to balance the highly-theoretical coverage of the early chapters of this book. Also, many details in this section relate to the troubleshooting tools introduced in module 1 of the course.

If you prefer to skip topics that are not mentioned in the online curriculum, skip forward to the section “Troubleshooting Basics for IP.”

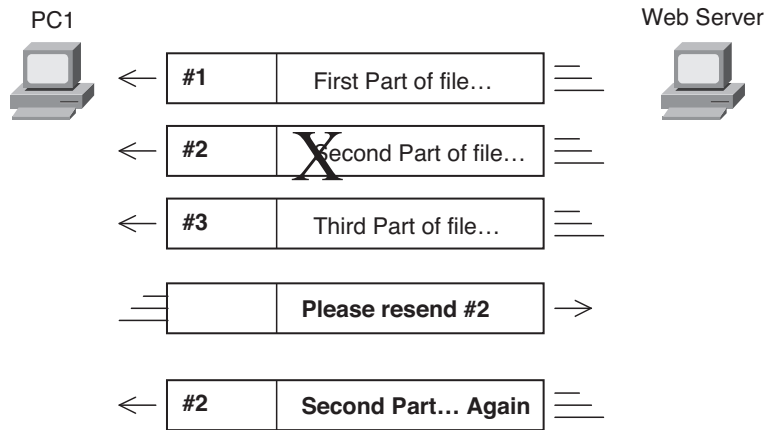
TCP/IP Transport Layer and the TCP Protocol

The TCP/IP transport layer is a group of protocols that provide services to application layer protocols. In fact, each layer of any networking model provides services to the layer just above it. This section describes one example of how a protocol at one layer provides a service to the adjacent higher layer by using the TCP and HTTP protocols in the example.

TCP happens to provide an important and popular service to many application protocols: the service of guaranteed delivery of data. Using HTTP (application layer) and TCP (transport layer, one layer below the application layer) as examples, consider the difference in logic used by the software on a web server:

- **HTTP**—I need to transfer files to another computer’s web browser, but I do not have any way to recover data in case it gets lost.
- **TCP**—I have the capability to monitor transmitted data to determine whether the data arrived. If it gets lost, I can resend the data, which guarantees that all the data eventually arrives safely.

As you can see, the two protocols seem like they were made for each other. In fact, TCP predated HTTP by quite a few years, so when HTTP was created, the people who created HTTP simply decided to use TCP to perform guaranteed delivery of the data through TCP’s error-recovery process. Figure 1-18 shows how the actual error-recovery process works.

Figure 1-18 Example of TCP Error Recovery

The TCP software, built in to the OS on the server, marks each packet with a *sequence number*, as Figure 1-18 shows. Because the second packet was lost, the PC on the left receives only the packets numbered 1 and 3. From that, the TCP software on the PC on the left—again, software built in to the OS on the PC—can surmise that packet 2 was lost somewhere along the way. To cause the server to resend the lost packet, PC1's TCP software sends a request to the server to resend packet 2, which it does.

The next section introduces IP, which provides the service of routing the packets from one computer to another.

TCP/IP Internet Layer and the Internet Protocol

The single most important protocol in the TCP/IP networking model is IP. It exists as part of the internet layer in the TCP/IP networking model. IP defines many things, but two features stand out by far as the most important. A large percentage of your work, thought, and learning in the Networking Academy CCNA classes somehow revolve around IP and its main two features:

- Routing
- Logical addressing

These features are linked in many ways, so they are best understood together. Although this chapter provides an introduction to networking, including networking software that implements TCP/IP, learning the basics of IP now helps provide some insight into where you will go in this class.

IP Addressing and Routing

Networks that use TCP/IP, which includes almost every network today, assign an *IP address* to each network interface. The IP address uniquely identifies that interface inside the network. After each network interface has a unique IP address, data can be sent from device to device by using these IP addresses, delivering the data to the one device that uses a particular IP address.

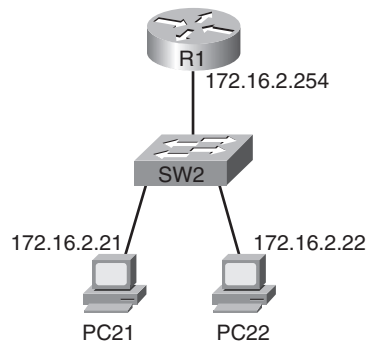
The preceding paragraph, although accurate, is relatively generic, so some concrete examples can help. IP addresses are 32-bit binary numbers. To make things easier on us humans, IP addresses might be written in *dotted-decimal* form. Dotted decimal means that the 32-bit number is represented by four decimal numbers separated by periods (dots). Each decimal number represents 8 bits. For example, the following line shows the same IP address in both binary form and dotted-decimal form:

00001010000000010000001000000011 10.1.2.3

Certainly, it is much easier to keep track of the dotted-decimal version of the IP address than the binary version of the same IP address. In this case, the decimal 10 represents the first 8 bits, the 1 represents the next 8 bits, and so on. For example, the 8-bit binary equivalent of decimal 10 is 00001010, which begins the 32-bit IP address just shown.

Now, consider the simple LAN shown in Figure 1-19. It has two PCs, each with an IP address.

Figure 1-19 Single LAN with Two PCs and One Router

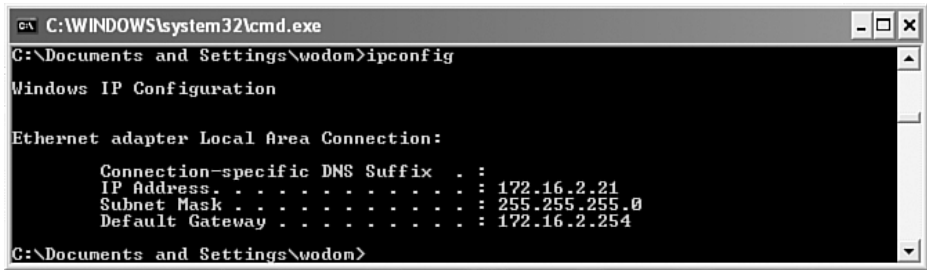


IP defines that each network interface that uses IP must have an IP address. In Figure 1-19, each PC has a single Ethernet LAN NIC. A NIC provides an interface to the network so the PCs can assign an IP address to the NIC. In fact, you can easily see the IP address assigned to a NIC on most any computer, if you know the details. On Microsoft Windows XP, for example, you can use the **ipconfig** command from a command prompt, as shown in Figure 1-20. Figure 1-20 shows the **ipconfig** command output from PC21 in Figure 1-19.

Tip

The four decimal values inside an IP address each represent 8 bits. Most people use the term *octet* to refer to a single one of the four numbers inside an IP address. The decimal values must be between 0 and 255 and inclusive. Chapters 9 and 10 cover additional restrictions.

Figure 1-20 Sample **ipconfig** Output



The **ipconfig** command shown in Figure 1-20 can be performed using the Packet Tracer tool in Real-time mode. Feel free to experiment with the NA01-0112 configuration previously used in this chapter, and look at the IP addresses on Fred, Barney, Wilma, and Betty.

Using these IP addresses, the PCs can send each other packets of data. The packet includes end-user data and headers added by some other protocols, including the IP header added by the IP software on a computer. The IP header includes a source IP address and a destination IP address.



Lab 1.1.6 PC Network TCP/IP Configuration

In this lab, you learn the methods of discovering your computer’s network connection, hostname, MAC (Layer 2) address, and network (Layer 3) address.

Note

IP addresses are considered to be logical addresses. The term logical is not meant to imply that other addresses are “illogical,” but rather physical. For example, a MAC address is permanently associated with a single physical NIC, so a MAC address is a physical address. An IP address can be assigned to any PC, so IP addresses are not physical.

IP Address Organization: Subnets

Figure 1-21 adds a router and two switches to the network shown in Figure 1-19. In this network, the router is connected to three LANs, so it has three network interfaces, one connected to each LAN. Therefore, the router has an IP address for each interface because IP addresses uniquely identify interfaces used to connect to a network.

The IP addresses in Figure 1-21 must conform to certain rules. In this example, all IP addresses on the same LAN must use the same first three decimal numbers in their respective IP addresses. Figure 1-22 shows how the numbers are grouped.

Figure 1-21 Network with Three LANs Connected by One Router

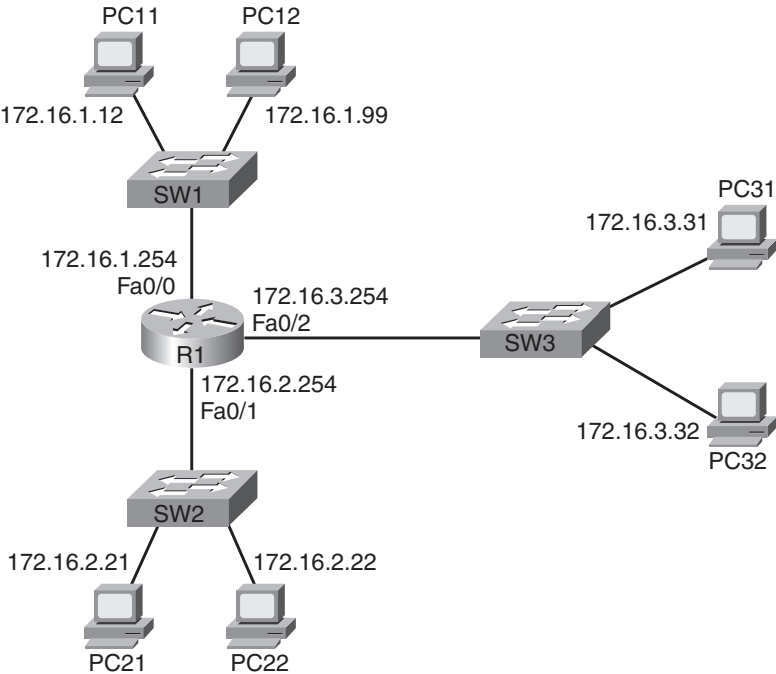


Figure 1-22 Grouping Effect of IP Addresses

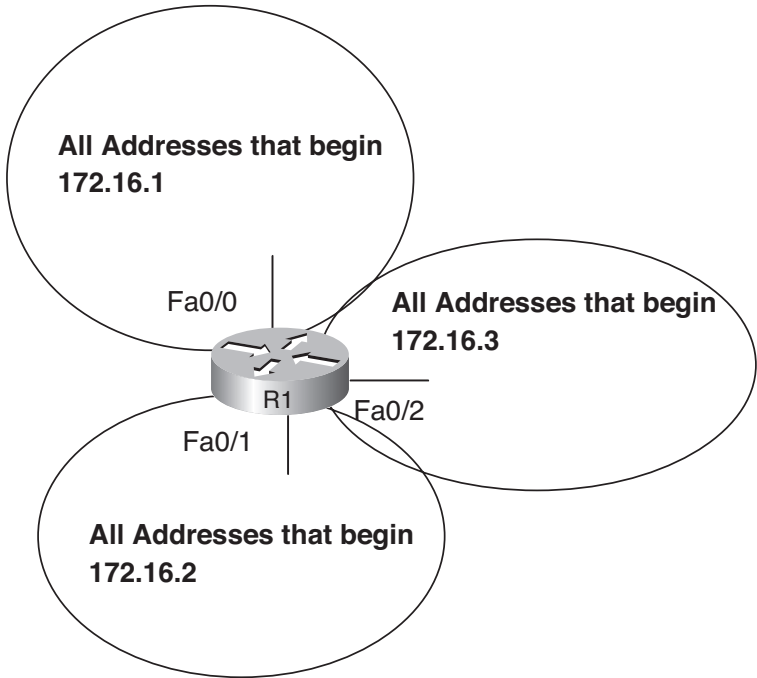


Figure 1-22 shows three separate IP subnets. A *subnet* is a group of IP addresses that share a common value for the beginning parts of the IP addresses in the subnet. Also, the members of a subnet must not be separated from each other by any routers. The three subnets in Figure 1-22 are loosely described as follows:

- All IP addresses beginning with 172.16.1
- All IP addresses beginning with 172.16.2
- All IP addresses beginning with 172.16.3

IP Routing with IP Routers

The rules about the IP addresses shown in Figure 1-22 allow for easy routing. The router keeps a table—called a routing table—that essentially lists all the groups of IP addresses that it can somehow reach and the interface out which it needs to send packets to reach those groups. For example, for Figure 1-22, router R1’s routing table contains the equivalent of what’s shown in Table 1-1.

Table 1-1 Pseudo-Routing Table on Router R1 in Figure 1-22

To Forward Packets Sent to Addresses That Begin With...	Send the Packets Out This Interface
172.16.1	Fa0/0
172.16.2	Fa0/1
172.16.3	Fa0/2

Armed with the information in Table 1-1, the router can receive data packets sent by any of the PCs in the network and make the correct decision about where to forward the packets.

Troubleshooting Basics for IP

Now that you know the very basics of IP addresses and routing, the rest of this section covers troubleshooting. Frankly, it is a bit ambitious to think about troubleshooting before covering more of the course. However, you can do some basic things that are both interesting and fun from any network-connected PC.

The online curriculum uses a how-to list, similar to the following one, that suggests general steps in how to approach problems when you troubleshoot:



- Step 1** Define the problem.
- Step 2** Gather the facts.
- Step 3** Identify possible solutions by doing the following:

Note
Fa is short for Fast Ethernet, which is a type of Ethernet LAN that sends bits at 100 Mbps.

Note
For those of you skipping the extra coverage of TCP, IP, IP subnetting, and IP routing, begin reading again at this heading.

- 1 Analyze the facts compared to the expected behavior of the network.
 - 2 Determine possible root causes of the unexpected behavior.
 - 3 Determine a set of actions that might solve the problem.
- Step 4** Create an action plan to implement the solution you chose in Step 3.
- Step 5** Implement the plan.
- Step 6** Observe the results.
- Step 7** Document the details.
- Step 8** Introduce problems and troubleshoot.

One of the most valuable tools when troubleshooting any network is to have ready knowledge or references that describe how the network is supposed to work. The computers and network devices that comprise a network do attempt to follow the protocols, and these protocols define a series of steps that must be taken under certain conditions. If you know the steps that should occur, you can try to find the first step that is not working correctly and then look for reasons why that step failed.

This section describes some of the basics of IP addressing and routing, to complete the picture of what should happen, along with some tools that are useful for troubleshooting.

Name Resolution Using a Domain Name System Server

Although a basic understanding of IP addressing is useful at this point in the class, most computer users never even think about IP addresses. However, they do know the names of things that can be translated into IP addresses. For example, you might see an advertisement on TV or in a magazine that mentions a website. The ad might list something such as `www.get-cisco-certified.com`. You might open a web browser and plug that into the field called *Address* at the top of the browser, and the website suddenly appears. Or you might already be on some website and click something on the screen, and another website appears. When you do that, the browser's Address field changes because your clicking action causes the web browser to jump to another web address.

The correct term for the text placed into the Address field of a web browser is *Universal Resource Locator (URL)*. Many people simply call this a web address. The URL itself has some structure. For example, the following URL represents the web page from which you can look at all the Cisco Systems documentation:

`http://www.cisco.com/univercd`

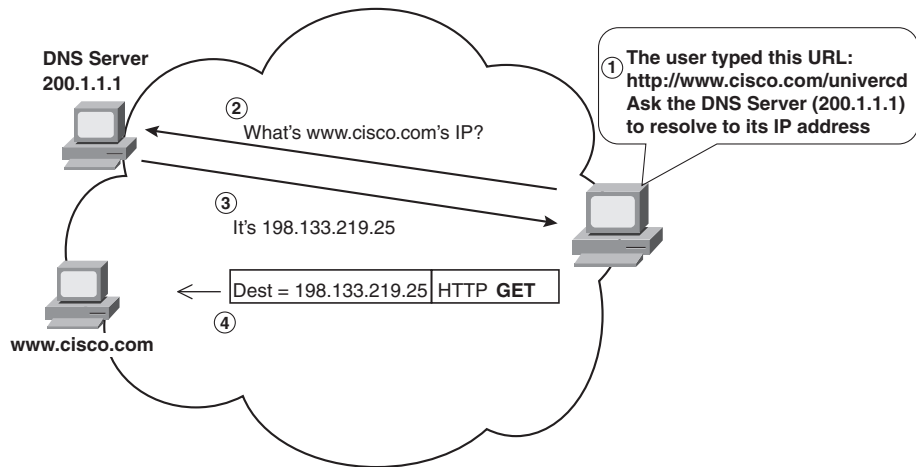
By closely examining that URL, you can separate the URL into its components. First, the stuff before the double slash simply identifies the protocol to use, which, in this case, is the

now-familiar HTTP protocol. The web server uses the stuff after the single slash to further identify the specific web page that the browser wants to see. The middle part of the URL—the part between the double slash and single slash—is called the hostname of the web server. When a PC user opens a browser and attempts to reach a URL, the following occur:

1. The PC finds the hostname inside the URL.
2. The PC requests name resolution from a Domain Name System (DNS) server to find the IP address of the server whose hostname is in the URL.
3. The DNS server supplies the IP address being used by the web server.
4. The PC can then send packets (like those containing HTTP get requests) to that IP address.

The overall process of asking a DNS server to supply the IP address associated with a name is called *name resolution*. Figure 1-23 shows an example of name resolution, with the steps in the preceding list referenced in the figure. (Chapter 9 covers DNS in more detail.)

Figure 1-23 DNS Resolution After Putting a URL into a Web Browser



Although Figure 1-23 shows what happens when you browse the web, similar things happen with other protocols. For example, when you send an e-mail from your PC, your e-mail software sends the e-mail to an e-mail server. Your e-mail software is configured with a reference to the name of the e-mail server (for example, `mail.skyline-ats.com`). So, before sending an e-mail to the e-mail server, your PC must ask DNS to tell your computer the IP address of the e-mail server based on its name.

Default Gateway/Router

After it resolves a hostname into its corresponding IP address, a PC must next make a simple but important decision: Is the destination IP address on my same subnet or not? As previously mentioned, a subnet is a group of IP addresses that have the same beginning, or prefix, in their

IP addresses. All the hosts on the same LAN should be in the same subnet. Also, by definition, hosts on different subnets should be separated from each other by at least one router. So, for a PC connected to a LAN, its logical next step can be separated into the following statements:

- If the destination IP address is in my subnet, I do not need to send the packets to a router; I can send them directly over the LAN to the destination.
- If the destination IP address is in another subnet, at least one router exists between me and the destination. Routers are in charge of packet delivery, so I must send the packets to a router that is attached to the LAN.

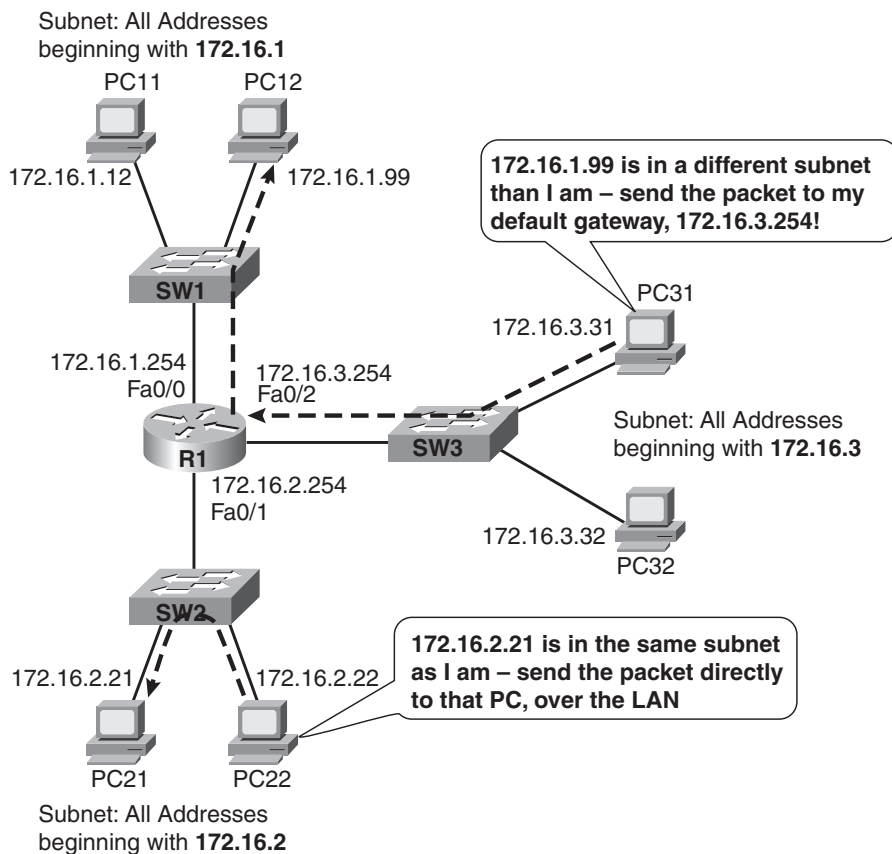
To perform the second step in this list, a PC uses a concept called a **default gateway** (also known as a default router). A PC's default gateway is the IP address of a local router, one on the same subnet, to which the PC sends all packets destined for another subnet.

Figure 1-24 shows an example in which one host needs to send a packet to another host on the same subnet, and a second host needs to send a packet to another host in a different subnet.

Tip

When the first routers were created, they were called gateways. In fact, the first Cisco Systems commercial routers had a G (meaning gateway) in the name instead of an R. Although the industry has been using the term router instead of gateway for a long time, the concept described in this section is still more often called default gateway instead of default router. So, it's useful to remember both terms and know that they mean the same thing.

Figure 1-24 Sending Packets to the Same Subnet or Different Subnet



Note

The examples in this section use PCs attached to LANs, but PCs that use modems, DSL, or cable to access the Internet use the same concept of a default gateway.



On PCs using Windows XP, the default gateway IP address is listed in the output of the **ipconfig** command. (Refer to Figure 1-20 for an example.) You can use any working PCs in the classroom to look at their default gateway IP addresses. You can also download the Packet Tracer configurations from www.ciscopress.com/title/1587131641 and then load the one named NA01-0124, which loads a configuration like the one shown in Figure 1-24. Then, you can use the **ipconfig** command on the PCs in the network and perform a simulation of the packets shown in Figure 1-24. Note that other OSs also have similar commands, such as **ifconfig** on Linux.

Note

The labs that come with the online curriculum typically use a convention by which the router IP address in each subnet uses the lowest number(s) in the subnet. If Figure 1-24 followed that convention, its IP addresses would have been 172.16.1.1, 172.16.2.1, and 172.16.3.1. However, that is just a convention; the router and all other computers on a subnet can use any valid IP addresses in that subnet.

Troubleshooting Tools

When a user does something on his PC that makes the PC want to send data, a couple of things happen, in order, as described in the preceding section. This section and the next one describe a few basic troubleshooting steps. First, for reference, here are the first three basic steps that you will examine now. Other related steps will be covered after you get into more detail in the class:

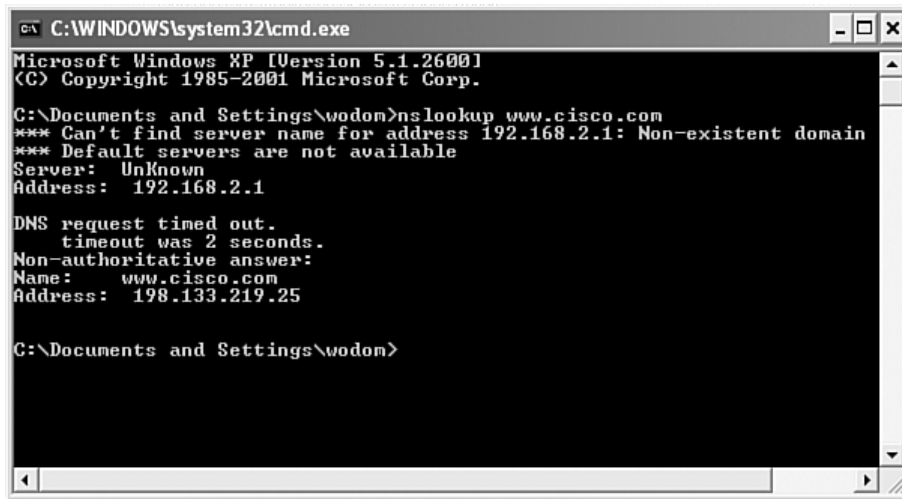
1. When the user types/implies a name of another computer (for example, by choosing to go to a website with the name embedded in the URL), the PC asks DNS to resolve the name into an IP address.
2. After it knows the IP address, if the IP address of the other computer is local, the PC directly sends the packet to the other PC.
3. After it knows the IP address, if the IP address of the other computer is on another subnet, the computer sends the packet to its default gateway.

A reasonable methodology for troubleshooting is to somehow verify whether the PC can succeed at Step 1. If that succeeds, somehow verify if Step 2 is working, and so on. Most PC OSs include built-in tools that allow such testing, such as **nslookup**, **ping**, and **tracert**.

nslookup

You can use the **nslookup** command from a command prompt on many PC OSs, including Windows XP. The name stands for *Name Server Lookup*, which means that the command does the same sort of DNS request/lookup that would be done by a web browser when looking for a web server. Figure 1-25 shows sample output from **nslookup** on the PC in my office, looking for www.cisco.com. As you can see in Figure 1-25, www.cisco.com successfully resolved to IP address 198.133.219.25.

Figure 1-25 Sample nslookup Output



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\wodom>nslookup www.cisco.com
*** Can't find server name for address 192.168.2.1: Non-existent domain
*** Default servers are not available
Server: Unknown
Address: 192.168.2.1

DNS request timed out.
    timeout was 2 seconds.
Non-authoritative answer:
Name:   www.cisco.com
Address: 198.133.219.25

C:\Documents and Settings\wodom>

```

ping

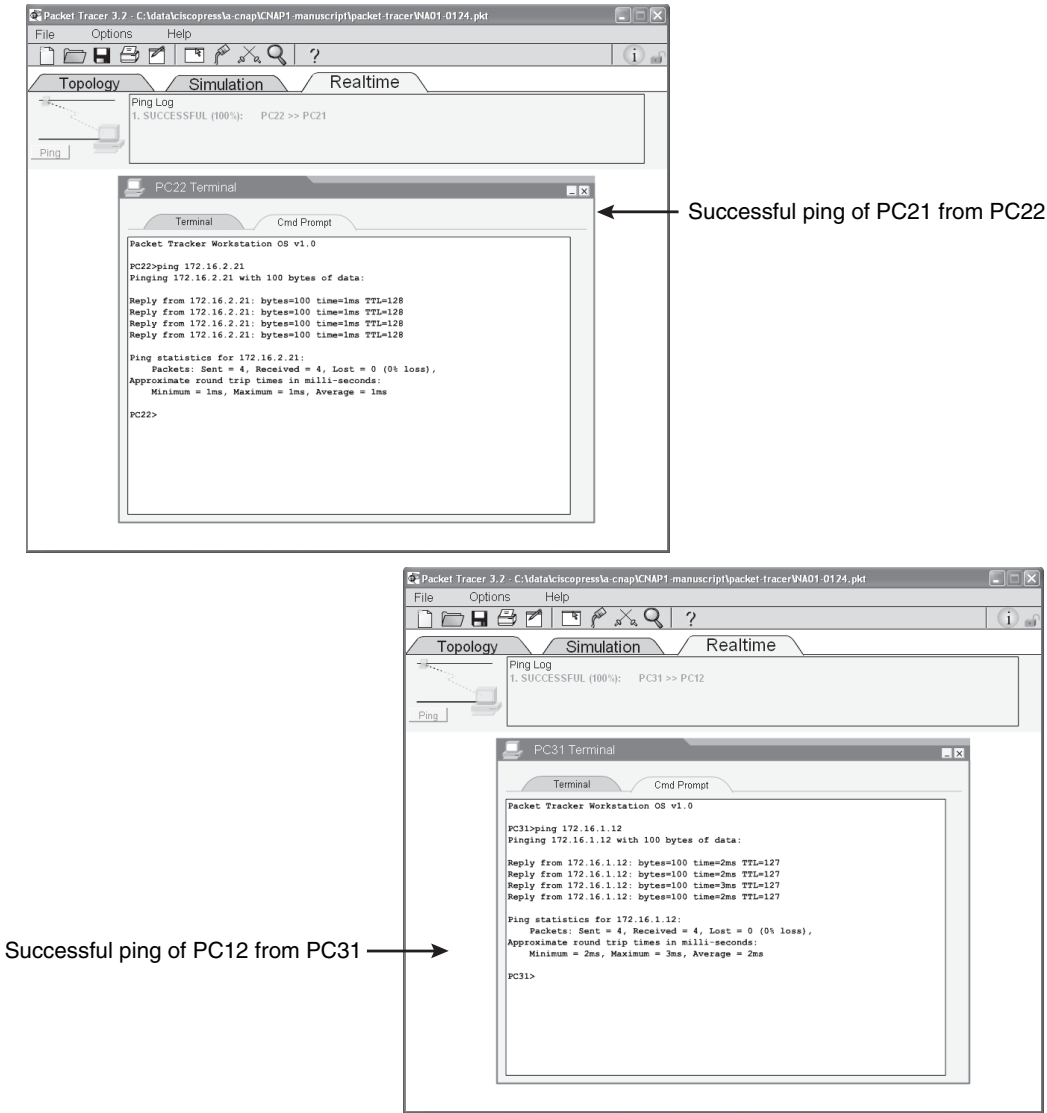
The **ping** command tests a PC's capability to successfully send a packet to another IP address and receive a response. The **ping** command sends a special packet, popularly called a *ping request packet*, to the IP address listed after the **ping** command. Computers that implement TCP/IP are required to support the capability to receive something loosely called a ping request packet and send a *ping reply packet* back to the original sender. If the **ping** command receives the ping reply packet, the command has verified that the two computers can send and receive packets to and from each other.

Figure 1-26 shows sample **ping** command output from Packet Tracer. With the Packet Tracer tool, you can simulate a **ping** command by using the Simulation tab and then clicking the PC. Figure 1-26 shows PC22 pinging PC21 and PC31 pinging PC12, as shown in Figure 1-24.



You can perform the same command, using Packet Tracer, by opening file NA01-0124, clicking the **Realtime** tab, double-clicking **PC22**, and issuing the **ping 172.16.2.21** command. Similarly, double-click **PC31** and issue the **ping 172.16.1.12** command to test the other route shown in Figure 1-24.

Figure 1-26 Sample Pings



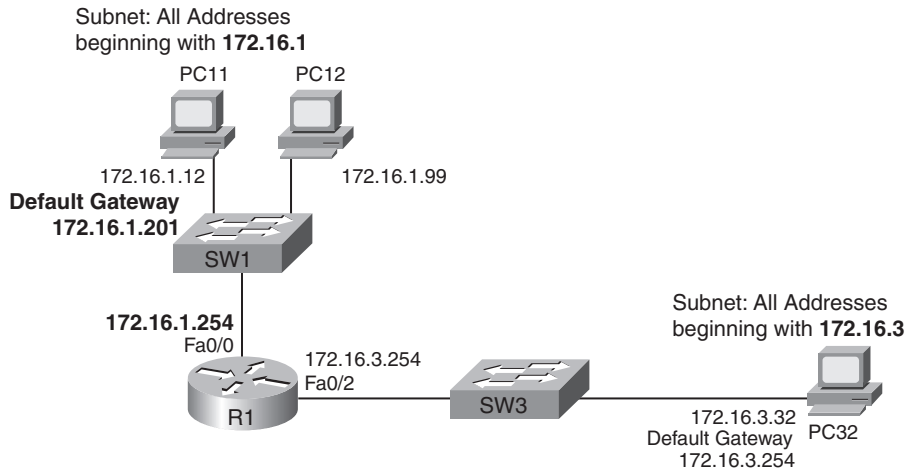
The **ping** command can also test the capability of a PC to send packets to its default gateway. This step might be particularly useful to verify whether a PC can send packets to another subnet. Figure 1-27 shows a one-router, two-LAN network; however, this time, PC11 has the wrong default gateway configured. Because of this misconfiguration on PC11, PC11 cannot successfully send packets to hosts on other subnets. However, it can send packets to hosts on the same subnet because PC11 does not need to use its default gateway to reach other hosts. Figure 1-28 shows some of the steps involved in troubleshooting PC11 (this time from a screen shot using Packet Tracer).

As you can see from Figure 1-28, PC11 cannot even ping its default gateway. If a PC cannot ping its default gateway, it cannot successfully send packets through that default gateway, which means that it cannot send packets outside the local subnet.



By loading Packet Tracer Configuration NA01-0124 and using real-time mode, you can duplicate the ping tests shown in Figure 1-28.

Figure 1-27 PC11 with a Misconfigured Default Gateway



When you use the **ping** command to troubleshoot a problem, a simple sequence can be used to find out how far packets can be delivered into the network. For example, one step is to ping a PC's default gateway IP address, as previously mentioned. The following list summarizes the steps in the order that they are normally used:



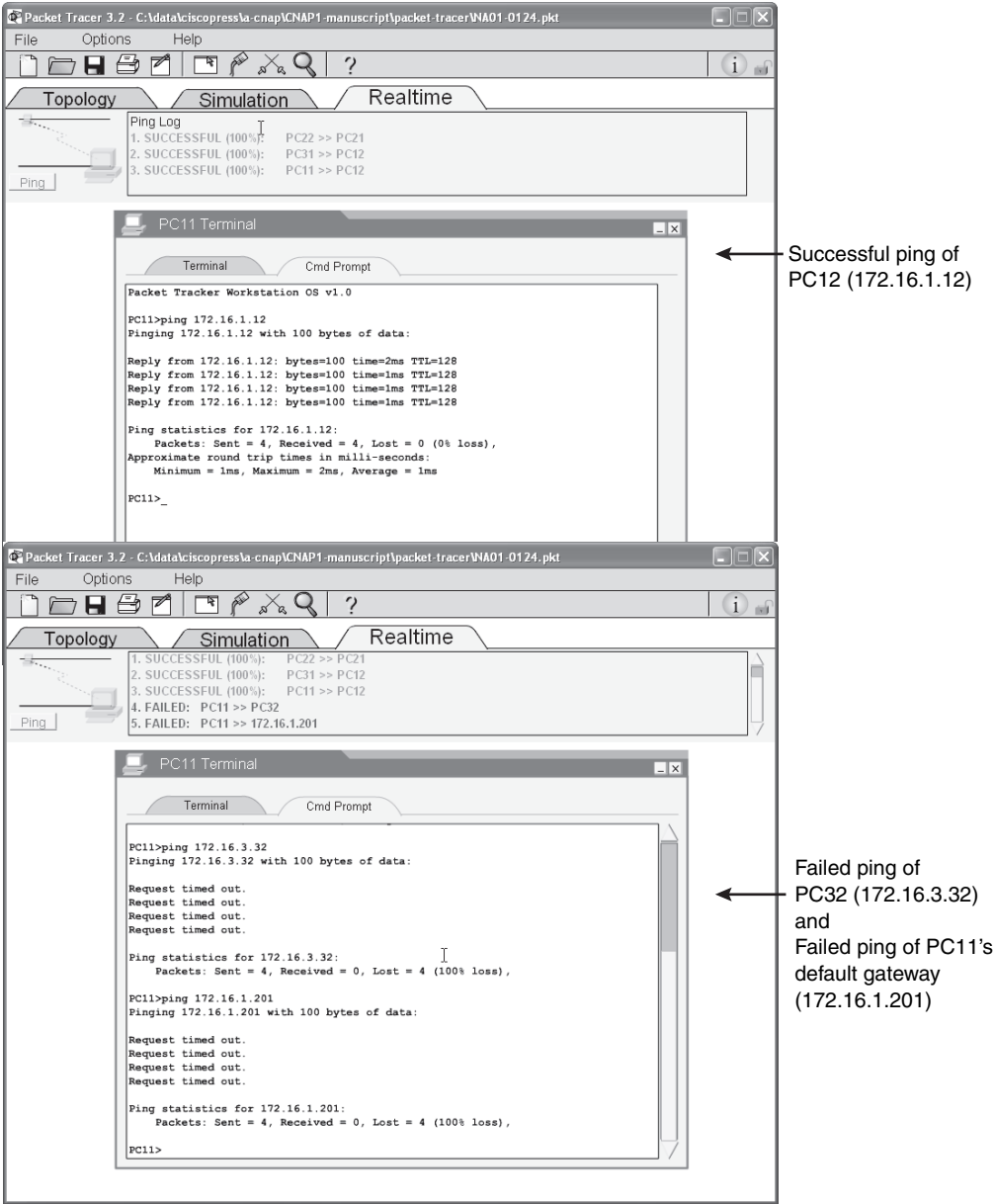
Step 1 ping 127.0.0.1—Sends a ping down the software and backup. It simply tests the software on the local computer. (IP address 127.0.0.1 is called the loopback IP address. It is automatically configured every time TCP/IP is installed and is reserved for this self-test purpose.)

Step 2 ping the PC's own IP address—Tests whether the PC can use its own NIC.

Step 3 ping the default gateway—Tests connectivity over the LAN, whether the PC has referenced an IP address of some default gateway, and whether that default gateway's LAN interface is up and working.

Step 4 ping the destination computer—Tests the complete path between the PCs.

Figure 1-28 Troubleshooting PC11’s Inability to Ping PC32



tracert

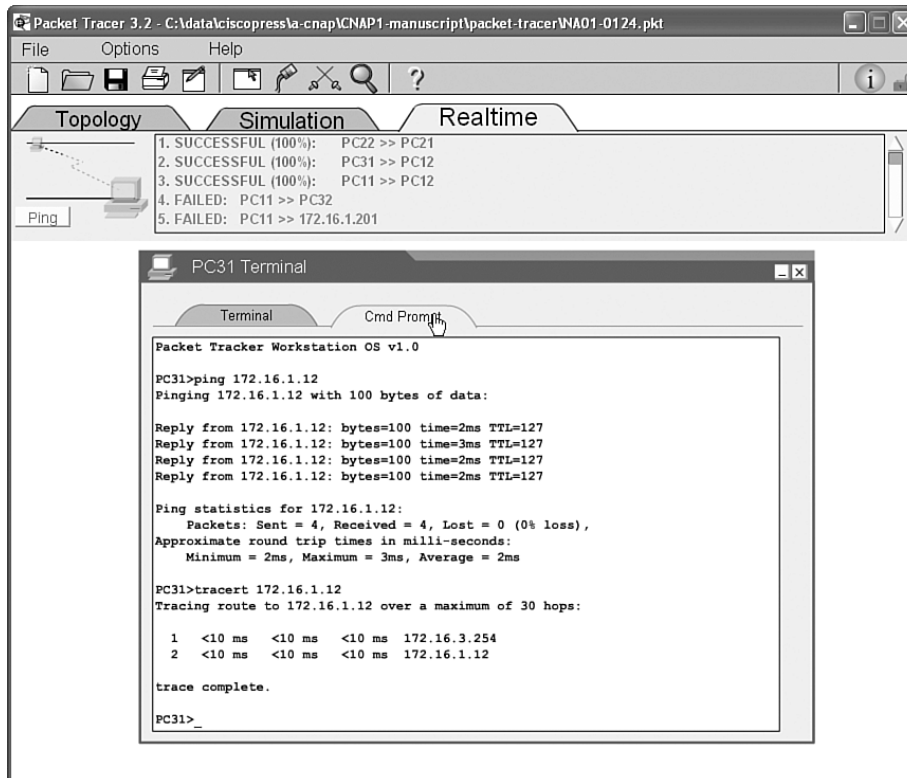
The `tracert` tool, typically pronounced “trace route,” is found on many OSs and traces the route a packet takes through a network. The actual name of the command differs depending on the OS. In Microsoft OSs, the actual command name is `tracert`; on Cisco routers, Linux, and

UNIX, it is **traceroute**. Although you have been introduced to only the most basic parts of IP routing, it is good to know something about the **tracert** command when you begin your study of networking. You can actually learn much about routing just by experimenting with the **tracert** command on sample networks in the Packet Tracer tool and on real PCs attached to working networks.

tracert sends packets through the network to discover the IP addresses—and sometimes names—of the routers between one computer and another. For example, when PC31 pings PC12 from Figure 1-24, the command works, with the **tracert** command listing the IP addresses of any intermediate routers. Figure 1-29 shows this exact example.

Figure 1-29 shows two important lines of output from the **tracert** command. The first line lists the first router in the route: 172.16.3.254. This IP address is the router's IP address on the same LAN as PC31. In effect, this line of output means that PC31 first sends the packet to the router. The second line of output lists PC12's IP address of 172.16.1.12. This line means that PC12 itself was the next device that received the packet.

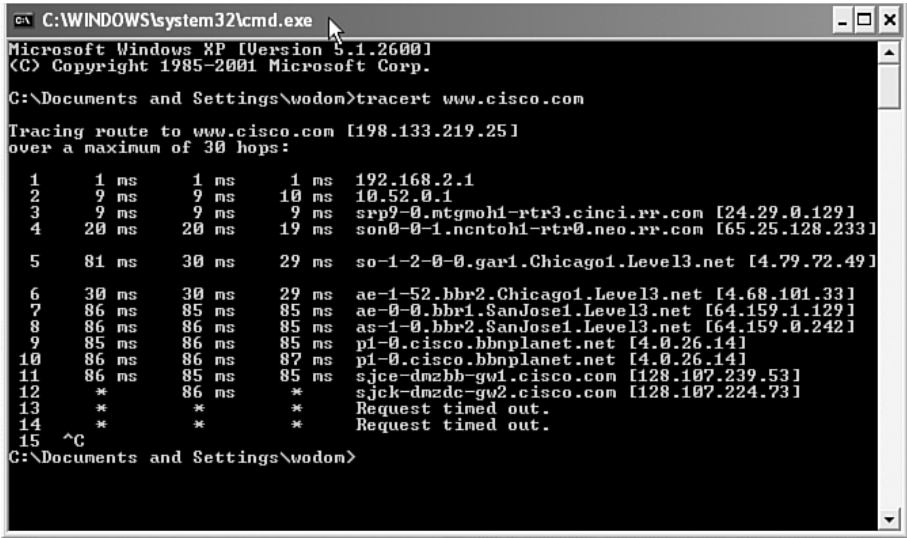
Figure 1-29 tracert on PC31 to PC12



What’s not listed in the output might be just as interesting. You know from Figure 1-24 that only one router is in the figure. The **tracert** command output confirms this fact. If there were three routers between PC31 and PC12, the **tracert** output would have listed a line of output for each router in the path as well as an ending line listing the IP address of the destination host.

If the DNS server is working correctly, **tracert** can also list the names of the devices. Packet Tracer does not have a DNS feature, so Figure 1-29 lists only the IP addresses. However, you can sit at a PC connected to the Internet, use the **tracert** command, and see many routers and their hostnames listed in a route. For example, Figure 1-30 shows a **tracert www.cisco.com** command issued on my desktop PC.

Figure 1-30 tracert from a PC to www.cisco.com



Tip

If the **tracert** command continues running and never completes, use the Ctrl-C key sequence to try and stop the command.

As you can see from the output listed in Figure 1-30, many routers sit between my PC and the server whose name is **www.cisco.com**. Also, the output shows that the command could not determine the last few routers, which are noted as asterisks in the last few lines of output.



Lab 1.1.7 Using ping and tracert from a Workstation

In this lab, you learn to use the TCP/IP **ping** and **tracert** commands to test connectivity in a network. In the process, you see name resolution occur.



Lab 1.1.9 Basic PC/Network Troubleshooting Process

In this lab, you apply the basic troubleshooting model to simple and common network problems. You also become familiar with the more common hardware and software problems.

TCP/IP Wrap Up

The preceding section covering TCP/IP introduced many concepts and answered several questions. However, it might also have created even more questions because, as in any introduction, many details were omitted. However, many of these questions are answered—even in the CCNA 1 course.

Some of the processes, concepts, and troubleshooting steps actually require some binary, decimal, and hexadecimal math. To prepare you for those upcoming topics, this chapter concludes with a section that covers these three numbering systems and how to convert between them.

Network Math

Understanding binary, decimal, and hexadecimal numbering systems is important to many aspects of working with computer networking and with computing in general. Binary numbering (Base 2) is necessary because it can represent the most basic operations on computers: operations that work with binary digits (called bits). Hexadecimal numbering (Base 16) is necessary because binary can be slightly difficult to work with, and hexadecimal numbering can easily represent those same binary numbers. Of course, the ability to work with decimal numbering (Base 10) is important because that's what we humans are accustomed to working with.

Bits and Bytes

At their most basic level, computers work with bits. Physically, these bits might exist in several states. For example, computer RAM (memory) consists of several chips, and the chips hold millions of little transistors. The transistors are components of a chip that can be placed into either an on or off state. The computer considers the off state to mean binary 0 and the on state to mean binary 1. Other parts of the computer might use other physical and electrical methods to store bits. (For example, a disk drive might store a magnetic charge onto the disk, with one type of magnetic charge meaning binary 0 and another meaning binary 1.)

Computers can also work with combinations of bits, the most common being an 8-bit byte. Some computers also use the term *word*, which refers to multiple bytes (typically, 4 bytes). The computer hardware might work with a byte at a time, a word at a time, or even a bit at a time, depending on what the computer hardware attempts to do.

Names and Abbreviations for Large Numbers of Bits and Bytes

Computers might process extremely large amounts of bits and bytes, so additional terms are needed to describe these large chunks of data—terms slightly more exact than *large chunk*, that is. Table 1-2 lists the terms and describes how many bits and bytes each term represents.

Table 1-2 Names and Units for Large Numbers of Bits and Bytes

Term	Number of Bits	Number of Bytes
Kilobit (Kb)	1000	125 (1/8th of 1000)
Kilobyte (KB)	8000 (8 * 1000)	1000
Megabit (Mb)	1,000,000	125,000 (1/8th of 1 million)
Megabyte (MB)	8,000,000 (8 * 1,000,000)	1,000,000
Gigabit (Gb)	1 billion	125 million (1/8th of 1 billion)
Gigabyte (GB)	8 billion (8 * 1 billion)	1 billion
Terabit (Tb)	1 trillion	125 billion (1/8th of 1 trillion)
Terabyte (TB)	8 trillion (8 * 1 trillion)	1 trillion

The abbreviations use a lowercase “b” when referring to bits and an uppercase “B” when referring to bytes. This convention is used throughout computing. Interestingly, when working with computing topics outside of networking, the terms that refer to bytes are used more often. However, networking usually refers to terms that refer to a number of bits.

Names for the Rate at Which a Network Sends Data

Networks transmit data from one device to another by using different transmission media. Depending on the media, and the type of device connected to the media, the device might send the data at a different rate. The names of these rates look similar to the names shown in Table 1-2, but in this case, the terms include the idea of some number per second. Table 1-3 lists the terms.

Table 1-3 Names and Units Transmission Rates

Term	Number of Bits per Second	Number of Bytes per Second
Kilobit per second (Kbps)	1000	125 (1/8th of 1000)
Kilobyte per second (KBps)	8000 (8 * 1000)	1000
Megabit per second (Mbps)	1,000,000	125,000 (1/8th of 1,000,000)
Megabyte per second (MBps)	8,000,000 (8 * 1,000,000)	1,000,000
Gigabit per second (Gbps)	1 billion	125 million (1/8th of 1 billion)
Gigabyte per second (GBps)	8 billion (8 * 1 billion)	1 billion
Terabit per second (Tbps)	1 trillion	125 billion (1/8th of 1 trillion)
Terabyte per second (TBps)	8 trillion (8 * 1 trillion)	1 trillion

When you download a file over the Internet, oftentimes a popup window appears that tells you the rate at which the file is being transferred. Note that when this happens, the units typically describe the number of bytes per second, not the number of bits per second. However, when an ISP sells its Internet access service, it typically describes the speed in bits per second (or Kbps or Mbps) because that is a measurement of the network’s speed. You can test the speed of your Internet connection by opening a web browser and going to http://reviews.cnet.com/Bandwidth_meter/7004-7254_7-0.html.

ASCII Alphanumeric Code

Most of the networking discussion in the next few chapters revolves around bits and bytes. Some of those bits and bytes represent numbers; in fact, most of the rest of this section discusses the important math behind manipulating bits and bytes as numbers. However, before discussing the numbers, this section discusses how computers represent text.

Computers represent text by using bits and bytes. Because text does not represent any particular number, however, computers need a way to correlate a particular set of bits to mean the letter “a,” another to mean the letter “A,” another for “b,” and so on. Such a convention is called alphanumeric code.

Today, the most popular alphanumeric code used by computers is *American Standard Code for Information Interchange (ASCII)*. For example, Table 1-4 shows a few capital letters and the 8-bit number that represents those letters on a computer.

Table 1-4 Sample ASCII Codes for Capital Letters A Through H

Letter	Binary ASCII Code	Decimal ASCII Code
A	01000001	65
B	01000010	66
C	01000011	67
D	01000100	68
E	01000101	69
F	01000110	70
G	01000111	71
H	01001000	72

Tip
The online course has an ASCII converter feature that allows you to type in any letter and see the ASCII equivalent (in decimal).

Table 1-4 shows the values as binary and decimal. Some of the upcoming sections describe how to convert any binary number, including ASCII binary codes, to decimal numbers for easier manipulation.

Decimal Numbering System: Base 10

Decimal numbering (Base 10) should be familiar because it's what you have been taught since early childhood. However, unless you love math, you probably have not thought about a few details because the concepts of decimal have become part of you. However, thinking about the following simple decimal concepts helps you better appreciate binary numbering, which is covered in the next section.

First, consider the number 235, for example. The number itself is made up of three numerals: 2, 3, and 5. *Numerals* are simply symbols that represent a number. The decimal numbering system uses numerals 0 through 9. The word *digit* (short for *decimal digit*) is often used instead of *numeral*. For example, 3 is the second digit of the number 235.

What does the number 235 really mean? Well, if you say the equivalent in English, you say something like, “two-hundred thirty-five.” To better appreciate how other numbering systems work, such as binary, consider a contrived and unusual expansion of the English-language version of 235:

Two 100s, three 10s, and five 1s

It's much easier to say “two-hundred thirty-five” than “two 100s, three 10s, and five 1s.” However, they both basically mean the same thing. You could even think of it in mathematical terms:

$$(2 * 100) + (3 * 10) + (5 * 1) = 235$$

Both the contrived English phrasing and the mathematical formula describe the core meaning of a multidigit decimal number. Each individual decimal digit represents its own value multiplied by a value associated with that digit's position in the number. It's more obvious to see this in a table, such as Table 1-5.

Table 1-5 Decimal Numbering: 1s, 10s, and 100s Digits

Powers of 10	10²	10¹	10⁰
Value Associated with That Digit or Column	100	10	1
Digits	2	3	5

With decimal numbering, the right-most digit in a number represents a value of that digit times 1. That digit is called the *1s digit*. The second from the right represents the value of the digit times 10. That digit is called the *10s digit*. The third from the right represents a value of that digit times 100. That digit is called the *100s digit*. This same logic continues for larger numbers; each successive digit to the left has a value 10 times the digit to its right. In Table 1-5, the 5 means “5 times 1” because it's in the 1s column. Similarly, the single digit in the 10s column represents “3 times 10.” Finally, the 2 in the 100s digit column means “2 times 100.”

Because you have used it all your life, the math is probably so intuitive that you don't need to think about decimal numbering to this depth. In the next section, you see how binary numbering works on the same basic premise, but with just two numerals or digits.

Binary Numbering System: Base 2

Binary numbering (Base 2) represents numbers in a different way than decimal (Base 10). Both decimal and binary numbering use numerals or digits to represent the idea of a particular number. However, binary uses just two digits: 0 and 1.

Binary numbering works on the same general principles as decimal numbering, but with differences in the details. The best way to understand the similarities and differences is to look at a sample binary number. Binary is simply another way to write digits that represent a number. For each decimal number, you can write the same number in binary. For example, the following binary number is the equivalent of the decimal number 235:

11101011

Similar to decimal, a multidigit binary number has assigned values for each digit in the number. Table 1-6 shows 11101011 with values assigned to each digit.

Table 1-6 Binary Numbering: 1s, 2s, 4s, 8s (and So On) Digits

Powers of 2	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
Value Associated with That Digit or Column	128	64	32	16	8	4	2	1
Number Itself	1	1	1	0	1	0	0	1

With decimal, the digits in a multidigit decimal number represent various powers of 10, with the right-most digit representing 10⁰, which is 1. With binary, the digits represent powers of 2, with the right-most digit representing 2⁰, which is also 1. As shown in Table 1-6, the right-most binary digit represent the number of 1s (2⁰), the second from the right represents the number of 2s (2¹), the third from the right represents the number of 4s (2²), and so on.

Table 1-6 shows the value associated with each digit (or column), with each being a consecutive power of 2, increasing from right to left. So, what does this mean? Well, just like the decimal number 235 means (2*100) + (3*10) + (5*1) = 235, the binary number 11101011 means the following, but written in all decimal numbers for clarity:

$(1 * 128) + (1 * 64) + (1 * 32) + (0 * 16) + (1 * 8) + (0 * 4) + (1 * 2) + (1 * 1) = 235 \text{ decimal}$

If you add up the numbers, you actually get 235 decimal. The number 235 (decimal) and 11101011 (binary) both represent the same number; they’re just written in a different format.

Converting From 8-Bit Binary Numbers to Decimal Numbers

Many times in networking and computing, it is convenient to work with a number in both its decimal and binary form. To do that, you need to be able to convert between the two formats. This section describes how to convert from binary to decimal. (The next section after that describes how to convert from decimal to binary.)

In networking, the most frequent reason to convert numbers from decimal to binary relates to IP addresses. IP addresses are indeed 32-bit binary numbers, but they are frequently written in dotted-decimal notation. With dotted decimal, each decimal number represents an 8-bit binary number. So, this section focuses on examples that use 8-bit-long binary numbers.

Converting from binary to decimal is actually relatively straightforward, at least compared to converting from decimal to binary. In fact, you’ve actually already seen the math in the text following Table 1-6. To convert a binary number to decimal, you just have to think about the binary number in a table, such as Table 1-7, and apply what the table’s numbers mean.

Table 1-7 An Example of Binary-to-Decimal Conversion: 10101101

Powers of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Value Associated with That Digit or Column	128	64	32	16	8	4	2	1
Number Itself	1	0	1	0	1	1	0	1

Table 1-7 looks exactly like Table 1-6, except the binary number itself is slightly different to show another example. To convert to decimal, you simply multiply each pair of numbers that are in the same column of the last two rows in the table and add the numbers from the results of each product. Table 1-8 repeats the same information shown in Table 1-7, but now the conversion process math is shown.

Table 1-8 Converting 10101101 to Decimal: Multiplying Each Column and Then Adding Them Together

Value Associated with That Digit or Column	128	64	32	16	8	4	2	1
Number Itself	1	0	1	0	1	1	0	1
Product of Two Numbers in Same Column	128	0	32	0	8	4	0	1
Sum of All Products	173							

The process is indeed simple as long as you remember all the powers of 2! When working with IP addressing, you need to memorize all the powers of 2 up through 2^8 , or 256, as shown in the previous tables. The basic algorithm to convert binary to decimal can be summarized as follows:



- Step 1** Write the powers of 2, in decimal, in the top row of a table, similar to Table 1-8.
- Step 2** On the second line, write the binary number that is to be converted, lining up each binary digit under the powers of 2.
- Step 3** Multiply each pair of numbers (the numbers in the same column).
- Step 4** Add the eight products from Step 3.



Lab 1.2.6 Binary-to-Decimal Conversion

In this lab, you learn and practice the process of converting binary values to decimal values.

Tip

The online curriculum has a tool that gives you sample 8-bit binary numbers, asks you to provide the decimal equivalent, and then checks your answer.

Converting from Decimal Numbers to 8-Bit Binary Numbers

Converting from decimal to binary requires more effort and work compared with converting from binary to decimal. The generic process to perform the conversion has several steps. For this book’s purposes, the conversion process is slightly simplified by assuming that the goal is to convert a decimal number to an 8-bit binary number. To ensure that 8 bits are enough, the only decimal values that can be converted are 0 through 255 (inclusive). Note that the only decimal values allowed as octets of an IP address are also 0 through 255.

Generic Decimal-to-8-Bit-Binary Conversion Process

This book shows a slightly different process than the online curriculum does for converting from decimal to binary. If you are happy and accustomed to the version described in the online course, you might want to skip to the section “Converting IP Addresses Between Decimal and Binary.”

Begin by writing down eight powers of 2, starting with 128 on the left ending with 1 on the right (similar to Table 1-9). The blank lines below the powers of 2 are placeholders in which the binary digits will be recorded.

Table 1-9 Convenient Table for Converting Decimal to 8-Bit Binary

Power of 2	128	64	32	16	8	4	2	1
Binary Digits	—	—	—	—	—	—	—	—

Beginning with 128 and moving toward 1, repeat the following step eight times, once for each power of 2:



Step 1 If the decimal number is greater than or equal to the power of 2, do the following:

- a Record a 1 as the binary digit underneath the power of 2.
- b Subtract the power of 2 from the decimal number, which results in a number called the “remainder.”
- c Use the remainder for the next step/power of 2.

Step 2 If the decimal number is less than the power of 2, do the following:

- a Record a 0 as the binary digit underneath the power of 2.
- b Move to the next power of 2 and use the same remainder as in this step.

Example 1 of the Generic Conversion Process: Decimal 235

The first example shows how to convert decimal 235 to 8-bit binary. To begin, record the eight powers of 2, as shown in Table 1-9. Then, start with 128, and determine whether to use Step 1 or Step 2 of the algorithm based on where the decimal number is greater than or equal to 128.

In this case, it is clear that 235 => 128, so Step 1 in the algorithm is used. Record a binary 1 for the binary digit under 128. Then, calculate 235 – 128 = 107, and use this remainder in the next step. Table 1-10 shows the work in progress at this point.

Table 1-10 Results: Converting 235 After 128’s Digit Is Found

Power of 2	128	64	32	16	8	4	2	1
Binary Digits	<u>1</u>	—	—	—	—	—	—	—

For the next step, the 64’s digit is determined by using a remainder of 107. 107 => 64, so Step 1 is performed. Record a binary 1 for the binary digit under 64. Then, calculate 107 – 64 = 43, and use this remainder for the next digit to the right. Table 1-11 shows the results.

Table 1-11 Results: Converting 235 After 64’s Digit Is Found

Power of 2	128	64	32	16	8	4	2	1
Binary Digits	_1_	<u>_1_</u>	___	___	___	___	___	___

For the next step, the 32’s digit is determined by using a remainder of 43. $43 \Rightarrow 32$, so Step 1 is performed. Record a binary 1 for the binary digit under 32. Then, calculate $43 - 32 = 11$, and use this remainder for the next digit to the right. Table 1-12 shows the results.

Table 1-12 Results: Converting 235 After 32’s Digit Is Found

Power of 2	128	64	32	16	8	4	2	1
Binary Digits	_1_	_1_	<u>_1_</u>	___	___	___	___	___

For the next step, the 16’s digit is determined by using a remainder of 11. $11 < 16$, so Step 2 is performed. Record a binary 0 for the binary digit under 16, and use the same remainder (11) for the next digit to the right. Table 1-13 shows the results.

Table 1-13 Results: Converting 235 After 16’s Digit Is Found

Power of 2	128	64	32	16	8	4	2	1
Binary Digits	_1_	_1_	_1_	<u>_0_</u>	___	___	___	___

For the next step, the 8’s digit is determined by again using a remainder of 11. $11 \Rightarrow 8$, so Step 1 is performed. Record a binary 1 for the binary digit under 8. Then, calculate $11 - 8 = 3$, and use this remainder for the next digit to the right. Table 1-14 shows the results.

Table 1-14 Results: Converting 235 After 8’s Digit Is Found

Power of 2	128	64	32	16	8	4	2	1
Binary Digits	_1_	_1_	_1_	_0_	<u>_1_</u>	___	___	___

For the next step, the 4’s digit is determined by using a remainder of 3. $3 < 4$, so Step 2 is performed. Record a binary 0 for the binary digit under 4, and use the same remainder (3) for the next digit to the right. Table 1-15 shows the results.

Table 1-15 Results: Converting 235 After 4’s Digit Is Found

Power of 2	128	64	32	16	8	4	2	1
Binary Digits	_1_	_1_	_1_	_0_	_1_	<u>_0_</u>	___	___

For the next step, the 2’s digit is determined by again using a remainder of 3. $3 \Rightarrow 2$, so Step 1 is performed. Record a binary 1 for the binary digit under 2. Then, calculate $3 - 2 = 1$, and use

this remainder for the next digit to the right. Table 1-16 shows the results.

Table 1-16 Results: Converting 235 After 2’s Digit Is Found

Power of 2	128	64	32	16	8	4	2	1
Binary Digits	_1_	_1_	_1_	_0_	_1_	_1_	_0_	__

For the eighth and final step, the 1’s digit is determined by using a remainder of 1. $1 \Rightarrow 2$, so Step 1 is performed. Record a binary 1 for the binary digit under 1. No subtraction is needed here, but the remainder is always 0 at this point. Table 1-17 shows the final results.

Table 1-17 Results: Converting 235 After 1’s Digit Is Found

Power of 2	128	64	32	16	8	4	2	1
Binary Digits	_1_	_1_	_1_	_0_	_1_	_1_	_0_	_1_

Alternative Decimal-to-Binary Conversion Process

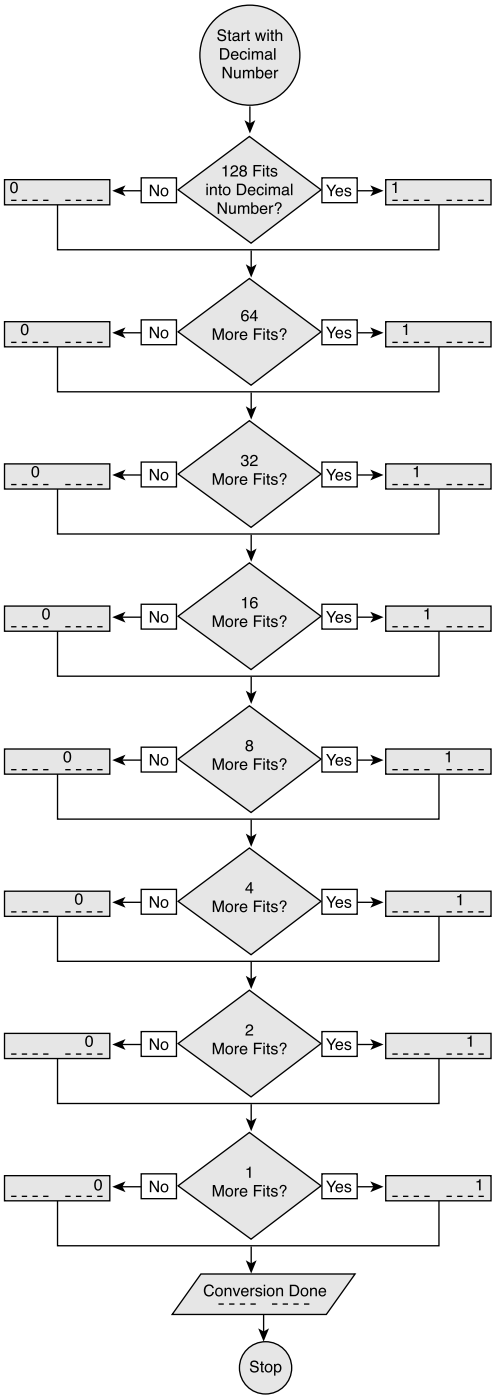
The Networking Academy CCNA 1 curriculum describes another process for converting decimal to binary. Both are valid, as well as other methods your instructor might teach in class. The goal of all these tools is to help you learn how to convert decimal to binary; feel free to use any valid method that makes sense to you and results in the correct answer. The lab referenced here provides some extra decimal-to-binary conversion practice, using the process from the course in the explanation. For reference, Figure 1-31 repeats the flowchart for decimal-to-binary conversion, which is taken from the online curriculum.



Lab 1.2.5 Decimal-to-Binary Conversion

In this lab, you practice converting decimal values to binary values.

Figure 1-31 Decimal-to-Binary Conversion Steps Used in the Online Curriculum



Example 2 of the Generic Conversion Process: Decimal 192

As you can see, the process is not particularly difficult at any one step, but it is laborious. Before leaving the process of converting decimal to its binary equivalent, however, this chapter includes one more example.

This example describes the steps taken to convert decimal 192 to its binary equivalent. Table 1-18 shows the binary equivalent, with each digit under the respective power of 2 that it represents (for easier reference).

Table 1-18 Results: Decimal 192 with Binary Equivalent

Power of 2	128	64	32	16	8	4	2	1
Binary Digits	_1_	_1_	_0_	_0_	_0_	_0_	_0_	_0_

Tip

The other popular decimal values when working with IP addressing are 0, 128, 192, 224, 240, 248, 252, 254, and 255. These numbers might be good values with which to practice the conversion process.

Tip

The online curriculum has a tool that gives you sample decimal numbers, asks you to provide the 8-bit binary equivalent, and then checks your answer.

The following list explains what happens at each step of the process, starting with the 128's digit:

- Step 1** 192 => 128, so the 128's digit is 1. $192 - 128 = 64$, with 64 then being used for the comparisons for the 64's digit.
- Step 2** 64 => 64, so the 64's digit is 1. $64 - 64 = 0$, with 0 then being used for the comparisons for the 32's digit.
- Step 3** For the last six steps, the remainder (0) is always less than the power of 2. Therefore, record 0s for the remaining six digits.

This example shows how you can shorten the process once the remainder is 0. Essentially, once the remainder is 0, the rest of the binary digits are also 0.

Converting IP Addresses Between Decimal and Binary

IP addresses are 32-bit binary numbers, but because humans find it inconvenient to write 32-bit numbers, the addresses are written in dotted-decimal format. In dotted-decimal format, each octet has a decimal number between 0 and 255 (inclusive). Each decimal number represents 8 binary digits. So, to convert an IP address from decimal to binary (or vice versa), you must break down the problem into four different conversions between a decimal number and an 8-bit binary number.

By registering your book at the following website (www.ciscopress.com/title/1587131641) and navigating to the Extra Practice section, you can access multiple practice problems for all the conversion processes covered in this chapter. This Extra Practice section includes problems for conversions between binary, decimal, and hexadecimal, as well as conversions between binary and decimal IP addresses.

This section first describes how to convert dotted-decimal IP addresses to their binary equivalents, and then it describes the opposite.

Converting Decimal IP Addresses to Binary IP Addresses

You already read the math behind the conversion process between decimal and binary. To convert IP addresses, you simply need to follow a few additional rules:

- How To

Step 1 When converting a decimal IP address to binary, convert each of the four decimal numbers in the decimal IP address to an 8-bit number, which results in a total of 32 bits.

Step 2 Include leading 0s in the binary values; otherwise, the IP address will not have 32 bits.

Step 3 Form the 32-bit binary IP address by simply writing each of the four sets of 8 bits in order.

Note

This process assumes that the decimal conversion yields an 8-bit binary number, including any leading 0s.

The three-step process to convert a decimal IP address to its 32-bit binary equivalent simply requires that you convert each of the four decimal numbers, keeping any leading 0s, and combine the results into one long 32-bit binary number. For example, Table 1-19 shows a sample conversion of the IP address 100.235.2.2.

Table 1-19 Conversion of Decimal IP Address 100.235.2.2 to Binary

	1st Octet	2nd Octet	3rd Octet	4th Octet
Decimal Octet	100	235	2	2
Each Octet Converted to Binary (Step 1)	01100100	11101011	00000010	00000010
Resulting 32-Bit Number (Step 2)	01100100111010110000001000000010			

Table 1-19 begins with the decimal IP address in the first row and the results of each conversion step in the next two rows. The actual math for converting the decimal numbers (Step 1) is not shown, but you can refer to the previous section for examples using decimal 100 and 235. Step 2 just lists all 32 bits in succession. In real life, there is no need to actually write Step 3; you can just see the four sets of 8 bits in a row and think of it as a 32-bit number.

Converting Binary IP Addresses to Decimal IP Addresses

To convert a binary IP address to its decimal equivalent, you already know the 32-bit IP address. The process is relatively simple compared to converting decimal to binary:

- How To

Step 1 Separate the 32 bits into four groups of 8 bits (4 octets).

Step 2 Convert each binary octet to decimal.

Step 3 Insert a period between the four decimal numbers.

The algorithm can be shown with a sample binary value:
01100100111010110000000100000001. Table 1-20 organizes the bits into octets with 8 bits each.

Table 1-20 Conversion of Binary IP Address to Decimal

	1st Octet	2nd Octet	3rd Octet	4th Octet
Binary Value, Separated into Four Octets (Step 1)	01100100	11101011	00000001	00000 001
Each Octet Converted to Decimal (Step 2)	100	235	1	1
Decimal IP Address in Dotted-Decimal Format (Step 3)	100.235.1.1			

For some reason, many people trip up when completing the first step of this process. Whenever you convert a binary IP address to decimal, the conversion process must use four sets of 8 bits. Chapter 10 describes IP subnetting, for which you might work with parts of IP addresses that are not 8 bits in length. However, to convert binary and decimal IP addresses, you must always work with 8 bits at a time.

Using a Conversion Chart

You can always use a calculator to do the math of converting a decimal number to binary and vice versa. However, because IP addresses use only decimal numbers between 0 and 255, you can also use a binary/decimal conversion chart. A binary/decimal conversion chart simply lists decimal numbers and their binary equivalents. That way, you can look at the chart and find the numbers without doing all the math previously covered in this chapter. Appendix B, “Binary/Decimal Conversion Chart,” contains a binary/decimal conversion chart.

For example, to convert 100.235.1.1 to binary, you can look in the chart and find the decimal number 100. Beside the number 100 is the 8-bit binary number 01100100. You can simply copy down those binary digits as the first 8 binary digits. Next, you find 235 in the chart, find the binary value beside it (namely, 11101011), write that down, and move to the next octet.

You can also use the chart to convert binary IP addresses to decimal by reversing this process.

Caution

The assessments for this course do not allow the use of calculators, nor do they provide sample tables like the ones shown in this chapter. You must practice the processes, particularly for converting IP addresses to and from decimal and binary, to prepare for these tests.

Hexadecimal Numbering System: Base 16

Hexadecimal numbering (Base 16), popularly called “hex,” is another number system that is used frequently when working with computers because it can represent binary numbers in a more readable form. The computer performs computations in binary, but there are several instances in which a computer’s binary output is expressed in hexadecimal form to make it easier to read. Each hex digit represents 4 bits, so the output is much smaller, which makes reading it much easier on the eyes.

The hexadecimal number system uses 16 symbols. Hex uses the same 10 numerals as decimal (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9), plus six more. The additional symbols are the letters A, B, C, D, E, and F. The A represents the decimal number 10, B represents 11, C represents 12, D represents 13, E represents 14, and F represents 15, as shown in Table 1-21.

Table 1-21 Converting Between Hexadecimal Digits, Binary, and Decimal

Hexadecimal Digit	Binary Equivalent	Decimal Equivalent
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

In some parts of the computing world, converting between hexadecimal and decimal can be important. For networking, the most common conversion using hex is the conversion between

hexadecimal and binary and vice versa. The conversion can be easily accomplished by simply using the information shown in Table 1-21.

Although hex-to-binary conversion is not required often in networking, occasionally, the need does arise when working with a Cisco router feature called the configuration register. Although the definition of what the configuration register does is not covered until the CCNA 2 course, the value, which is a four-digit hex number, can be manipulated. For example, most routers' configuration registers are set to hex 2102. However, individual bit values in the register have different meanings, so it is common to need to convert it to binary, as shown here:

0010000100000010



Lab 1.2.8 Hexadecimal Conversions

This lab requires that you practice the process of converting hexadecimal numbers into binary and decimal numbers. Note that hex-to-decimal conversion is not covered in the online course, or this chapter, but the lab does cover the process.

Boolean or Binary Logic

Boolean math is a branch of mathematics created by George Boole. Boolean math creates a way to use math to analyze a large set of problems, including logic, electrical circuits, and certainly computing. Boolean math typically involves applying Boolean functions to 1 or 2 bits.

Two Boolean math operations are popular when performing IP subnetting calculations—namely, the Boolean AND and OR operations. Both functions take two different bits as input, and they provide a result of a single bit. Table 1-22 summarizes the AND and OR operations.

Table 1-22 Boolean AND and OR

First Bit	Second Bit	Results of an AND of These 2 Bits	Results of an OR of These 2 Bits
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Although Table 1-22 shows the formal results, the logic of AND and OR can be simply phrased:

- A Boolean AND yields a 1 only if both bits are 1.
- A Boolean OR yields a 1 if at least one of the 2 bits is a 1.

In addition to the AND and OR, another Boolean function mentioned in the online course material is the NOT operation. This function takes a single bit as input and yields a result of the other bit. In other words, taking the NOT of 0 yields a 1, and the NOT of 1 yields a 0.

IP Subnet Masks

Chapter 10 shows how Boolean logic, particularly Boolean AND, can analyze and work with IP addresses. Some number of the first bits of the 32-bit IP addresses represents the group in which the IP address resides. These groups are called IP networks (or subnets). The number of these initial bits that represents the group (network or subnet) varies based on choices made by the people implementing the network. By considering IP address design concepts—which are covered in Chapters 9 and 10 and in other CCNA courses—a network engineer chooses the number of initial bits to use to identify a subnet.

After the engineer completes his analysis, he must tell the computers and networking devices how many initial bits have been chosen. To do so, the engineer uses a second 32-bit number called a *subnet mask*, which is a guide that determines how the IP address is interpreted. It indicates how many bits in the first part of the address identify the subnet. To do so, the mask lists several consecutive binary 1s and then all binary 0s. The number of initial binary 1s defines the number of initial bits that identify the subnet.

For example, earlier in this chapter, Figure 1-21 showed a sample internetwork, and Figure 1-22 showed the concept of subnets used in that internetwork. In Figure 1-22, a subnet mask of 255.255.255.0 was used. This mask, in binary, is as follows:

```
11111111 11111111 11111111 00000000
```

With 24 initial binary 1s, this mask means that the first 24 bits of an IP address must be the same for all hosts in the same subnet. (Refer to Figure 1-22 to see this same logic in simple text.)

255.255.255.0 is just one example of a subnet mask. Many possible masks exist, but they must all begin with a number of binary 1s and then end in all binary 0s—the 1s and 0s cannot be mixed. Just for perspective, the following lines show two other popular and simple subnet masks:

```
255.0.0.0      11111111 00000000 00000000 00000000
```

```
255.255.0.0    11111111 11111111 00000000 00000000
```

The first of these masks means that all IP addresses must have their first 8 bits in common (first octet) to be in the same subnet. The second line means that all IP addresses in the same subnet must have their first 16 bits in common (first and second octets) to be in the same subnet.

Using Boolean Math with Subnets

IP uses a number called a subnet number to represent all IP addresses in the same subnet. A Boolean AND can find the subnet number. Although Chapter 10 covers this concept in more depth, Module 1 of the online curriculum introduces the concept, so it is introduced briefly in this chapter.

The following process allows you to find a subnet number, given an IP address and the subnet mask used with the IP address:



- Step 1** Convert the IP address and mask to binary. Write the IP address first and the subnet mask directly below it.
- Step 2** Perform a bitwise Boolean AND of the two numbers.
- Step 3** Convert the resulting 32-bit number, 8 bits at a time, back to decimal.

A bitwise Boolean AND means to take two equal-length binary numbers, AND the first bit of each number, AND the second bit of each number, then the third, and so on until all bits are ANDed together. Table 1-23 shows an example based on Figure 1-21 (IP address 172.16.2.21 and mask 255.255.255.0).

Table 1-23 Using a Bitwise Boolean AND to Find the Subnet Number

	1st Octet	2nd Octet	3rd Octet	4th Octet
IP Address 172.16.2.21 (Step 1)	10101100	00010000	00000010	00010101
Mask 255.255.255.0 (Step 1)	11111111	11111111	11111111	00000000
AND Result (Subnet Number [Step 2])	10101100	00010000	00000010	00000000
Decimal Subnet Number	172.16.2.0			

Summary

The online course points out that a connection to a computer network can be broken down into the physical connection, the logical connection, and the applications that interpret the data and display the information. This book further describes a network as computer hardware (for example, NICs and modems), software (for example, TCP/IP and web browsers), networking devices (for example, routers and switches), and network cabling.

TCP/IP software configuration includes an IP address, a subnet mask, and a default gateway. To test a connection, tools such as **ping** can verify whether packets can be delivered across a network. The **tracert** command can also help isolate routing problems.

Access to the global TCP/IP, known as the Internet, requires some form of Internet access. This access can be gained using a modem, DSL, or cable modem. After connecting, a user can use applications, such as web browsers, which might use plug-ins, to view content held on servers in the Internet.

Computers recognize and process data using the binary (Base 2) numbering system. Often, the binary output of a computer is expressed in hexadecimal to make it easier to read. The ability to convert decimal numbers to binary numbers is valuable when converting dotted-decimal IP addresses to machine-readable binary format. Conversion of hexadecimal numbers to binary, and binary numbers to hexadecimal, is a common task when dealing with the configuration register in Cisco routers. The 32-bit binary addresses used on the Internet are referred to as IP addresses.

Boolean logic (a binary logic) allows two numbers to be compared and a choice generated based on the two numbers. Subnetting and wildcard masking use Boolean logic.

Check Your Understanding

Complete all the review questions listed here to test your understanding of the topics and concepts in this chapter. Answers are listed in Appendix A, “Answers to Check Your Understanding and Challenge Questions.”

1. What is the function of a modem?
 - A. Replace a LAN hub
 - B. Allow two computers to communicate by connecting to the same modem
 - C. Modulate a signal it sends and demodulate a signal it receives
 - D. Demodulate a signal it sends and modulate a signal it receives
2. What is the main circuit board of a computer?
 - A. PC subsystem
 - B. Motherboard
 - C. Backplane
 - D. Computer memory
3. Select three popular web browsers.
 - A. Mozilla Firefox
 - B. Adobe Acrobat
 - C. Internet Explorer
 - D. Netscape
 - E. Windows Media Player
4. What is a NIC?
 - A. A WAN adapter
 - B. A printed circuit board or adapter that provides LAN communication
 - C. A card used only for Ethernet networks
 - D. A standardized data link layer address
5. Which of the following is/are the resource(s) you need before you install a NIC?
 - A. Knowledge of how the NIC is configured
 - B. Knowledge of how to use the NIC diagnostics
 - C. Ability to resolve hardware resource conflicts
 - D. All answers provided are correct

6. Which number system is based on powers of 2?
- A. Octal
 - B. Hexadecimal
 - C. Binary
 - D. ASCII
7. The terms and definitions in the following table are scrambled. Match the following terms with their definitions.

Term	Definition
Bit	Standard measurement of the rate at which data is transferred over a network connection
Byte	Approximately 8 million bits
kbps	Smallest unit of data in a computer
MB	Unit of measurement that describes the size of a data file, the amount of space on a disk or another storage medium, or the amount of data being transferred over a network

8. What is the largest decimal value that can be stored in 1 byte?
- A. 254
 - B. 256
 - C. 255
 - D. 257
9. What is the decimal number 151 in binary?
- A. 10100111
 - B. 10010111
 - C. 10101011
 - D. 10010011
10. What is the binary number 11011010 in decimal?
- A. 186
 - B. 202
 - C. 218
 - D. 222

11. What is the binary number 0010000100000000 in hexadecimal?
- A. 0x2100
 - B. 0x2142
 - C. 0x0082
 - D. 0x0012
12. What is the hexadecimal number 0x2101 in binary?
- A. 0010 0001 0000 0001
 - B. 0001 0000 0001 0010
 - C. 0100 1000 0000 1000
 - D. 1000 0000 1000 0100
13. Which of the following statements are true of **ping**? (Select the two best answers.)
- A. The **ping** command tests a device's network connectivity.
 - B. **Ping** discovers the IP address of every router between two computers.
 - C. The **ping 127.0.0.1** command verifies the operation of the TCP/IP stack.
 - D. All of the answers are correct.

Challenge Questions and Activities

These activities require a deeper application of the concepts covered in this chapter, similar to how answering CCNA certification exam questions requires applying detailed concepts to a particular scenario.

The following two activities are difficult for this point in the class and, in some cases, have not yet been covered in the text. They are indeed meant to give you a challenging set of problems, ones that most readers are not yet able to fully answer. For those of you looking for an extra challenge, try the following exercises. By the end of the CCNA 1 course, if given enough time, you should be able to easily solve such problems.



Activity 1-1: Using Packet Tracer, load the *enterprise-working* configuration, which can be downloaded at www.ciscopress.com/title/1578131641. The configuration matches Figure 1-13. Characterize the subnets used in the design. For example, a LAN's subnet might be "All IP addresses that begin with 10."



Activity 1-2: Using Packet Tracer, load the *enterprise-broken-1* configuration, which can be downloaded at www.ciscopress.com/title/1578131641. The configuration matches Figure 1-13, except that some things were misconfigured on purpose. Test which PCs can **ping** other PCs. When a **ping** fails, work to discover the problem. Note that some problems might be caused by a problem that has not yet been fully explained at this point in this book; however, if you cannot solve them all, write what you can about the problems.

Note

The network topologies shown in the Packet Tracer examples are not meant to show the same network topologies that appear in the online curriculum or the labs; instead, they purposefully use different topologies to show alternative examples.