CONNECTING COMPUTERS

CHAPTER 10

ABOUT THIS CHAPTER

This chapter introduces computer networking and some basic issues in network security. We look at the following:

- Digital networking and reliable transmission
- Building and operating a small-scale LAN
- Reliability and error detection
- Network protocols and the protocol stack
- Network applications and resource sharing

10.1 The Network Security Problem

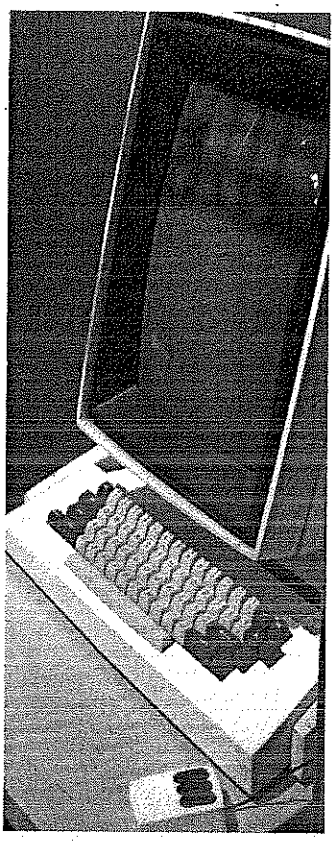
In this chapter, we graduate from isolated computers to the problems of networked computers. This brings us closer to the entire range of modern information security risks and thus increases the complexity of the problem. For now, we narrow our focus to ***local area networks*** (LANs).

The modern local network emerged at the same time and place as the modern personal computer: in the 1970s at Xerox Corporation's Palo Alto Research Center, commonly called PARC. Researchers at PARC produced the Alto, the prototype of today's personal computer (Figure 10.1). PARC installed a prototype ***Ethernet*** to connect the Altos together. Alto users shared printers, files, and other resources across this early LAN.

Local networks became an essential desktop feature in the 1980s. Office desktops rarely had Internet connections until the late 1990s, but all had LANs to share files and printers. In most cases, these LANs used improved Ethernet products.

When we talk about computer networks, even the simplest network has two parts: the hosts and the links. A ***host*** is a computer connected to the network. A ***link*** is a component that connects the host to other hosts or to the networks. The simplest network consists of two hosts with a single wire linking them. More complex networks may include separate devices that connect multiple hosts together or that connect networks together.

Even today, wired network connections remain popular. As wireless connections improve in speed and reliability, so do wired connections. It seems likely that a wired LAN always will run faster and more reliably than wireless. It is easier to control and



Photographed by Dr. Richard Smith at the Computer History Museum, California

Figure 10.1
The Xerox Alto.

protect a signal on a wire. Wireless LAN signals, on the other hand, must compete with other sources of electronic signals and noise, including motors, microwave ovens, audio/video equipment, and so on. Not only do physical wires isolate a LAN from random noise and interference, they also protect it from outsiders. When we wire up a LAN, it is usually inside a building. We can lock the LAN away from outsiders. We can keep track of which computers we attach to the LAN and which we omit.

As networks grow larger, however, it becomes harder to keep track of its computers. Most wired networks use standard plugs and jacks. If someone has an unused jack that's connected to the network, it's easy to connect another computer. Some organizations try to keep a strict accounting of every authorized computer, but many don't. A malicious visitor can easily plug in to an unused jack; now the network is no longer isolated from attacks.

The rest of this section reviews the following:

- Possible attacks on the network
- Defending the network via physical protection
- Defending hosts against attack

The rest of the chapter gives a tutorial on networking fundamentals. These technical details will give us a better understanding of the risks and how our defenses work.

10.1.1 Basic Network Attacks and Defenses

Figure 1.6 in Chapter 1 lists six general attack types, all of which apply to networks. However, we place a different emphasis on these threats, so we examine them in a different order:

- **Physical theft.** Someone steals network hardware, like wires, hubs, or other equipment that keeps the network running.
- **Subversion.** Someone modifies or otherwise takes over part of the network so that it enables an attack. For example, an attacker might reroute traffic to allow its interception. Note that in networking, this threat involves *physical* or *logical* changes to network components. It does not involve changes to network traffic.
- **Disclosure.** An attacker's computer intercepts copies of network data intended for others. While this may pose no risk for a lot of network traffic, this type of eavesdropping may yield passwords or other data that enables a more serious attack.
- **Forgery.** Someone constructs a bogus message or modifies a legitimate message as part of an attack. For example, a bogus order could send merchandise without collecting payment.

- **Masquerade.** A person tricks the network into sending messages claiming to be originated by someone else. In the networking environment, this behaves like a particular type of forgery.
- **Denial of service.** An attack that makes some or all of the network unusable. Typical attacks either flood parts of the network with traffic or render network components unusable.

Example: Sharing a Network

Let us consider the attacks and their risks in terms of a specific problem:

> Bob's office is in the Old Masonic building. Many years ago, the tenants formed a cooperative to install and maintain LAN wiring and a shared printer inside the building. Everyone in the building is a member of the cooperative. The wiring doesn't extend outside the building, and the cooperative's rules exclude outsiders from membership.
>
> The cooperative restricts access to tenants of the building. All tenants pay a monthly fee to cover supplies and utility costs.

Let's apply the risk management framework to the network cooperative. The assets are network equipment, printer supplies, and utility costs. The risk is that outsiders—those who aren't paying for the network or its supplies—will use up the printer supplies or slow down the network for the rest.

Policy Statements

There are three statements in the Old Masonic tenant cooperative's policy:

1. All tenants shall contribute to utility bills and printer supplies.
2. All tenants shall have access to the network and the printer.
3. Network and printer access shall only be granted to tenants.

Potential Controls

In Chapter 2, Figure 2.13 lists six general categories of security controls. These same categories provide basic network defenses.

- **Physical**—physically isolating some—or all—of the computers and network from potential threats.
- **Mechanical**—providing physical access via locked doors.
- **Logical**—restricting network behavior according to the origin or contents of a request: a user, a computer, or a network-oriented program. Network services use access rules to validate permissions before taking action.
- **Functional**—restricting network activity by not implementing risky actions.
- **Procedural**—protecting the network with operating procedures. For example, only trustworthy people get a key to the room with computers and the network.
- **Cryptographic**—using cryptography to protect network activities. We introduce this in Chapter 14.

What types of controls do the policy statements require? The first statement relies on procedural control. The Old Masonic cooperative has been part of the building for so long that the arrangement is part of each lease. The landlord collects the cooperative fee along with the rent and manages the utility bill.

10.1.2 Physical Network Protection

The remaining policy statements rely on physical controls to protect the cooperative's network. The entire network resides inside the Old Masonic building (Figure 10.2). Only building tenants have keys to the building; tenants on upper floors greet clients and customers at the front door and escort them. Computers must be physically connected to the network to use the printer or Internet connection.

Bob and Alice share a wall and a doorway. Alice shares a computer with Bob that is on the cooperative's network. But Alice's Arts is actually in the building next door. Alice can use the cooperative's printer and Internet connection because she shares Bob's computer, but she can't retrieve her printouts because she doesn't have a key to the printer closet in the Old Masonic building. Alice can't join the cooperative and have her own key to the printer closet because her store isn't in the Old Masonic building.

Alice would really like to connect her POS terminal to the Old Masonic network. It would be much less expensive than her current arrangement. The wiring job isn't especially difficult because there are already appropriate holes in their shared wall. Bob could install a low-cost network switch to share his connection with Alice's business.

Alice's hookup would clearly violate the existing policy. The cooperative could decide to include Alice, provide her with keys, and divide expenses with her. This reduces costs for everyone. They could rewrite the policy to say "Old Masonic building tenants and Alice."

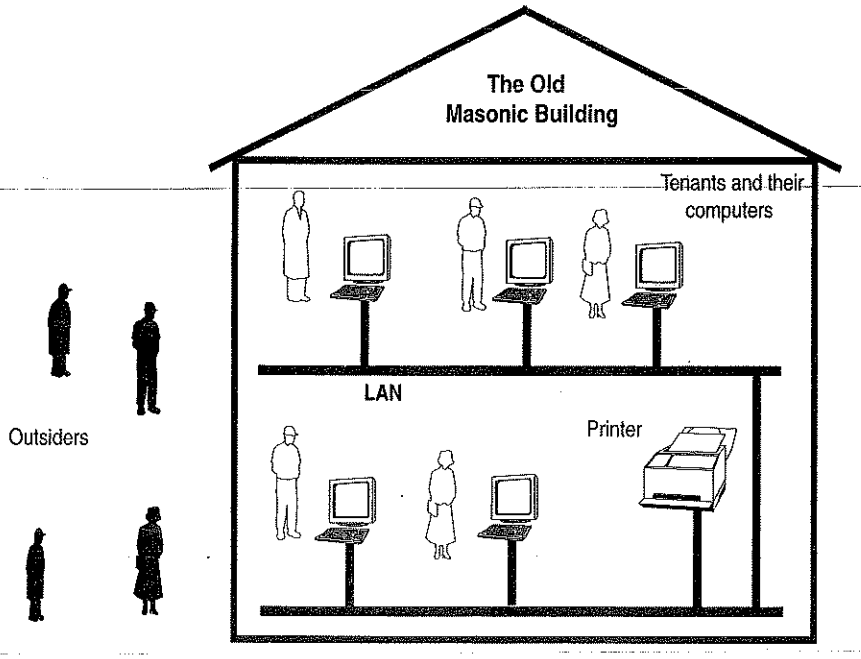


Figure 10.2

Physical protection of a LAN.

This extension poses another problem. Without Alice, the entire network is inside the Old Masonic building and under the physical control of the landlord. Once they run a wire into Alice's Arts, the landlord can't easily control what happens to that wire.

Protecting External Wires

Many companies implement a LAN for their employees' computers. If the company resides inside a single building, the LAN benefits from the building's physical protection. If the company has two adjacent buildings, they can run a network cable between the buildings. Now, however, they have to worry about attacks on the outside cable. Cables often run inside physically strong enclosures, like metal conduits.

An attacker must be highly motivated to break into a metal conduit. Although most attackers are not, sites may face this risk if their networks carry classified defense information. A network requires special physical protections if it carries information through areas open to people not allowed access to that information. We discuss this in greater detail in Section 17.5.

Not only do wires need protection, but the networking equipment needs protection, too. The suite's printer network relies on a few low-cost switches to hook up the computers. If Alice has physical access to a switch and passes a wire through Bob's wall, then she can hook into the network regardless of the cooperative's approval. However, she would still have some explaining to do when she tried to pick up a printout.

The tenants and landlord verify the integrity of their network by looking at its physical condition. They must account for every device wired in to it. An extra wire suggests an unauthorized user.

The Old Masonic cooperative is a close-knit group. They do not need risks or policy statements related to privacy, eavesdropping, or disclosure. However, some network users may use it for private matters, like printing a tax return or a love letter. This shouldn't pose a problem as long as the tenants may be trusted.

10.1.3 Host and Network Integrity

Integrity is another important part of modern network security. We must ensure that all host computers on the network are virus-free and can resist obvious attacks. Although this isn't as much of a problem on a small, isolated network like that in the suite, it remains a risk.

> The U.S. military protects its classified networks against attack by isolating them from other networks. If a worm infects millions of computers on the Internet, it won't infect the classified networks. There is no effective connection from the Internet to the classified networks, so there is no way for the infection to enter the classified networks.
>
> In December 2008, the classified networks were in fact infected, reportedly by the "agent.btz" worm. The infection was eventually traced to a USB flash drive that was plugged into a U.S. military laptop at a base in the Mideast. When plugged in, the laptop automatically ran a program on the drive that infected the computer and its connected network at the U.S. Central Command. The worm soon spread through both classified and unclassified networks.

Shortly after the attack, military commands issued orders forbidding the use of portable USB drives on military computers.

In Section 3.2.4, we reviewed different types of malware, some of which propagate using a USB drive. The same strategy penetrated sensitive U.S. military networks.

Network Worms

The Old Masonic building has had its LAN for a very long time. It was originally intended to share a printer and was not connected to the Internet. The original risk assessment and security policy didn't consider worms or other Internet-related risks. Now that the LAN connects to the Internet, it is exposed to a much larger—and riskier—user community. Worms now pose a serious risk to the tenants.

Whose fault is it if a worm infects a tenant's computer? Is it the cooperative's fault, since the network connects to the Internet? The policy says nothing about protecting tenants from outside threats. If the cooperative wanted some measure of protection, they could decide on a "firewalling" policy (see Section 12.4), but the most effective security measures usually involve up-to-date patching and proper management of individual computers. Worms thrive on errors in older, unpatched network software.

Like viruses, worms copy themselves to other places and spread from those other places. While viruses may infect USB drives, diskettes, and application programs, worms infect host computers. When a worm starts running, it takes these four steps:

1. Decide how to locate computers on the network.
2. Send messages to each computer to determine if it has particular security weaknesses. This may consist of simply trying various attacks; this is how the Morris worm operated.
3. If a computer is vulnerable to one of its attacks, the worm penetrates the computer.
4. After penetrating the computer, the worm installs a back door to give control to an attacker.

Note that worms search *automatically* for vulnerable computers. If a computer has not been patched to fix its vulnerabilities, a worm will probably find the computer while scanning the network. This could happen in a matter of minutes.

Early network worms often served no particular purpose. Many were intended to illustrate nothing more than a particular programmer's prowess at designing malware. In fact, early virus and worm authors often were caught because they bragged to others in black-hat hacker discussion groups. Today, however, worms serve a practical if illegal purpose: They help people construct botnets.

Botnets

Many of the malware packages discussed in Section 3.2.4 create and operate botnets for monetary gain. To create a botnet, the attackers must infiltrate host computers via a worm, virus, or Trojan. The malware then installs a ***rootkit***, software that remains hidden and that provides backdoor control of the computer. The infected host computer becomes

a *bot*, a working member of the botnet. Although the most dangerous rootkits have administrative control of a bot, some operate simply as a user. In either case, the rootkit gives the botnet controller the software to control the bot.

A host often becomes a bot because its owner is careless about computer security. For example, the owner might not keep the software up to date, which leaves the computer open to worm attacks. A careless computer owner might also install an infected USB drive or click on an email-based attack.

A bot's owner usually doesn't realize that the computer is part of a botnet. Successful backdoor software is stealthy and has no obvious effect on the computer's behavior. If the software degrades the computer's behavior, the bot's owner may try to get the computer fixed. The diagnosis and repair process often uncovers the rootkit, allowing the backdoor software to be removed.

Botnets in Operation

A botnet controller may ask its bots to do any of several things. Most bots can download additional malware. This allows the botnet controller to modify the bot to resist attacks on it and to add capabilities. Typical botnets may perform one or more of these operations:

- Steal authentication credentials from the bot's owner. The ZeuS botnet focuses primarily on this theft.
- Harass the bot's owner with pop-up windows. Sometimes the pop-up is simply advertising. In some cases, the pop-up claims to be a virus warning. The bot's owner then is offered antivirus software that's actually additional malware. Some botnets actually charge the bot's owner for this malware download.
- Send spam (see Section 15.3.1).
- Perform ***distributed denial of service*** (DDOS) attacks. Such attacks transmit a modest amount of data from each bot to the target system. If the botnet has thousands or tens of thousands of bots, the target system may be overwhelmed by the traffic load.

Botnets are often in competition with one another. No botnet wants to share a victim host with another botnet. If a host computer contains two or more bot packages, then their combined effect may slow the computer down and make the owner aware that there's trouble. If a botnet penetrates a host belonging to another botnet, the new one will try to remove the other network's bot software.

Fighting Botnets

Botnets routinely lose bots as individuals replace infected computers or install security software that detects and removes the bot software. However, this doesn't normally affect a large, healthy botnet. The operators continue to locate and incorporate new machines that replace the ones lost.

The most effective way of fighting botnets so far has been to go after a botnet's control mechanism. Botnet operators use a special set of hosts to send commands to the botnet. These command hosts are usually a botnet's weak point.

There have been numerous efforts to disable large botnets. Such "takedowns" often involve police organizations in several countries working in conjunction with software and network service providers. In 2014, a takedown disabled a large botnet based on "Gameover/ZeuS" software that was distributing Cryptolocker malware. In 2013, another takedown targeted the Citadel botnet community, which was estimated to have caused over a half a billion dollars in financial losses. An earlier, multiyear effort took down the Pushdo/Cutwail botnet, an earlier ZeuS-based botnet, and the Bredolab botnet. This effort decreased global spam levels at least 40 percent between August and October 2010.

The Insider Threat

Do any of the Old Masonic tenants pose a network-based threat to other tenants? Eve, for example, is curious about everyone and everything. What if she sets up the printer to make her copies of everything it prints? This would be an insider threat. Eve is part of a community that trusts its members. The security environment assumes that people will follow the rules, both written and unwritten.

Many networks don't know they have an insider threat until something unfortunate happens. For example, Eve might be able to program the printer to send copies of everything to her own computer. If another tenant prints his tax forms, the printer sends a copy to Eve, too. If anyone finds out, the cooperative needs to reassess its risk environment.

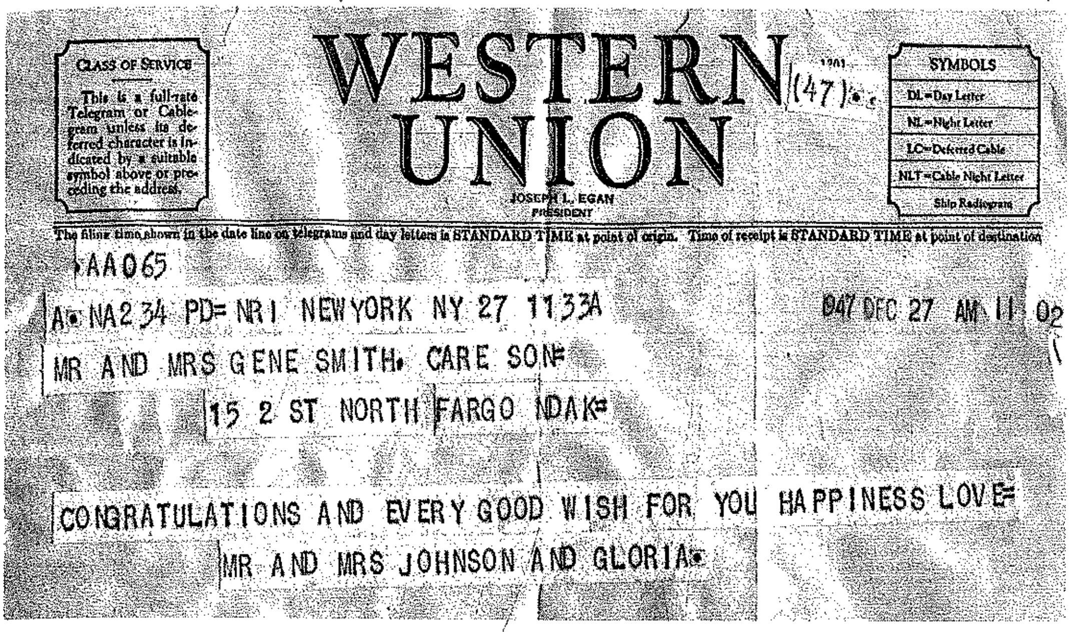
Eve might not be the only insider threat. If the cooperative says "No" to Alice's proposal, but Bob is willing to help her out, she could still join the network. Bob installs a switch and a wire to Alice's Arts, and other tenants might not notice. No one would be likely to notice Alice's connection unless someone "maps" the network (see Section 11.5.2). Alice's connection violates the cooperative's policy, which relies heavily on the tenants for self-policing and enforcement.

10.2 Transmitting Data

Humanity has carried written messages for long as it has had written words. ***Data networks*** eliminate the physical messages: they carry the text using a common alphabet or other set of symbols, often at the speed of light. Signal fires were perhaps the earliest such technique, though telegraphs were the first data networks to cover the globe.

The first true telegraph sent messages a letter at a time via sets of flags; the electrical telegraph coded letters as a series of clicks. The telegraph required skilled operators. To send or receive a message, the operator had to convert it between written text and telegraph code. Speed and reliability depended on operator skill, and good operators were hard to find.

This posed a particular challenge to stockbrokers and other investors. Money was made or lost by responding quickly to price fluctuations. For many years, brokers relied on runners to bring price information to their offices, and it wasn't practical for brokers to keep a staff of telegraph operators.


WESTERN UNION

CLASS OF SERVICE

This is a full-rate Telegram or Cablegram unless its deferred character is indicated by a suitable symbol above or preceding the address.

1201 (47)

SYMBOLS
DL=Day Letter
NL=Night Letter
LC=Deferred Cable
NLT=Cable Night Letter
Ship Radiogram

JOSEPH L. EGAN
PRESIDENT

The filing time shown in the date line on telegrams and day letters is STANDARD TIME at point of origin. Time of receipt is STANDARD TIME at point of destination

AA065

A NA234 PD=NR1 NEWYORK NY 27 1133A 1947 DEC 27 AM 11 02

MR AND MRS GENE SMITH CARE SON=

15 2 ST NORTH FARGO NDAK=

CONGRATULATIONS AND EVERY GOOD WISH FOR YOU HAPPINESS LOVE=

MR AND MRS JOHNSON AND GLORIA

Courtesy of Dr. Richard Smith

Figure 10.3

A mid-20th century telegram from New York to Fargo.

In 1867, Edward Callahan produced a device to convert telegraph signals directly into printed text. This became the first "stock ticker," which printed stock transactions onto a stream of "ticker tape." In 1874, Frenchman Émile Baudot patented his "printing telegraph" that used a keyboard to transmit telegraph messages. A series of improvements led to electromechanical teletypes that fully automated telegraph transmission. By World War I, teletypes had replaced telegraph keys in major telegraph offices. The receiving office printed out the message and pasted it to a telegram form to deliver to the recipient (Figure 10.3).

While the ticker tape system broadcast the same messages to numerous recipients, most telegraph messages traveled between one sender and recipient. For example, Bob's great uncle in New York might have needed to send a telegram to his cousin in Boston. Because the cities are a few hundred miles apart and do a lot of business with each other, there was a telegraph line (a network link) directly connecting the two cities.

To send the telegram, Bob's great uncle wrote out the message and handed it to a telegraph clerk. The clerk collected payment, found a teletype, and connected it to the Boston link (Figure 10.4). The clerk typed the message in the New York office, which simultaneously typed out on a teletype in the Boston office. A clerk at that office dispatched a courier to take the telegram to the cousin's address.

If Bob's great uncle needed to telegraph wedding congratulations to his niece in Fargo, North Dakota, the process was more difficult. The great uncle still wrote out the message, handed it to the clerk, and paid for it.

However, the price was much higher because the message traveled farther and needed additional handling to reach its destination. At the time, Fargo was too far away and

Library of Congress, Prints & Photographs Division, FSA/OWI Collection, LC-USW3-032342-E

Figure 10.4

A clerk connects a teletype to a network link.

too obscure to have a direct line from New York. Instead, the telegram had to travel through a network of links that connected telegraph offices in major cities. Each office collected the message and retransmitted it down a link to take it closer to Fargo. As we can see on the telegram in Figure 10.3, it took almost a half hour to send a telegram halfway across the continent, crossing one time zone.

This example highlights a major difference between telegraph and telephone networks. A telephone conversation required a complete, end-to-end connection between both phones. The network had to establish the connection and keep it working for the duration of the call. Telegrams, on the other hand, were more like letters. The sender and recipient didn't have to be present at the instant of transmission. The message didn't have to traverse the network all at once; it could travel in stages.

When we combine this perspective of older networks with the capabilities of modern computer networks, we find three different strategies for sending data across a network.

1. *Message switching*—used by postal and telegraph networks.
2. *Circuit switching*—used by traditional telephone networks.
3. *Packet switching*—used by modern computer and cell phone networks.

We examine each of these here.

10.2.1 Message Switching

Postal systems and classical teletype networks use message switching. In the postal system, each letter or parcel moves as a single, indivisible unit. Each time a batch of mail is sorted, each letter is redirected to another office that brings it closer to its destination.

Moreover, the sender and recipient don't have to arrange a specific time to send and receive individual messages. The messages may arrive and wait for delivery. In the previous section, Bob's great uncle sent a telegram to his niece in distant Fargo. The niece did not abandon the wedding party to attend the telegram's arrival. The message reached the office at an unpredictable time and was delivered to the new bride by messenger.

Like letters and parcels, each telegram is a single, indivisible unit. The great uncle's telegram travels through the network as a single message. The message might traverse several offices on its trip across the country. Instead of retyping the message, the offices punched each incoming message on a stream of paper tape (Figure 10.5).

Then an operator placed the punched tape in a machine connected to the message's next destination, and the machine automatically retyped the message.

To send the great uncle's message, the clerk in the New York office first connects a teletype to a westbound link leading to Pittsburgh or some other city closer to Fargo. The clerk types in the message and it prints out in the Pittsburgh office. From there, another clerk relays the message to another office, perhaps in Chicago, moving the message even closer to Fargo.

Electronic mail systems use the same approach. When we send an email, the system handles and delivers the message as a single unit. We never receive part of an email; the system only handles and delivers complete messages.

Message switching has the following advantages:

- Independent receipt. Sender and recipient don't have to arrange a specific time to exchange a message.
- Completeness. The network delivers the entire message to the recipient or nothing at all. The network isn't designed to handle partial messages.

Message switching has these disadvantages:

- Size limits. Message size may be limited by equipment capacities. Longer messages are much harder to handle and are more likely to encounter errors.
- Longer delays. Each node must collect the entire message before it can forward it to another node in the network. This prevents the network from delivering different parts of a long message in parallel, which would reduce its delivery time.

10.2.2 Circuit Switching

Telephone systems traditionally use circuit switching. This dates back to the earliest telephones. When connecting a phone call, the network establishes a specific circuit to carry that individual call from the originating phone to the destination. This is because the network must carry the analog sound signals produced by the two speakers. The

Library of Congress, Prints & Photographs Division, FSA/OWI Collection, LC-USW3-032342-E

Figure 10.5

Teletype clerks check a message on paper tape.

sound must travel as fast as possible between the phones. Even a short delay may make it hard for the speakers to carry on a conversation.

To create a circuit for a phone call, the network assigns pieces of the network to that specific call (Figure 10.6). It assigns wires in each network link that connects the two phones. It also assigns circuitry inside each exchange.

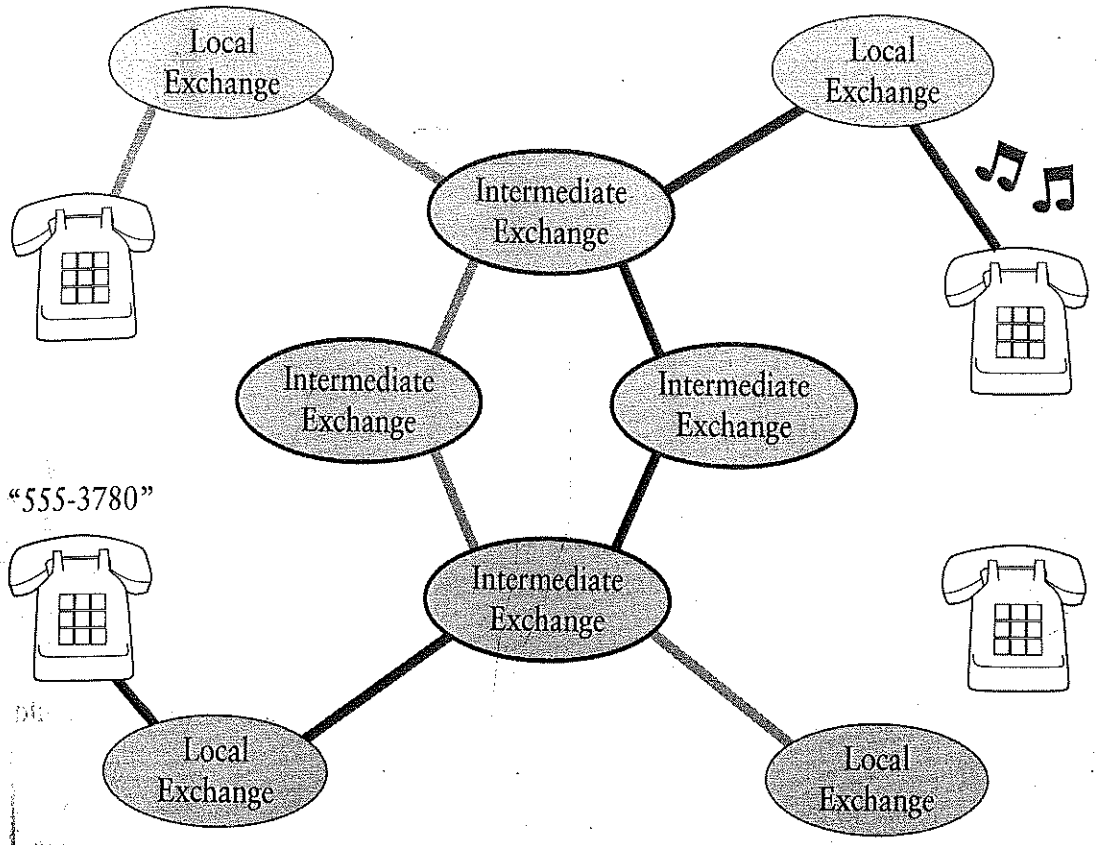


Figure 10.6
A phone call makes a circuit between two phones.

Whenever either person speaks, the electric signal passes along the link from the phone to the local exchange. From there, it travels across a circuit to take it to the network link leading to the next exchange that carries the call. This continues until it reaches the local exchange at the other end of the call. The signal then passes along the link that connects to the call's recipient. When either party hangs up the phone, the network reclaims the circuits and data links for use in another phone call.

The 20th century telephone system was so successful that their network engineers had trouble seeing the possible benefits of other networking strategies. Many such experts argued strongly against the use of packet-switching technology when it was new. Their connection-oriented network reliably supported a broad range of services, so their argument seemed plausible at the time.

Circuit switching provided these advantages:

- Rapid connections. The network could quickly establish a connection between any two endpoints.
- Low delays. Unexpected pauses can disrupt the natural flow of a conversation and discourage people from using the connection. The network needs to transmit and receive messages without unnatural delays.

- Multiple messages. Once the endpoints were connected, they could exchange numerous messages.

Circuit switching also had these disadvantages:

- Concurrent receipt. The sender can't send a message until the recipient is present to receive it.
- High overhead. The network had to set aside the resources to carry a person's voice even while the person wasn't actually speaking.

10.2.3 Packet Switching

Packets are blocks of digital data that are neither too large nor too small to handle efficiently. Each packet travels across the network independently. Small messages travel as a single packet. Larger messages are broken into pieces, and each travels in its own packet (Figure 10.7).

This is a completely different approach from either message switching or circuit switching. This strategy emerged in the 1960s as researchers imagined new ways to build self-managing computer networks.

Packet switching is similar to sending messages on postcards. We can easily send a short message, but a very long message poses a challenge. We still can send the message, but we must write part of it on each card, and then send the cards individually. Each postcard travels through the postal service (the network) separately. The recipient can start reading the message as soon as the first postcard arrives, but the message might not make sense until *all* postcards arrive.

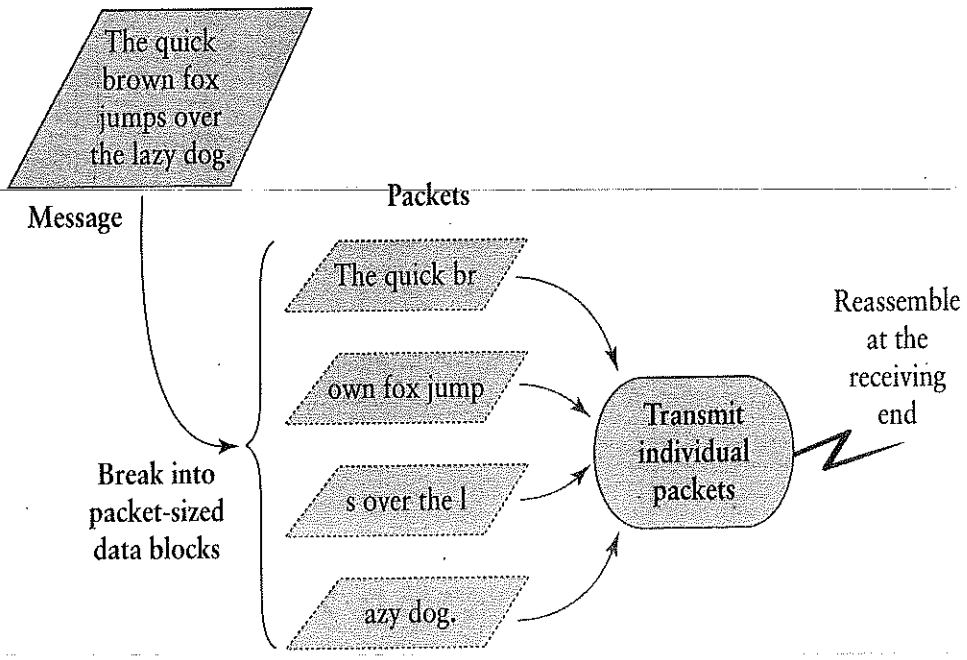


Figure 10.7

A packet network sends messages a packet at a time.

From the network's point of view, a packet needs the same basic information as a postcard:

- The destination—required, or the network can't deliver it.
- Damage detection—implicit in postcards, because we see any tears, smudges, or other injuries.
- The data to carry—optional, but a card seems pointless without it.
- The sender—optional information; the sender writes down a claimed identity but the claim is not always trustworthy.

Every packet contains a special data area, the *packet header*, which contains this information. The network itself may use the header information, or it may be used by the recipient host.

Packet switching provides these advantages:

- Resource efficiency. The network only sets aside enough resources to handle individual packets.
- Flexible routing. Each packet may be routed individually even if all packets are going to the same destination. If network conditions change while packets are in transit, each may take the best path the network finds for that packet.
- Parallel transmission. When we send a series of packets to the network, each packet starts on its way as soon as the network accepts it.
- Service flexibility. We can build an efficient circuit- or message-switched service with a packet service.

Packet switching also has these disadvantages:

- Reliability. Either the endpoint hosts or the network itself must have additional technical features to guarantee reliable and orderly transmission.
- Variable delay. Different packets traverse the network at different rates. There is no way to control or even predict the delay encountered.
- Concurrent receipt. The network only delivers packets to recipients who are actively accepting them. The sender may send packets without knowing if the recipient is available; if the recipient can't or won't accept the packets, then they are discarded.

Like postcards, it's possible to lose packets in a computer network. Technology is never 100 percent effective, and packets can disappear into a network without a trace. Likewise, the post office won't know which postcards, if any, disappeared.

The simplest and lowest-cost packet networks, like Ethernet, take no special steps to detect lost packets and ensure their delivery. When the network does not take steps to ensure reliable delivery, we call the packets *datagrams*.

Many of the earliest packet networks tried to ensure that all packets were delivered in the order they were sent. This required additional information in packet headers, and a set of rules to identify packets that were—or were not—received. This provides a *protocol*, a set of rules to establish effective communication.

To provide reliable packet delivery, a reliability protocol had to be incorporated either into the network or into the endpoint hosts. Experience showed that the system achieved higher reliability by placing that task in the endpoint hosts. Thus, the network itself focused on packet transmission. The endpoints kept track of packet order and verified packet delivery.

Mix-and-Match Network Switching

Different types of communication require different types of handling. Regardless of the underlying network, we often have to transmit messages, send a stream of data, or send a number of short messages to many destinations. Although message, circuit, or packet switching may seem ideal for specific types of traffic, we can in fact carry any type of traffic atop any type of switching.

The first packet-switching networks were built atop the circuit-switched telephone network using leased lines. Today, many telephone networks rely on packet-switched backbone networks to provide circuit-oriented connections. We also use the packet-switched Internet to provide reliable circuit-oriented connections between hosts and to deliver message-switched email. Researchers even have encapsulated packets in messages to provide packet switching atop message-switched systems.

10.3 Putting Bits on a Wire

Digital systems store and transmit bits. Analog systems store and transmit a value within some range. Digital systems store values more reliably because they self-correct.

When a circuit handles an ***analog signal***, it can be hard to tell if the signal's value is correct or if it has picked up some noise. The analog signal could carry any value within its range; if a "glitch" of some sort changes the signal a little, there is no way to detect and correct that change.

When handling a ***digital signal***, the electronic circuits can self-correct minor errors. We use analog electrical signals to represent digital data. For example, the circuit in Figure 10.8 treats a 0 volt signal as digital 0 and a 3 volt signal as a digital 1.

A typical glitch might modify the 0 volt signal by a few percent while it travels from one part of a circuit to another. The resulting signal might reach 0.1 volts. However, even if the glitch changes the signal by 30 percent, the value still represents a 0 bit. The next stage of the circuit will correct the value back to 0 volts. This is how digital circuits self-correct.

This also illustrates how circuits convert between analog and digital signals. To transmit a digital value within an analog signal (for example, to transmit on a phone line), we *modulate* the analog signal to reflect the digital signal value. To convert the analog signal back to its digital value, a receiving circuit *demodulates* the signal. Again, minor glitches may modify the analog signal, but the circuit can self-correct. The acronym ***modem*** describes a device to send digital data on an analog link: a "modulator-demodulator."

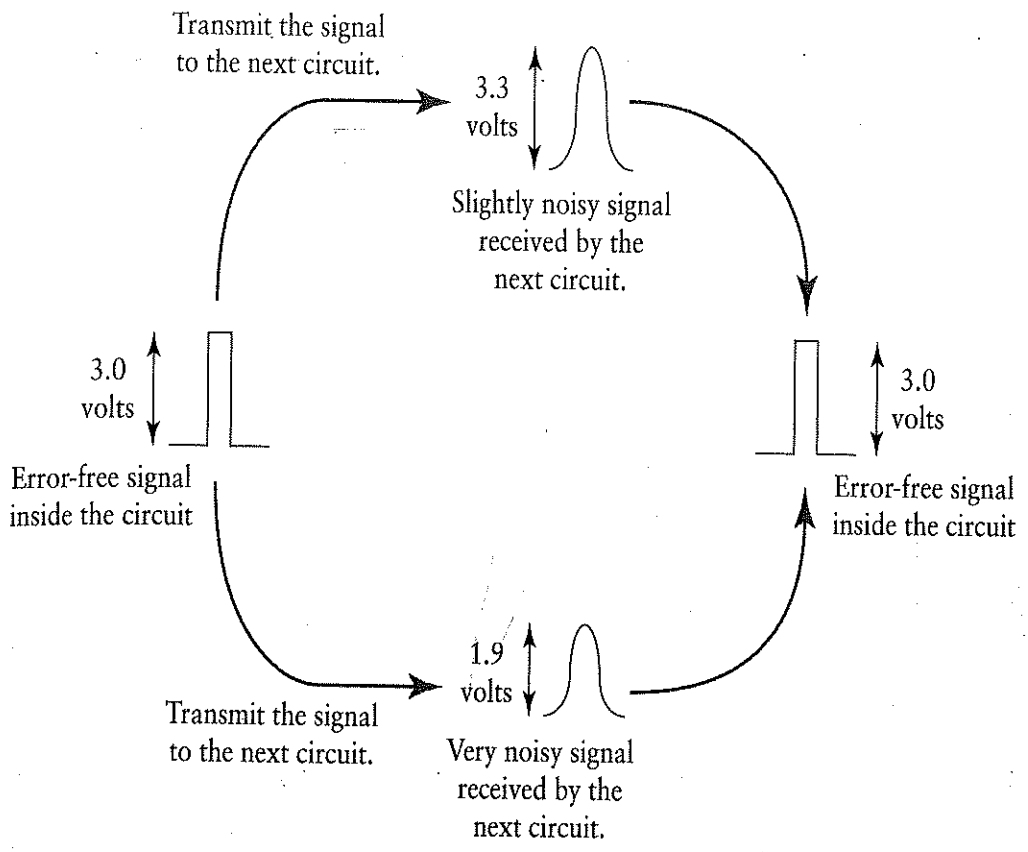


Figure 10.8
Digital circuits self-correct minor signal glitches.

Synchronous Versus Asynchronous Links

We build network links out of wires, optical fibers, and broadcast radio signals. When we transmit digital data, we code the bits into signals that travel reliably across the link. If we use a simple coding like the one in Figure 10.8 (3.0 volts = a 1 bit; otherwise it's a 0 bit), how do we tell when the link carries data? If nothing is being sent, we might expect the wire to carry 0 volts. On the other hand, maybe the sender is sending a really large number of 0 bits for some reason. And if the signal swings up suddenly, how many 1 bits does that represent?

Some network links, especially in older systems, sent data continuously. These were called *synchronous* links. Any time a receiver looked at the data link, it would be carrying data from the sender. If there was in fact no usable data to send, then the link transmitted a special "Idle" message that told the recipient to ignore the data.

Older teletypes and newer, faster networks use *asynchronous* links. A teletype line carried no current except when sending a typed character. When the clerk typed on the teletype keyboard, it sent the typed character as a sequence of signals. The first signal was a "1" value that indicated the start of a character. After the start signal, the character code of 0s and 1s followed, each bit a few milliseconds long. At the end, the current returned to a zero level until the next character was typed. Modern network devices, like the Ethernet, are also asynchronous, but they use more sophisticated signaling techniques.

10.3.1 Wireless Transmission

Wireless or radio transmissions format binary data in the same way as wired transmissions. We define a particular pair of signals as representing 0 and 1, respectively. The transmitter produces the exact signal, and the receiver looks for a signal near a 1 or 0.

Wireless systems fall into four general categories:

1. Low power, local connections
2. Directional connections, like microwave
3. Terrestrial wide-area broadcast radios
4. Satellite-based networks

Frequency, Wavelength, and Bandwidth

Sounds and radio signals travel in waves (Figure 10.9). When we speak of *frequency*, we refer to the rate at which the wave cycles from beginning to end. Higher frequencies, thus, have shorter wavelengths. We describe frequency in terms of cycles per second; the unit of measure is *Hertz* (Hz).

We call the wave's measured height its *amplitude*. The simplest way to transmit sound with a radio is to vary the amplitude to match the change in sound.

The earliest radios transmitted Morse code, so they essentially broadcast a series of binary "on" and "off" signals. The data transmission rate was expressed in *baud*, which counts the number of symbols transmitted per second. The term was named for Emile Baudot (1845–1903), the French teletype inventor.

Early equipment sent data one bit at a time, and not very quickly by modern standards; the maximum data rate, or ***bandwidth***, was around 100 bits/sec. By the 1970s, traditional telephone circuits transmitted digital data at 300 baud. However, clever designs embedded multiple values in each signal, yielding a bandwidth of 1200 bits/sec, and then 19,200 bits/sec.

As designers tried to transmit more and more data over a narrow signal channel, they hit a limit; we can transmit bits at best at *half* the rate of the underlying signal. For example, if a radio circuit is limited to transmitting at 80,000 Hz or less, then its maximum

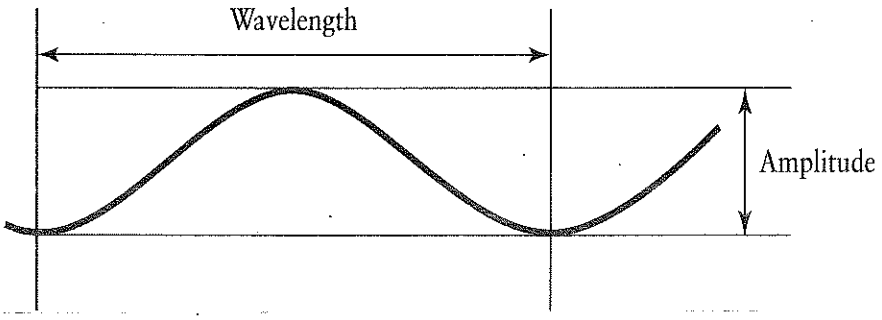


Figure 10.9

Amplitude and wavelength.

bandwidth is 40,000 bits/sec. This restriction allows the receiver to reliably distinguish between the presence and absence of a signal.

AM and FM Radio

Traditionally, radio transmissions are assigned to specific frequency bands for specific purposes. In the United States, the Federal Communications Commission (FCC) assigns radio frequencies. Traditionally, each radio and television (TV) station is assigned a particular frequency for their broadcasts.

Each station has exclusive use of that frequency band within its approved broadcast range. Stations in nearby cities used different frequencies, but a station in a distant city may reuse a particular frequency. In theory, the transmitters in the respective cities should not be strong enough to interfere with each others' signals.

Older transmission systems used amplitude modulation (AM) to transmit signals. To receive an AM signal, the radio tuned to a specific frequency. Variations in the signal strength (the amplitude) at that frequency carried the variations in the sound being transmitted (Figure 10.10).

These variations in turn vibrated the magnetic coil in the radio's speaker to produce sound. Unfortunately, any other transmissions on the same frequency will distort the signal. Such transmissions occur naturally through lightning and artificially through electric motors and other modern appliances. This makes AM radios more vulnerable to static and other distortion.

In the 1930s, radio designer Edwin Armstrong developed an improved radio that used frequency modulation (FM). This technique transmitted sound by subtly varying the transmission frequency around the transmitter's basic frequency (Figure 10.10). This approach was far less vulnerable to static and yielded a much clearer and more reliable signal. Subsequent broadcast systems, like television, used FM transmission.

Frequency Sharing

Not all radio systems use dedicated frequencies. Public two-way radio systems often share frequency bands with one another. Each type of radio has its assigned channels within its authorized frequency range. If too many radio users are on one channel, users

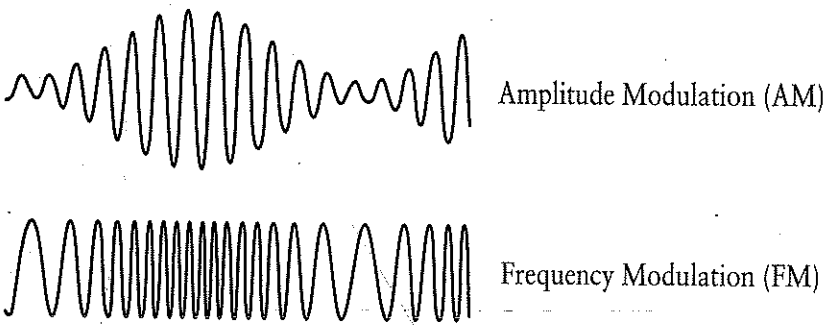


Figure 10.10
AM and FM waveforms.

can switch to another. Examples include the "citizen's band" (CB), marine band, and some commercial two-way radio bands. System users had to be licensed by the FCC; the licensing process is supposed to ensure that radio users know how to share the channels effectively and not interfere with one another unnecessarily.

Digital networks can share frequencies very efficiently. The transmitters can take steps to avoid transmitting at the same time as other stations. High bandwidth techniques can keep transmissions short, giving other stations an opportunity to share the channel. Fortunately these techniques work well, because many digital devices share a relatively small number of channels.

The FCC assigned digital devices a section of the low-power appliance frequency band. Local wireless networks share their channels with portable digital phones, garage door openers, and other products. The channels operate efficiently as long as the devices are designed for efficient channel sharing. For example, wireless LAN devices typically use a multiplexing technique that minimizes interference when multiple devices use the same frequency.

Radio Propagation and Security

Wires have a natural advantage when it comes to security; attackers can't eavesdrop or worse, unless they physically reach the wires. If we physically protect the wires, we protect our network traffic.

Wireless networking is therefore vulnerable to external attacks. Anyone within radio range who uses compatible wireless networking equipment can use our LAN, eavesdrop on our traffic, and even interfere with our messages. Fortunately, modern wireless systems incorporate cryptographic security, as described in Chapter 14. Since being introduced in the late 1990s, wireless security has improved dramatically. This is fortunate because early implementations contained serious vulnerabilities. There is off-the-shelf software available for cracking the earliest wireless LAN encryption.

10.3.2 Transmitting Packets

Figure 10.11 shows the format of a typical packet, which is similar to the disk sector format introduced in Figure 5.5. Like a disk sector, the packet contains three sections:

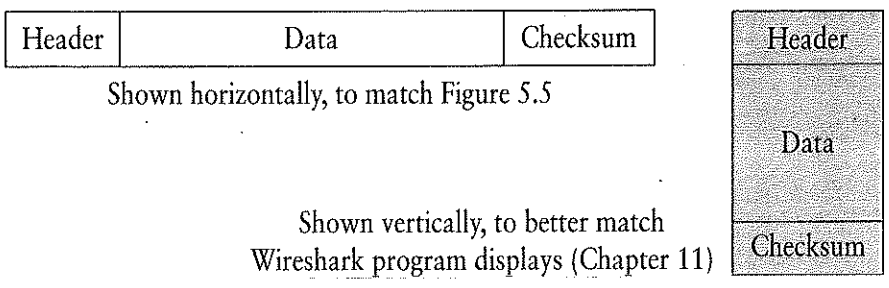


Figure 10.11

Basic packet format.

the header, the data, and the checksum. Like a disk sector, we handle a packet as a single, independent data item. The checksum allows us to detect data errors using the same techniques we applied to disk sectors in Section 5.4.1.

Packet headers contain more data than sector headers. A typical packet header contains the following:

- Address of the packet's sender
- Address of the packet's recipient
- Control information to keep the recipient up to date on the status of transmissions

Many networks use special hardware and software that processes packets without ever looking at the data they carry. However, we mustn't make decisions about handling packets based on faulty data in the header, especially if the data is caused by an error. To avoid this, some networks include a separate checksum in the packet's header. When the network processes a header, it verifies the checksum. If the checksum fails to match its expected value, the network discards the packet.

Acknowledgment Protocol

If the network system is to provide reliable transmission, there must first be a way to detect the success or failure of delivery. We generally send a message called an ***acknowledgment***, or ACK, back to the sender to indicate successful receipt.

When a packet arrives at its destination, the recipient verifies the packet's checksum. If the checksum fails, it discards the packet. If the packet arrives intact, the recipient transmits the ACK. For this very simple example, the recipient sends a separate ACK packet for each data packet received. Figure 10.12 illustrates this protocol using a ***sequence diagram***.

In the sequence diagram, the messages flow between the two network endpoints over a period of time. We begin at the top of the diagram, with the host on the left sending a data packet containing the message "Pay $109." Time progresses as we move downward. The arrow from left to right moving downward shows the data packet traveling to its destination. Midway, the packet arrives at the destination host, who responds with an "ACK" packet. The exchange ends at the lower left, when the ACK arrives at the sender, telling it to send the next packet.

This simple protocol ensures that the sender and recipient both understand the status of the transmission. This is a basic example of ***distributed processing***; two computers, remote from one another, exchange information to achieve a shared goal. In this case, the goal is simple: Ensure that data moves from one computer to the other.

We make the protocol more efficient if we revise it to transmit several packets at once, without awaiting individual ACKs. We can number the individual packets, and the sender indicates the packet number when sending ACKs. The Internet's ***Transmission Control Protocol***, or TCP, uses a similar strategy. TCP numbers each byte consecutively. When sending an ACK, it transmits the number of the *next* byte it expects to receive. If it receives a packet out of order, it does not report its bytes as received until the earlier packets arrive. We study TCP further in Chapter 12.

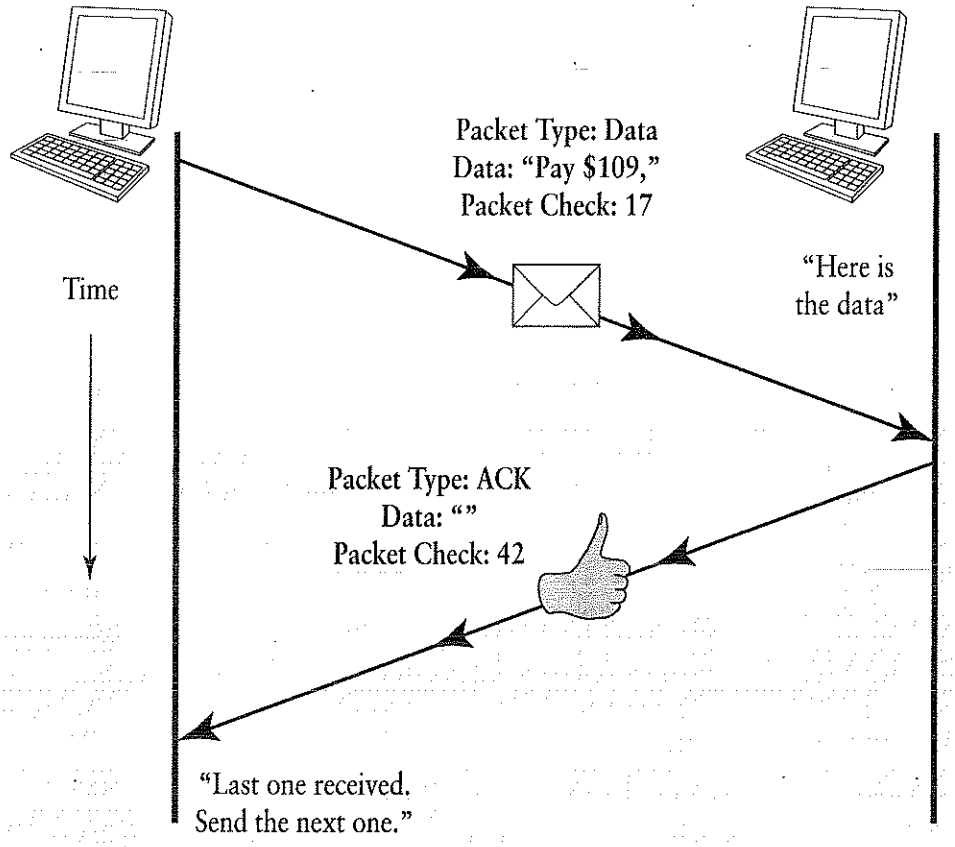


Figure 10.12

A sequence diagram showing a simple ACK protocol.

Network Efficiency and Overhead

When we assess the efficiency of different networks, we look at the ratio of the total number of bits the system can carry and the amount of end-user data actually carried. In packet networks, we calculate packet efficiency with the following ratio:

$$\frac{\text{Size of the data field in bits}}{\text{Total packet size in bits}}$$

Packet efficiency provides a simple estimate of network efficiency. Packet headers and checksums represent networking overhead. The users choose the data to send in the data field, and the rest of the packet is required to make the network operate. Thus, we try to make the packet header as small as possible, but no smaller.

Packet efficiency isn't the only element in network efficiency. We introduce other inefficiencies if we omit essential information. For example, we could omit the sender's address from packets. In theory, the network only needs the recipient's address to deliver a packet. In practice, however, it makes sense to include both addresses. Some networking hardware uses the sender's address to ensure that responses go to the correct host.

Moreover, the recipient host needs the address of the claimed sender in order to respond with an acknowledgment or with other data. If omitted from the packet header, the sending host would still need to provide the information in the data field.

10.3.3 Recovering a Lost Packet

Whenever a host sends a packet, it sets a deadline for receiving the corresponding ACK. If the recipient doesn't return an ACK by then, the sender assumes the packet was lost or damaged somehow. The sender then resends the packet. This is called a ***timeout*** mechanism.

Some protocols also use a "negative ACK," or NAK, packet. If the recipient knows that the packet should have arrived or the packet arrives with an intact header but damaged data, then the recipient may send a NAK. Most protocols rely on a timeout instead because the recipient host can't detect all lost packets. For example, the host might not expect another packet or might not know when the next packet *should* arrive. If the host receives a packet with a damaged header, it won't know what host sent the damaged packet.

Network traffic moves most efficiently if we avoid sending unnecessary packets. Even so, a recipient shouldn't delay in sending ACKs. The recipient should process all packets it has received and try to send ACKs in a single batch. However, the recipient should never wait for "just one more" packet before sending an ACK.

As noted earlier when discussing TCP, the hosts must uniquely mark each packet to indicate the data items they contain. Ideally, the marking indicates the order in which the packets were sent. This allows the recipient to put packets in the correct order. It also allows the recipient to detect duplicate packets. An inevitable side effect of retransmission is that we will occasionally create duplicate packets.

If the ACK is delayed too long, the sender will retransmit the packet. This yields the right result if the packet was actually lost. If the ACK was lost or simply delayed, the retransmission creates a duplicate packet. The sequence diagram in Figure 10.13 shows how this happens.

The sending host transmits a series of packets. The sender sends packet 2, and the recipient sends the ACK in a timely manner. Unfortunately, the network delays the ACK packet for some reason. Because the sender doesn't receive the ACK within the timeout interval, it retransmits packet 2. The recipient then receives a second copy of packet 2.

Although this may look troubling, it is exactly how the protocol should work. Each host only sees the packets it actually receives, and it can't track the status of packets it has sent. The only way a host knows *anything* about its packets is if another host, or node, sends it a packet to indicate what happened. It may seem inconvenient or complicated to deal with duplicates, but it is the price we pay for reliable data transmission.

Modern networks rely on the endpoints to retransmit lost packets, but this is effective only if packets are very rarely lost. Most modern LANs provide adequate reliability and rarely need to retransmit a packet.

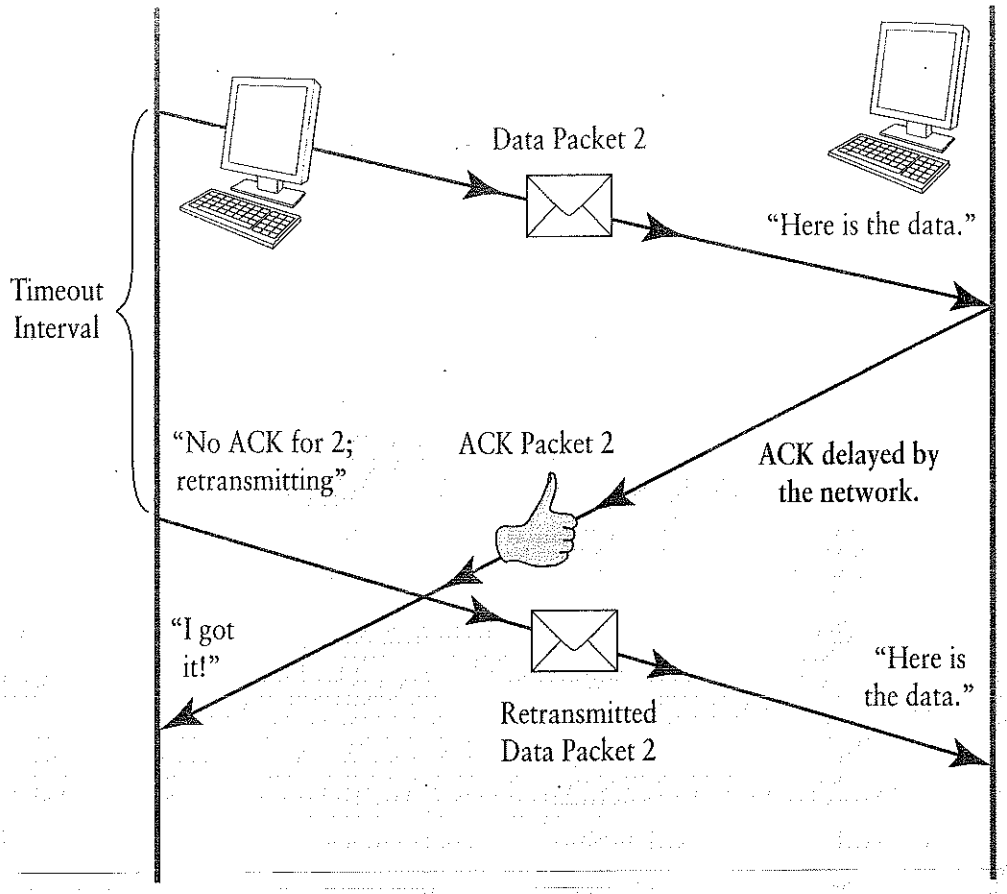


Figure 10.13

A delayed ACK yields a duplicated packet.

10.4 Ethernet: A Modern LAN

Most modern LANs use a form of the Ethernet network. When originally developed at Xerox PARC, Ethernet provided a bus network using a ***coaxial cable*** ("coax"). This is the same type of cable used with cable TV connections: an insulated metallic jacket surrounding an insulated core with a single, thick wire. Each host computer connected to the shared coax. Modern Ethernet rarely uses coax, but it often still uses a bus connection (Figure 10.14).

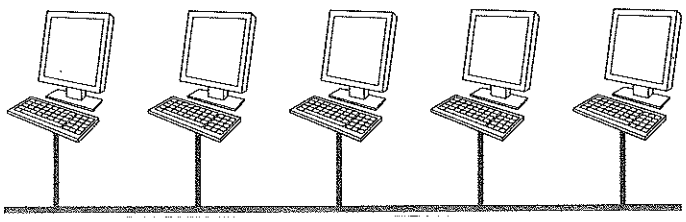


Figure 10.14

The Ethernet LAN connects hosts on a bus.

Because the Ethernet uses a bus, the hosts must take turns using it. To send a message, a host first listens to the bus to ensure that no other hosts are sending a message. If the bus is silent, the host sends its message. While sending, the host listens to the bus. Ideally, the host will hear only the packet that it sends itself. This indicates that the other hosts should hear the packet clearly, including the destination host.

Because Ethernet uses a bus, all hosts can hear all packets. Each packet header contains the source address of the sending host and the destination address of the receiving host. Each host listens for its own, distinctive destination address. If a packet contains a host's destination address, then its network interface will copy the entire packet into the host's RAM. If the packet contains some other destination address, the network interface ignores the packet.

Today's Ethernet

Although the Ethernet was originally a proprietary protocol created and promoted by particular companies, it is now an industry standard that covers a broad range of networking technologies. Local and metropolitan networks fall under the IEEE 802 standard, and the 802.3 standard covers coax-based Ethernet as well as modern versions using copper wires or optical fibers. Wireless networking falls under IEEE 802.11. In general, 802 networks use similar packet structures. This allows the hardware to change dramatically while not requiring major software changes.

Wired

Wired Ethernet networks use cables containing twisted pairs of wires and telephone-style "registered jack" RJ-45 plugs and jacks. Ethernet devices usually are marked to indicate their operating speed and the type of connectors it requires. Here are networks commonly seen in household and small business computing equipment:

- 1000baseT—1000 Mb/second (1 Gb/sec), using RJ-45
- 100baseT—100 Mb/sec, using RJ-45
- 10baseT—10 Mb/sec, using RJ-45

A few older networks might use 10base2, based on the original Ethernet coaxial cable.

Optical

Faster networks may use *optical fiber* instead of copper wires. These networks transmit messages using lasers along fibers that carry the light waves even around cable bends. Optical technology achieves the highest performance of today's networks.

Wireless

The 802.11 wireless network technology evolved during the 1990s. Today, it appears in several common variants:

- 802.11b—2.4 GHz frequency; maximum 11 Mb/s data rate
- 802.11.g—2.4 GHz frequency; maximum 54 Mb/s data rate
- 802.11n—2.4 and 5 GHz; maximum 150 Mb/s data rate

The *Wi-Fi* trademark appears on many, but not all, 802.11-compatible products. The Wi-Fi Alliance is an industry group that promotes 802.11 technology and establishes product interoperability standards. A device becomes "Wi-Fi Certified" if it passes the alliance's interoperability tests. Although the term Wi-Fi is not synonymous with 802.11 wireless networking, many circles treat it as such.

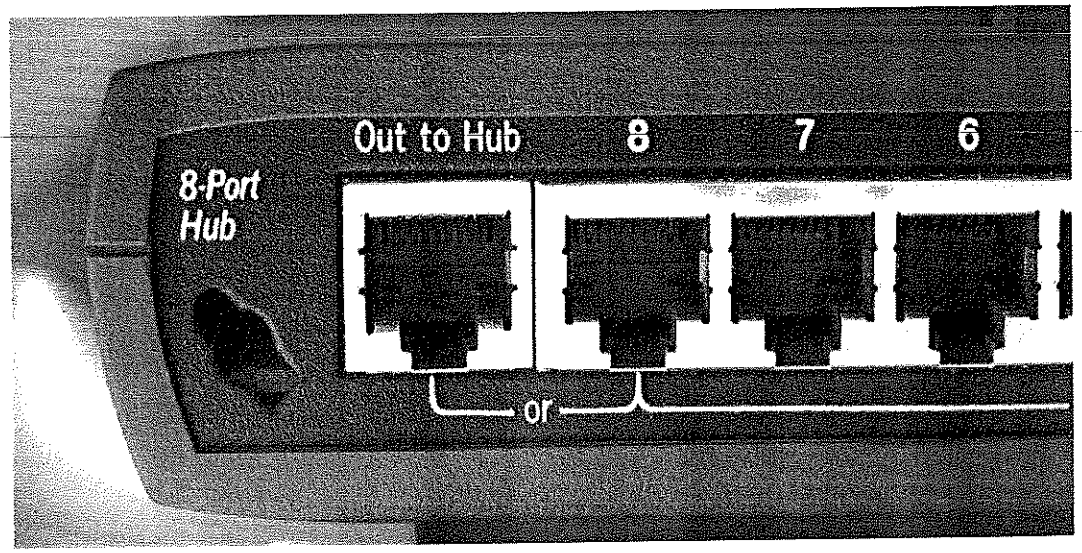
10.4.1 Wiring a Small Network

Most modern computers include a built-in Ethernet interface that provides a 100baseT or 1000baseT connection. To build a small network, we connect to one of these types of nodes:

- *Network switch*—high-speed node that connects pairs of hosts together using the Ethernet LAN protocol.
- *Network hub*—lower-speed node that shares Ethernet packets among the connected hosts. These usually are restricted to 100baseT or 10baseT speeds.

We can build a small network of four, five, eight, or a few more hosts, with a single node. The node provides a single RJ-45 socket for each connection. Figure 10.15 shows the sockets on the back of a typical device. To connect more computers, we connect the switch or hub to either another switch or another hub.

While it is simple to build a smaller network, a larger network requires careful design. The network must form a hierarchy of nodes. This provides exactly one route between every pair of endpoints. If we cross-connect branches of the hierarchy, the nodes won't be able to deliver the packets reliably.



Courtesy of Dr. Richard Smith

Figure 10.15

RJ-45 connectors on an Ethernet hub.

Network Cables

Wiring for telephone and computer networks fall into standard grades and categories. The lowest are for traditional home telephone wiring. A modern high-speed 100baseT network requires *Category 5* wiring, nicknamed "Cat 5." Faster networks, like 1000baseT, require *Category 5 Enhanced* ("Cat 5 E"), or Category 6 wiring. These categories indicate the number of strands of wire, the wire sizes, the insulation, and how they are twisted together.

Traditionally, Ethernet-style RJ-45 cables are directional; the connection to the computer drives the electricity in a particular direction on each wire. When we plug the cable into another device, it must be wired to accept the signals in the opposite direction. For example, we can't always plug the ends of one cable into two computers; both will send a signal "outward" on the same wire.

This is true of the hub shown in Figure 10.15. The numbered ports can connect to a computer; they are called *downlinks*. The leftmost port, marked "Out to Hub," is an *uplink*, which may plug into a downlink on another hub or switch.

Cable directions are less of a problem with recent equipment. Newer, faster 1000baseT switches and network interfaces often can detect the signal direction. This allows us to use any available port to connect switches together.

10.4.2 Ethernet Frame Format

A *frame* is a single data packet on an Ethernet. At the electronic level, an Ethernet interface transmits each frame asynchronously. There are framing signals that mark the beginning and end of each frame. A frame begins with a "preamble" containing 7 bytes' worth of alternating 1s and 0s, followed by a special data byte to mark the start of the frame. The network marks the frame's end with an "idle" signal.

Once a LAN interface receives an Ethernet packet, it interprets the contents as shown in Figure 10.16. This is the "Ethernet II" format, which is the common format in modern LANs.

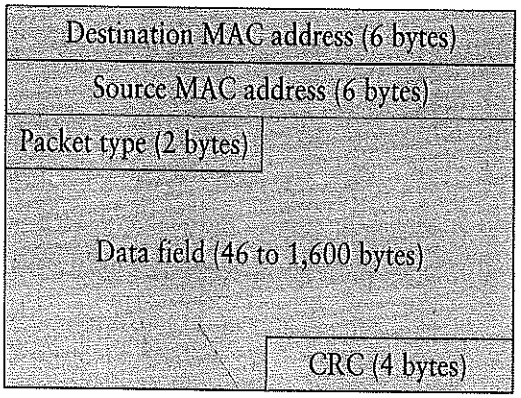


Figure 10.16

Ethernet packet ("frame") contents.

Because the Ethernet idle signal indicates the packet's start and end, the frame itself doesn't need a built-in length field. Some implementations use the "Type" field to contain the length, but Ethernet II packets don't interpret it that way. Instead, it relies on the hardware to find the packet length, and uses those results to find the end of the data field and the 4-byte CRC value.

LAN Addressing

Addresses are very important in Ethernet, because they let each host sort its own traffic out from other host traffic. If we think of the network hardware as our "communications medium," then the address provides us with ***media access control*** (MAC). Because of this, we typically call an Ethernet address a *MAC address*.

Most network interfaces contain a built-in MAC address. These addresses must be unique. If two hosts had the same MAC address, they could not distinguish between each other's traffic on the LAN. To ensure uniqueness, each manufacturer is assigned a range of addresses. Each manufacturer assigns a unique address within their assigned range to every interface they manufacture.

Address Format

The standard MAC address contains 48 bits arranged as shown in Figure 10.17. The first 24 bits identify the manufacturer. The remaining 24 bits are assigned by the manufacturer to the individual LAN interfaces they manufacture.

Because MAC addresses are assigned in blocks of bits and bytes, it is most convenient to refer to them in hexadecimal notation. The entire address consists of 12 hex characters grouped in 8-bit pairs. A typical address in hex notation might be:

00:17:F2:00:C3:F9.

The following are a few of the 24-bit (3 byte) identifiers assigned to major manufacturers:

- 00:01:02 3Com Corporation
- 00:01:42 Cisco Systems
- 00:02:B3 Intel Corporation
- 00:06:5B Dell Computer
- 00:17:F2 Apple Computer

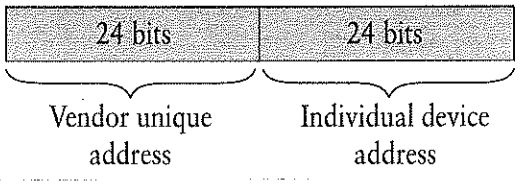


Figure 10.17

MAC address format.

This is by no means a complete list; the latest list is over 1,500 pages long. Many larger vendors, including those just listed, actually have two or more blocks of addresses assigned to them. The additional blocks ensure that no vendor, no matter how large, has to reassign an older MAC address to a newer product. Thus, every MAC address is unique, no matter when it was assigned.

In addition to the individual interface addresses, the Ethernet also uses a preassigned address of FF:FF:FF:FF:FF:FF to indicate packets that should be broadcast to all hosts on the LAN. If a packet uses this as the destination address, then all Ethernet interfaces will accept the packet as if it were addressed to them individually.

MAC Addresses and Security

Because MAC addresses are unique and assigned by the hardware manufacturer, they would seem to provide an ideal unique identifier for every network device. In fact, some firewall and gateway products will block or pass network traffic based on MAC addresses (see Section 12.4.2).

For example, the suite might be able to configure Eve's printer so that it only accepts packets from computers belonging to suitemates. We enter the accepted MAC addresses into a network traffic filtering device or perhaps into the printer itself.

Unfortunately, this isn't very reliable in practice. Most Ethernet interfaces allow the computer software to change the built-in Ethernet address to a different value. For example, we will assume that the Old Masonic cooperative decided to give Alice a legitimate connection, and is using MAC addresses to identify approved network users. One of Alice's clerks wants to connect her own laptop to the network. She can't simply connect her laptop because the printer and network gateway won't recognize her MAC address. If she knows an approved MAC address belonging to someone else, she can configure her laptop to use that other MAC address instead. If Alice isn't in the store, she can use Alice's MAC address and masquerade as Alice.

10.4.3 Finding Host Addresses

Except for switches and hubs, every device on an Ethernet LAN has a unique MAC address. We often can find the addresses by inspecting the device. Because MAC addresses are assigned by the manufacturer, we may find the MAC address printed on a label attached to the LAN plug. When a network printer prints a network status page, the page often includes the MAC address.

Most computer systems have one or more utility programs that can retrieve the system's MAC address. Systems have keyboard commands to display network addresses as well as window- and menu-based commands.

Addresses from Keyboard Commands

All Unix-derived operating systems, including Linux and OS X, provide the MAC addresses by typing "ifconfig" into a command shell. The output is dense and arcane, but we will find the contraction "ether" followed by a six-byte hexadecimal MAC address.

```
C:\Users\ob>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : BobWin7
   Primary Dns Suffix . . . . . . . : comcast.net
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled . . . . . . . : No
   WINS Proxy Enabled . . . . . . . : No
   DNS Suffix Search List . . . . . : smat.us

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix . . . . :
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Network Connection
   Physical Address . . . . . . . . . : 00-35-3C-29-35-20
   DHCP Enabled . . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::3be3:9c73:55d8:6832%11(Preferred)
   IPv4 Address . . . . . . . . . . . : 10.20.30.28(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained . . . . . . . . . . : Saturday, May 01, 2010 12:01:09 PM
   Lease Expires . . . . . . . . . . : Sunday, May 02, 2010 12:01:09 PM
   Default Gateway . . . . . . . . . : 10.20.30.10
   DHCP Server . . . . . . . . . . . : 10.20.30.10
   DHCPv6 IAID . . . . . . . . . . . : 234901590
   DHCPv6 Client DUID . . . . . . . . : 00-01-00-01-BB-23-4F-26-00-35-3C-29-35-20

   DNS Servers . . . . . . . . . . . : 10.20.30.10
   NetBIOS over Tcpip . . . . . . . . : Enabled
```

Figure 10.18

Windows command to retrieve host addresses.

On Microsoft Windows, we must start an MS-DOS command window to use keyboard commands. The MS-DOS command "ipconfig" prints out address information. Figure 10.18 shows the printout of this command; the command itself is underlined. The MAC address is listed under the "Ethernet Adapter" as "Physical Address."

Addresses from Mac OS

We also may find the addresses with Mac OS "System Preferences" by following these steps:

- Open "System Preferences."
- Select "Network."
- On the left-hand side of the screen are the network interfaces. Choose the one of interest. It should be active and contain a green indicator. This will display basic information about that interface.

- Click the "Advanced" button in the lower right. That displays a set of tabs giving information about different network services and protocols.
- Click the "Ethernet" tab on the far right. This will display the MAC address.

Addresses from Microsoft Windows

Microsoft also provides a graphical interface. We find the addresses by examining the network interface, which we may reach through the "Network and Sharing Center." Open the "Network and Sharing Center" and follow the steps depending on what you see.

- Look on the right-hand side. If there is a section in the upper middle saying "Access Type: Internet" and "Connections: Local-Area Connection," then:
  - Click on the "Local Area Connection" to display the Network Status for that device.
- Otherwise, look on the left-hand side. There is a list of links to other functions.
  - Click on "Manage network connections." This opens a display of physical network devices.
  - Right-click on the local networking device. This produces a context menu. Select "Status" from the menu to display the Network Status for that device.
- In the middle left of the Network Status window, there is a "Details" button. Click on the button.
- This opens the "Network Connection Details" window. The third or fourth line lists the "Physical Address," which gives the MAC address.

10.4.4 Handling Collisions

Host computers use two mechanisms to share the Ethernet reliably and safely. First, the hosts listen before transmitting as noted earlier. Second, the hosts look for ***collisions*** and automatically retransmit their packet if they detect one. A collision occurs if two or more hosts try to transmit a packet at once. This can happen even if both hosts listen before transmitting; they might be far enough apart so that one host doesn't detect that the other has started.

Hosts will detect a collision because they listen while they transmit. If they don't hear their own packet on the network as they transmit, they know the signal has been mixed with another host's transmission. When this happens, the host immediately stops transmission, then it picks a random amount of time to wait. Once that wait time expires, the host tries to transmit again. If the network is busy, the host waits for the other host to finish. Otherwise, it transmits its packet again.

If the host detects another collision, it randomly picks another wait time, but doubles the possible range of the wait time. Thus, if it collides again and again, it will wait longer and longer before it tries again. Eventually it will find the network idle, and it will transmit its packet. This is called "exponential backoff."

Wireless Collisions

A wireless LAN faces a more significant collision problem. At first, it may seem like a similar problem with a similar solution: The transmitter simply needs to listen for an idle channel before it transmits. However, the packet travels between two devices, and they may be far away from one another. Even if the transmitter hears an idle channel, the receiver might be in range of a different, and active, transmitter.

The wireless protocol takes two steps to address this problem. First, the transmitter and intended recipient both transmit messages to warn away other traffic. Figure 10.19 illustrates the procedure. Second, the recipient will immediately request a retransmission if a transmission fails.

In Figure 10.19, Host 1 wants to send a message to Host 2. At about the same time, Host 3 wants to send a message to Host 2. Hosts 1 and 3 are out of radio range of one another, and Host 2 is between them.

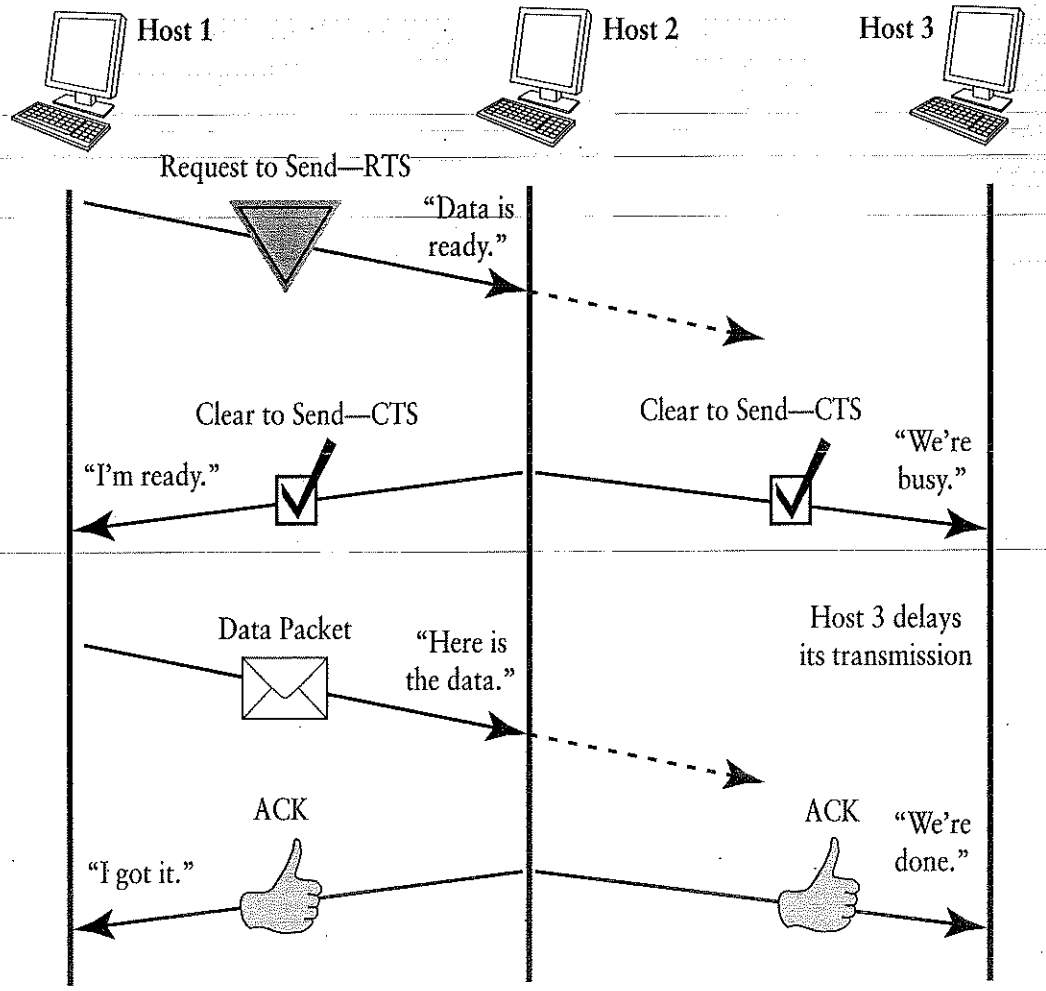


Figure 10.19

Wireless protocol for avoiding collisions.

Before sending the actual data packet, Host 1 transmits a warning message called *Request to Send* (RTS). The message is addressed to Host 2. The RTS message is small to reduce the risk of collisions. The message warns Host 2, and everyone else in radio range of Host 1, that a packet is coming. Because Host 3 is out of radio range, it doesn't see the RTS.

When Host 2 sees the RTS, it responds with a *Clear to Send* (CTS) message. This tells Host 1 that it's ready to receive the message. It also warns everyone within its radio range that Host 2 is awaiting a message from Host 1. Host 3 sees the CTS message and delays its transmission.

Upon receiving the CTS, Host 1 transmits its packet. Host 2 waits for the packet to arrive and sends an ACK when it arrives successfully. Because Host 3 hears the ACK, it now knows that Host 1 is through sending its message. Host 3 then may send its own RTS, announcing data it wants to send to Host 2.

Wireless Retransmission

If, after sending the CTS, Host 2 never receives the data packet or it receives a garbled packet, it sends Host 1 a NAK. The wireless protocol is one in which NAKs serve a practical purpose. Host 2 knows that it should receive a packet, and it can reliably tell Host 1 whether the packet arrived. Moreover, wireless transmission is much less reliable than wired transmissions. These retransmissions make the wireless system reliable enough for effective Internet communications.

10.5 The Protocol Stack

In Section 5.7, we saw how operating system designers mastered complexity by organizing the complicated details of OS software into layers. Network designers use the same strategy to produce the network's *protocol stack*. As with the I/O system, the protocol stack forms an hourglass structure (Figure 10.20).

At the bottom of the stack we have device drivers, all for network devices. At the top, we have *application protocols*, which are network protocols designed to do a specific computing task, like file sharing, email, printing, or Web access.

The network protocol stack itself focuses on services shared by device drivers on the bottom and applications on the top. In the LAN world, the stack contains two essential protocols:

- *Transport protocol*—associates a packet with an application process and may also detect corrupted or duplicate packets.
- *Link protocol*—formats the packets for actual transmission and handles MAC addresses for the hosts.

As a message traverses the protocol stack on the way to the device driver, each layer applies its own formatting (Figure 10.21). This often appears as an additional header prefixed to the outgoing message data.

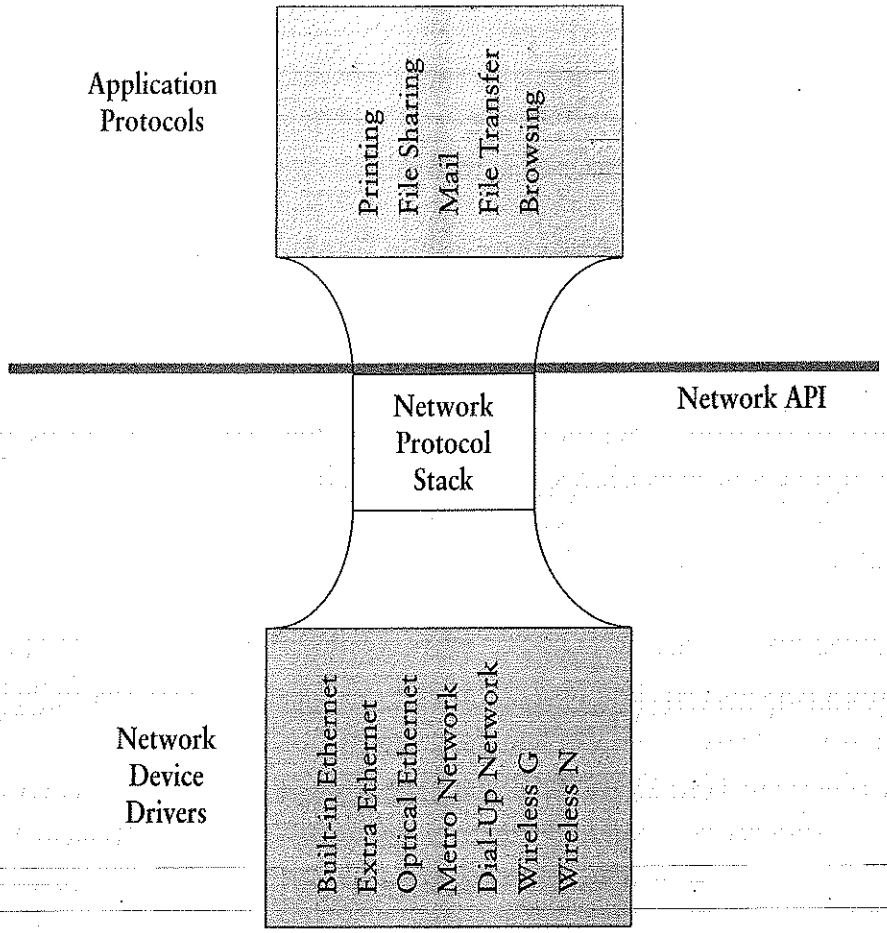


Figure 10.20

"Hourglass" structure of network protocols.

For example, imagine that we are sending an email message. The application layer inserts fields in the email message's header. It opens a connection to an email server and adds more heading information to identify and route the message.

The transport layer collects enough of the message (including the application header) to fill up a packet. The layer leaves the data undisturbed except perhaps to split it into separate packets. It adds its own transport header, which indicates the application process using this connection, and includes any ACKs pending for data from the recipient. The header also may include a checksum calculated over the packet's data. The layer then passes the packet to the link layer along with the destination address. If the message fills several packets, the transport layer fills up as many packets as needed.

The link layer accepts packets one at a time, each with its destination address. The layer adds a header to the packet that includes the MAC addresses for the source and destination hosts and indicates the type of transport header in the packet.

Then the link layer passes the packet to the device driver. The driver tells the network interface to transmit the packet across the network.

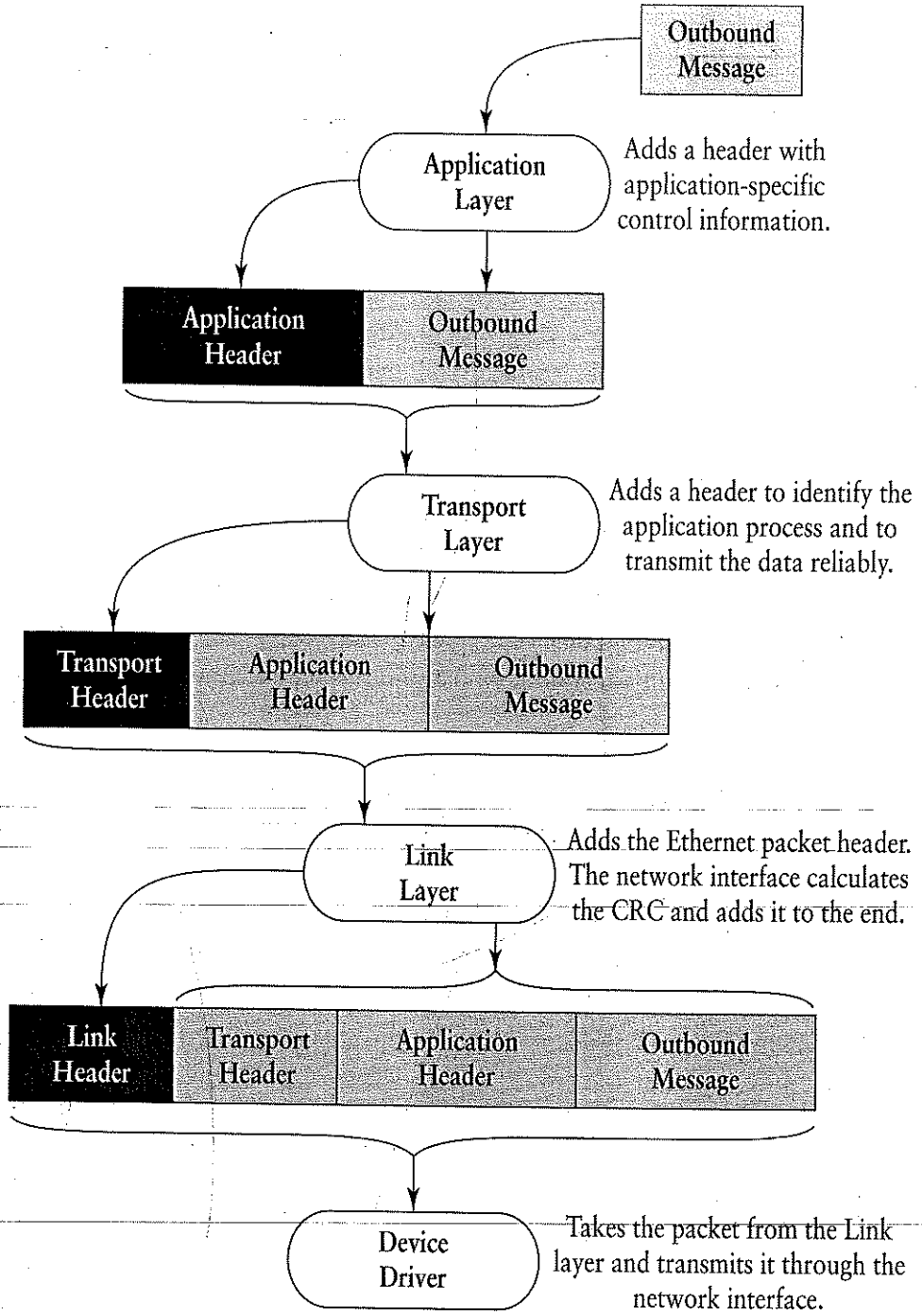


Figure 10.21

Each protocol layer adds a header to the packet.

10.5.1 Relationships Between Layers

As data travels down the protocol stack, each layer adds its distinctive "envelope" around the data. When the packet arrives at the destination host, the actual data is inside this series of envelopes, each added by a protocol layer. Moving up the recipient's protocol stack, each layer removes and examines the envelope added by its twin in the sending host.

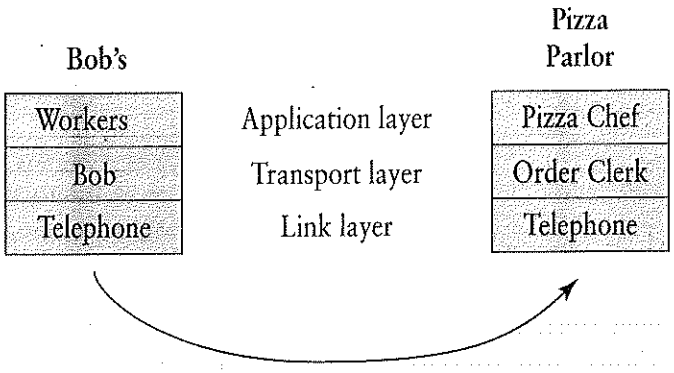


Figure 10.22

Protocol layers in ordering pizza.

In practice, most layers simply add a header and don't actually "envelop" the enclosed data. Protocol layers on different hosts communicate and cooperate by sharing data in their respective headers. William Stallings illustrates this with a simple analogy, shown in Figure 10.22.

> Monday evening finds everyone at Bob's is working late. Tina suggests ordering pizza. After some negotiation, Bob writes down the consensus on pizzas to order and collects money.
>
> Bob phones the pizza parlor. The order clerk talks to Bob, collects the order, and negotiates the price. The total is less than the money Bob collected, so Bob agrees to the order.
>
> The order clerk turns to the pizza chef and gives him the order for the office pizzas. The chef prepares and cooks them.

The essential relationship is between the workers at Bob's and the pizza chef: The chef executes their desires—but they only talk to the chef indirectly, through Bob and the order clerk. Even if they were physically present at the pizza parlor, the chef wouldn't talk directly to the table about their order; she stays in the kitchen and makes pizzas.

The order clerk keeps customers from annoying the chef and ensures the orderly collection and payment of pizza orders. Bob has ordered from him several times before and knows his style, so it's easiest for the suite to let Bob handle the order. Because the workers are at the office and not in the pizza parlor, Bob contacts the clerk by phone.

Figure 10.22 suggests an analogy between our pizza-ordering protocol layers and the local network protocol layers. The "practical end" of the protocol is, of course, the application layer. The transport layer enforces some reliability and structure on the data transfer, while the link layer simply carries data.

10.5.2 The OSI Protocol Model

When the network community speaks of protocol layers, it relates them to a recognized set of layers called the *Open Systems Interconnect* (OSI) *model*. Here are the layers, numbered from bottom to top:

1. **Physical layer**—the physical wiring and signaling between nodes. For example, Cat 5 cables are wired differently from 4-wire telephone cables.
2. **Data link layer**—manages the structure and content of data carried by the physical layer. Though OSI includes the word "data" in the title, we will continue to call it the *link layer*. Ethernet frames are an example of a data link protocol.
3. **Network layer**—manages intranetwork routing of packets. The ***Internet Protocol*** (IP) resides at this layer, and we sometimes call it the "internet layer."
4. **Transport layer**—associates packets with specific application processes in endpoint hosts, and ensures reliability. The TCP protocol resides at this layer.
5. **Session layer**—handles a set of transport connections used for a particular purpose; rarely used today.
6. **Presentation layer**—reformats host data to meet network-wide standards and vice versa; rarely used today.
7. **Application layer**—provides a specific service to the user on a host computer. Examples include email and Web protocols.

There are several mnemonics for remembering the seven protocol layers; for example:

Please Do Not Throw Sausage Pizza Away.

When people speak of "Layer 2" or "Layer 7" network activities, they are referring to the corresponding OSI layers. The simple LAN protocols in this chapter only use four protocol layers, as shown in Figure 10.23. We add Layer 3 when we work with internet protocols.

The hubs and switches that make up a wired LAN are "Layer 2" devices. They carry the transport and application data while paying no attention to its contents. The transport layers at the endpoint hosts ensure the reliable transmission of the application data.

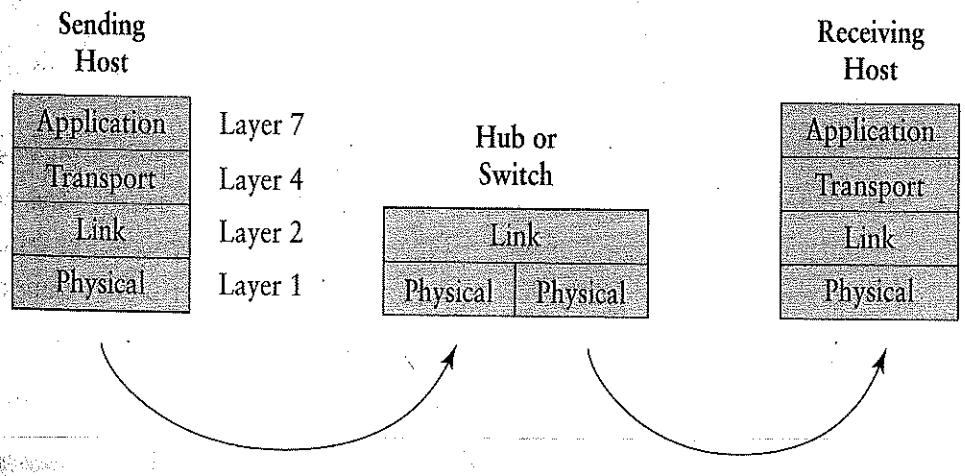


Figure 10.23

Protocol software layers on a simple LAN.

We make two simplifying assumptions when using these protocol layers. First, we treat the physical layer as consisting entirely of link wiring or radio transmissions. Second, we consider everything above the fourth layer, the transport layer, to be part of the application.

Because the physical layer consists of physical phenomena, we implement physical or mechanical security controls at the physical layer. All other controls are the realm of higher-level protocols.

In the world of protocol software, we distinguish between the protocol stack and other components. The protocol stack contains the network software integrated into the operating system. Everything above it is application software. Below it are the device drivers. The link layer may be part of the protocol stack, and it also may be part of the associated device drivers, depending on the network.

The Orphaned Layers

Modern network applications have evolved dramatically since the OSI model first appeared. In fact, the OSI model predates the Internet itself. The operating system incorporates Layers 2, 3, and 4 into the protocol stack. Application software takes responsibility for the higher layers. However, there are no modern protocols that reside in Layers 5 and 6 and also provide their assigned functions.

If we interpret Layers 5 and 6 in the context of modern networking, they handle these two functions:

- Sessions (Layer 5): persistent dialogs between pairs of hosts; each dialog is made up of multiple interactions at lower protocol layers.
- Consistent presentation (Layer 6): the ability to receive standard information from the network and reformat it to display correctly on the host's desktop.

Web browsers have developed their own mechanisms to handle sessions and to handle platform-specific differences for presenting information. On the Web, we implement sessions to provide individuals with personalized "shopping carts" when visiting an e-commerce site. The Web session mechanism associates a particular shopping cart with a particular user desktop. If someone else visits the same site with a different computer or with a different login from the same computer they receive their own shopping cart.

Most Web applications implement sessions using "cookies," a special mechanism built into Web browsers. We discuss this mechanism further in Section 16.3.2.

Web software provides compatible graphics for different systems by providing custom browser software to run on the different platforms. For example, Apple distributes both a Macintosh and a Windows version of their Safari browser. Mozilla distributes different versions of the Firefox browser for Windows, Macintosh, and for different versions of Unix and Linux.

10.6 Network Applications

Most of us are familiar with Web browsers and email clients; we know how to visit websites and send email. Both of these, however, illustrate only half of the networking

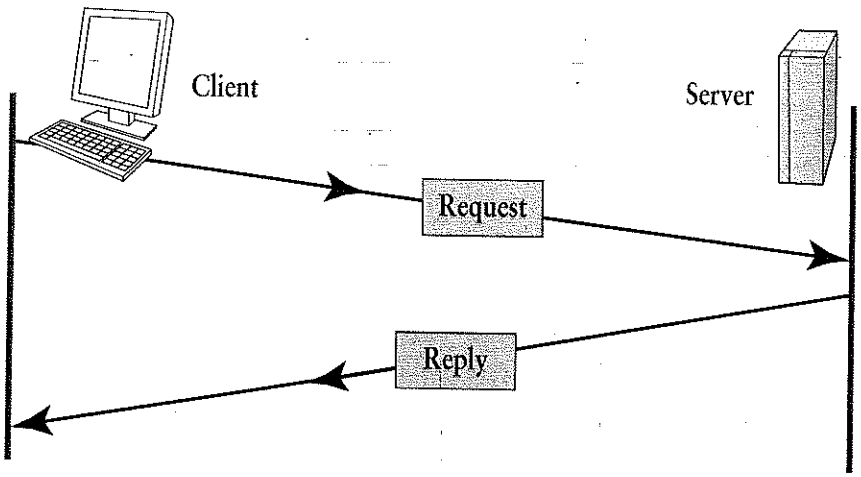


Figure 10.24

A client requests service and the server replies.

task: Web pages must reside somewhere and email messages must be delivered somehow. Web and email services operate in a mode called *client/server*. Our desktop or laptop host (the *client*) requests a network service from a distinctive host (the *server*) with the hardware and/or software to provide that service (Figure 10.24).

For example, when Bob wants to print on the shared printer, he directs his printing software at that printer. Inside the OS, the software makes a connection to the printer's server software, then the OS sends the file to the printer's server, which feeds it to the printer.

The security issues in client/server protocols generally arise from the challenges of network resource sharing. We discuss those challenges in Section 10.6.1.

Servers

The server software provides services for one or more clients. If a client needs service, it contacts the server and requests the service. If the server can provide the service, it does so and sends the result to the client.

The server has a well-known location, or it has one that is easily found. On the Internet, well-known servers have well-known names (google.com, amazon.com, whitehouse.gov, and so on). Resources on local networks don't tend to have well-known names. For example, we can't just assume that the printer has the name "Printer," because there may, in fact, be two printers on the same LAN. We might even have two printers of exactly the same make and model. Instead, LANs often use broadcast protocols to locate resources like printers and shared file storage; then the clients choose the desired resource from a list of locally available resources.

Peer-to-Peer

Client/server isn't the only strategy for providing network services. Internet file transfer and sharing programs like "Bittorrent" use a different strategy called *peer-to-peer*. In

this approach, a server may become a client, and a client may become a server. In essence, the peer-to-peer software package includes both client and server capabilities.

For example, people use Bittorrent to download really large files, like ready-to-run images of Linux hard drives. If dozens of clients all try to download the file from the same server, the server will bog down as it sends the same block of data to dozens of different hosts. With Bittorrent, the hosts work cooperatively to download the file. The server itself may provide different parts of the file to different clients, and these clients in turn distribute those parts to other clients. Thus, each client makes a small demand on the main server, and comparably small demands on the other clients.

Network Applications and Information States

Section 3.5.3 introduced the three fundamental information states: Processing, Storage, and Transmission. Section 7.1.3 described how the states related to encryption. Networking also affects our interpretation of information states.

In that earlier section, we drew a state diagram that did not allow a transition directly from Processing to Transmission. That restriction is not true for network applications. Such programs can send and receive data across the network, thus making the transition between Processing and Transmission.

As noted earlier, information faces risks when it leaves a protected environment. Thus, the network application can put data at risk by changing it to the transmission state: by sending it across a network.

10.6.1 Resource Sharing

In the 1960s and 1970s, the U.S. government spent a great deal of research money on computer networking because it offered a way to share scarce and expensive computer resources among researchers all over the country. In the 1980s and 1990s, the same argument justified installing LANs in offices. Everyone on the network could share a common hard drive; this was important back when hard drives cost $100 per megabyte. Quality printers also were expensive, so printer sharing was another benefit.

Figure 10.25 illustrates the basic notion behind network printer sharing. The box on the top of the figure is Kevin's computer with its own, directly connected printer. When his word processor tries to print the paper he's written, it uses the Printer API to pass the paper to the printer driver. Like any good API, the Printer API provides an abstraction that hides the details of printing from the word processor.

When Bob tries to print his own paper, his word processor also passes the paper through the Printer API. In this case, however, Bob has selected the network printer. Invisible to the word processor, the paper is passed to the "print client," which contacts the "print server" across the LAN. The server accepts the paper and passes it to the printer driver. The printer itself can't tell, and doesn't care, whether the paper came across the network or not.

Network systems use the same general strategy to share all sorts of computer resources:

- **Resource API**—presents a uniform way to use the resource, regardless of whether it is present on the computer or shared over the network.

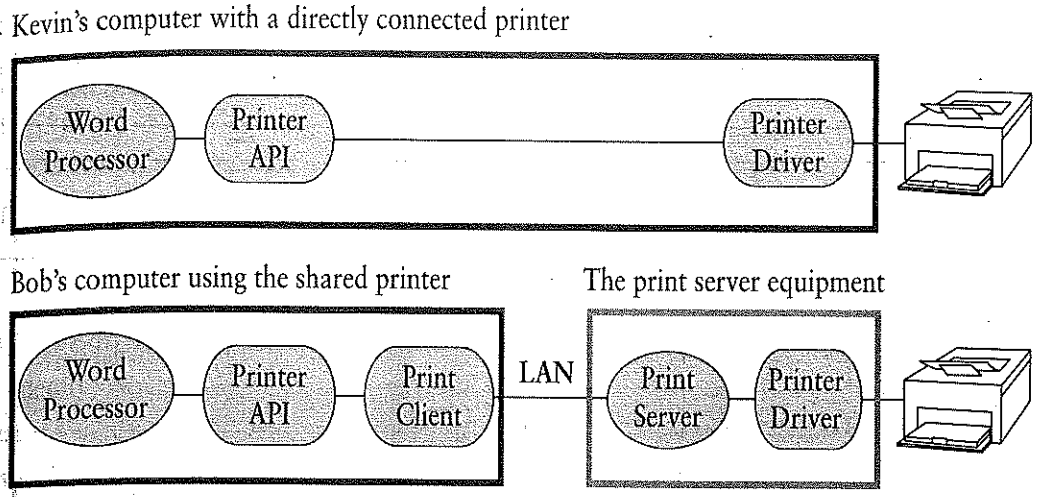


Figure 10.25

A directly connected printer versus sharing one across a LAN.

- **Client**—software contacted by the API to access the resource across the network.
- **Server**—software visible to clients on the network that provides the resource to those clients.

Simple versions of resource sharing would allow anyone on the LAN to use the resource if they had the client software. More sophisticated systems implement access controls. Some systems control access according to the host computer making the request, while others make decisions based on an authenticated user identity. Access controls on resources can be as simple or complex as any established for files, as described in Chapters 3 and 4.

10.6.2 Data and File Sharing

Although many may share printing resources and some may share other I/O or computing resources, data sharing is often a primary requirement. In earlier days, we might share a hard drive to provide additional file storage at a lower cost. Today, cheap hard drives make this unnecessary. Instead, we share hard drives so that we can share data and files.

All modern operating systems provide file-sharing features: One host can connect to another to read and write some of its files. This is a client/server operation. The clients open and close files as if they reside on their own hosts. However, instead of retrieving the file from a local hard drive, they retrieve the file from a "network drive": a drive on a different computer whose contents are available through the network. In practice, the "drive" is usually a shared folder. Although the shared folder may look like a drive on the client side, the only shared files are those residing in the selected folder on the server side.

When using network drives, the servers exchange data with the clients and update the files. For example, Alice could connect to the suite's LAN, allowing both Bob and Alice to share files on each other's computers. Bob treats Alice's laptop as a server while Alice's laptop uses Bob's desktop as a server.

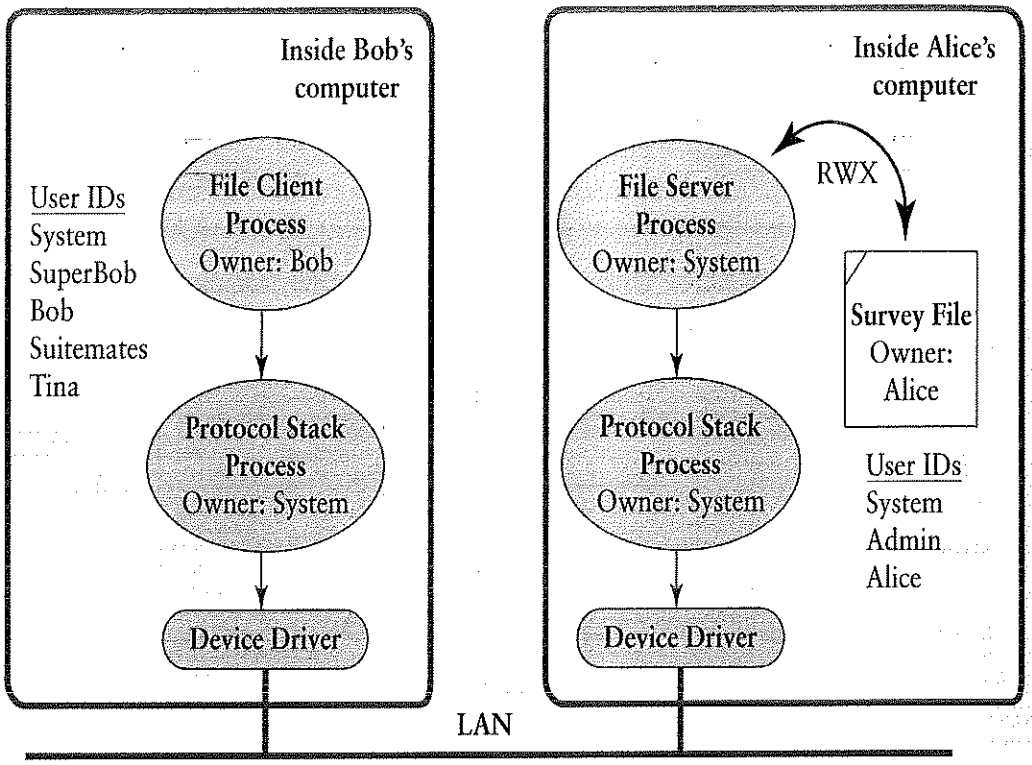


Figure 10.26

Sharing files on a LAN.

File sharing poses a security risk: If a host contains unprotected files, the file service makes them accessible to attackers who connect to the network. Because of this, many modern systems disable file sharing. A computer's owner must activate file sharing before it can be used.

Figure 10.26 illustrates the network software for a simple file-sharing service. Bob and Alice are working on another survey, and Bob wants to retrieve the copy of the survey file residing on Alice's computer.

On Bob's side, he runs a client process that requests the file from Alice's computer. Bob doesn't actually see the client process. Typical systems like Microsoft Windows let Bob select shared folders available to him on other peoples' computers; this is called "mapping" a network drive. Desktop software displays the network drive next to his other local drives. Internally, it activates the client process that accesses the remote files. When Bob tries to retrieve a file from Alice's computer, the file client contacts Alice's server software.

When Alice configured her computer to allow file sharing, it activated the server software. When Bob mapped one of Alice's shared folders, he told Alice's server that he wanted to access its shared files. When Bob retrieves a particular file, the server software retrieves the file and transmits its contents to Bob.

If Bob modifies the file, his client transmits the changes to Alice's file server process. The server process in turn updates the file on Bob's behalf.

Delegation: A Security Problem

The arrangement in Figure 10.26 poses a problem. If we look at the File Server Process in Alice's computer, we see it is run by the System process. Thus, the File Server Process has unrestricted access to Alice's shared files. In fact, the process has unrestricted access to *all* of Alice's files.

Matters aren't any better if we change the process owner to be Alice. We don't want the process to use Alice's permissions when working on Bob's behalf. We want it to enforce Bob's rights when running on Bob's behalf. This is the ***delegation problem.***

Ideally, Alice should be able to specify access restrictions on shared files. The File Server Process should enforce these restrictions. However, this gives the process the same access control powers as the file system itself. If a remote user can trick the File Server into granting access, then the file system won't protect the file.

Moreover, how can the File Server enforce different access permissions on different users, when the users aren't actually present on the host computer? The first step is to identify the remote users who make file-sharing requests. The second step is to ensure that the File Server enforces access rights on file sharing.

Figure 10.27 illustrates one strategy to do this. First, Alice configures her system to recognize Bob as a user. The second step is to use Bob's access rights when he accesses the shared files. We can do this by running the File Server under Bob's local user name.

When Bob maps a network drive on Alice's computer, the process requires him to log into Alice's computer. The mapping process collects the user name and password before

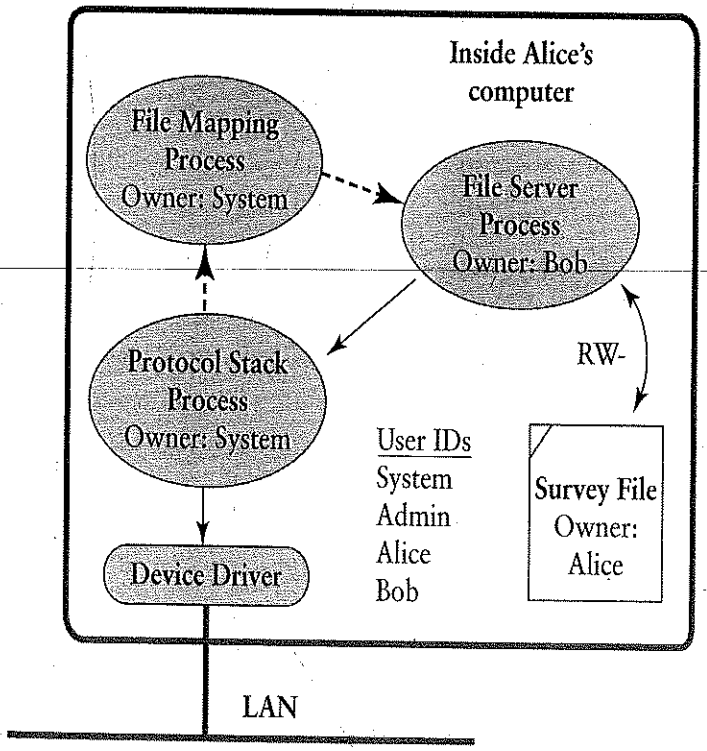


Figure 10.27

One way to delegate access to a file server.

it maps the drive. The user name must be one that exists on Alice's computer; she must set up a user identity for Bob. In some systems, the mapping process starts a separate File Server Process that uses Bob's user ID. This is shown by the dashed lines in Figure 10.27.

When Bob actually requests the survey file, the client puts his identity information into the request. The client then directs the request at Bob's File Server Process on Alice's computer. The process tries to retrieve the file. If Bob has permission to retrieve the file, then the process also has that permission, and it successfully retrieves the file. Otherwise, the access fails and the server returns an error message to Bob's client.

10.7 Resources

IMPORTANT TERMS INTRODUCED

ANALOG SIGNAL
APPLICATION PROTOCOL
BANDWIDTH
BOT
CIRCUIT SWITCHING
CLIENT
CLIENT/SERVER
COLLISION
DATA NETWORK
DATAGRAM
DELEGATION PROBLEM
DIGITAL SIGNAL
DISTRIBUTED PROCESSING
DOWNLINK
ETHERNET
FRAME
HOST
LINK
LINK PROTOCOL
MAC ADDRESS
MESSAGE SWITCHING
NETWORK HUB
NETWORK SWITCH
OPTICAL FIBER
PACKET SWITCHING
PEER-TO-PEER
PROTOCOL
PROTOCOL STACK
ROOTKIT
SEQUENCE DIAGRAM
SERVER
TIMEOUT
TRANSPORT PROTOCOL
UPLINK
WI-FI

ACRONYMS INTRODUCED

ACK—Acknowledgment
AM—Amplitude modulation
Cat 5—Category 5
Cat 5 E—Category 5, Enhanced
CB—Citizen's band
coax—Coaxial cable
CTS—Clear to Send
DDOS—Distributed denial of service attack
FCC—Federal Communications Commission
FM—Frequency modulation
Hz—Hertz
IP—Internet Protocol

LAN—Local area network
MAC—Media access control
modem—Modulator-demodulator
NAK—Negative acknowledgment
OSI—Open Systems Interconnect
PARC—Palo Alto Research Center
RJ-45—Registered Jack 45
RTS—Request to Send
TCP—Transmission control protocol
TV—Television

10.7.1 Review Questions

R1. For each of the six types of attacks, give an example of how the attack occurs on a network.

R2. Summarize and compare the three techniques for transmitting information on communications networks.

R3. Explain the differences between analog signals and digital signals, between parallel and serial links, and between synchronous and asynchronous communications.

R4. Explain the difference between baud and bandwidth.

R5. Explain how a typical acknowledgment protocol works, and why it provides reliable data transmission. How can the protocol produce duplicate packets?

R6. Briefly describe the different types of Ethernet LAN systems in use.

R7. Describe the fields in an Ethernet packet.

R8. Explain how collisions are handled on a wireless LAN. Compare it to how collisions are handled on a wired Ethernet LAN.

R9. Explain the relationship between layers of software in the network protocol stack and headers appearing in a packet.

R10. List the seven layers in the OSI protocol model in order from lowest to highest.

R11. Give examples of client/server and peer-to-peer network applications.

R12. Describe a typical approach to resource sharing on a network. What is the delegation problem?

10.7.2 Exercises

Note: Some exercises are marked with restrictions. Exercises marked "Small LANs only" should only be performed on a small LAN where the student has permission to perform the assignment, often a household LAN.

E1. (Small LANs only.) Identify a small, local network that you use. Ideally,

it should be a household network with at least one node. Determine the node's make and model. Visit the manufacturer's website and determine whether it provides switch, hub, and/or wireless interconnection between endpoints (switches and hubs are described in Section 10.4). Which is it?

E2. Using a host computer connected to a network, determine the following:

a. Manufacturer of the host's equipment

b. Host's MAC address

c. The vendor who assigned the MAC address. (Determine this from the "vendor unique address" in the MAC address.)

E3. (Small LANs only.) Identify a small, local network that you use. It should contain only a few hosts and no more than two network nodes. Do the following:

a. Identify a physical perimeter that provides some protection to that network.

b. Draw a diagram of the hosts, nodes, and links.

E4. Given the maximum size of an Ethernet packet (Figure 10.16), calculate the packet efficiency.

E5. Many early timesharing computers provided connections across LANs and other networks. Many of these computers accepted data typed at a keyboard one character at a time. Thus, many of the messages sent to these systems contained no more than 1 byte's worth of actual data, after all headers were removed. If we transmit the smallest possible Ethernet packet, and it only contains *1 byte* of data, what is the packet efficiency?