Chapter 15: Introduction to Data Communications

Chapter 15 Objectives

At the conclusion of this chapter, the reader will be able to:

- Draw a block diagram of a simple data communication system.
- Explain the difference between parallel and serial communications.
- Describe at least three different network topologies.
- Draw a block diagram of a modem, explaining the function of each part.
- Draw the waveform for an asynchronous character, given the data and parameters.
- Explain the operation of a UART.
- List and describe two methods used for error detection.
- List the basic communications pins of an RS232 interface, and give the purpose of each.
- Develop plans for troubleshooting data communications networks.
- Follow preventive procedures to maximize network reliability.

Computers are a basic ingredient of modern society. As communication tools, computers have redefined the telephone (smartphones are now common), and the Internet is a powerful broadcasting and advertising medium. The rapid acceptance of computers in business has even led to a formation of a discipline called *information technology*, or *IT* for short. Many companies have pooled their available talent into IT departments, which are collectively responsible for the proper operation of computers, networks, and communications equipment. The demand for IT professionals far exceeds the available supply!

A professional who wants to specialize in IT needs to understand the basics of data communications, networking fundamentals (Chapter 16), *plus* the essentials of any software (computer programs) that he or she will need to work with. A typical IT technician must have a working knowledge of many software applications, and must be able to communicate clearly with non-technical users.

Network equipment uses a staggering array of hardware and software. Hardware and software interact in an intimate fashion to make systems work. A detailed discussion of these technologies could certainly fill several volumes. Therefore, this chapter is intended as an *overview* of data communications techniques. Data communications techniques form the physical basis of all modern networks.

The technologies explored in this chapter are the basis of the networking technologies to be discussed in Chapter 16 from both hardware and software perspectives. You will find that familiarity with these key ideas will help you immensely when installing and maintaining both network hardware and softare.

15-1 Nature of Digital Data

Digital data refers to the form of information used by computers and other digital equipment. Digital systems use *binary numbers* to represent information. A binary number is a value expressed in base-2. Binary numbers are made up of *bits*, each of which can be either *1* or *0*.

The term "bit" is an abbreviation of the term *binary unit*, which is the simplest and smallest unit of information. A *byte* is a group of eight bits.

In an analog circuit, information is represented by a continuously variable voltage or current. A good example of an analog quantity is the voltage in a circuit. There are an infinite number of possible voltages. For example, we can measure potentials of 1 volt, 2 volts, 1.5 volts, 1.505 volts, and so on. There is no limit to how far this can go. The representation of digital information is quite different from that in analog circuits. A digital system can only represent a *finite* (countable) number of *states* or *conditions*. Consider the digital voltmeter of Figure 15-1.

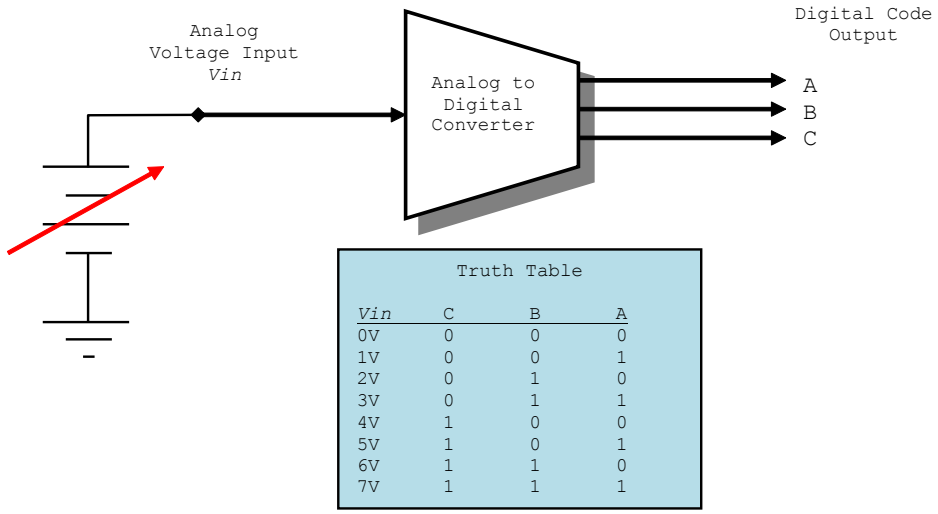

Figure 15-1: A Digital Voltmeter

The digital voltmeter in the figure can read voltages between 0V and 7V. The incoming voltage is converted into a 3-bit binary code by an *analog-to-digital converter*. Therefore, the voltage is represented by a 3-bit *binary code* at the output of the ADC. Each 3-bit binary reading from the ADC is called a *sample*. It is a snapshot of the input (analog) signal at some instant in time. A collection of samples can be put together to reproduce an analog signal.

Quantization

What will the above system register if the input is 2.2V? You can see that it is impossible to represent this exact value. Therefore, the digital system *rounds* its reading to the nearest reading, which would be *0 1 0*. We would say that the voltage information has been *quantized*. *Quantization* is the reduction of information with an infinite number of possible values (like a voltage) into something with a *finite* (countable) number of values (like the digital output of the ADC in Figure 15-1).

A digital system can never exactly represent analog information, but it can get close. The more bits that are used, the better the digital representation becomes. This is why 16-bit sound recordings (such as those on CDs or high-quality PC sound cards) sound much better than 8-bit recordings, and 24 bit recordings (such as those made with studio digital audio mastering equipment, and some high-end computer sound cards) sound even better.

The number of possible states in a digital sample can be easily calculated:

(15-1) $MOD \mid 2^N$

where *MOD* is the *modulus* or number possible counting states, and *N* is the number of bits in the digital code.

Example 15-1

How many different analog voltages can be represented in a computer *WAV* (digital audio) file if the samples are (a) 8 bits and (b) 16 bits?

Solution

Since these are binary numbers, we can treat them just like digital counters. Equation 15-1 can be directly applied in each case:

a) $MOD \mid 2^N \mid 2^8 \mid \underline{\underline{256}}$ different voltages.

b) $MOD \mid 2^N \mid 2^{16} \mid \underline{\underline{65536}}$ different voltages.

The 16-bit system represents a much better picture of the original sound, at the cost of doubling the number of bits required for each sample.

A Complete Digital Communication System

Figure 15-2 shows a complete digital communication system. The system in the figure begins with a *digital data source*. This data source could be any kind of information. It could be a temperature reading at a remote location in a manufacturing plant, a count of the number of cars passing by a point in a roadway, a stream of digital audio, or even the typed characters representing an electronic mail (e-mail) message.

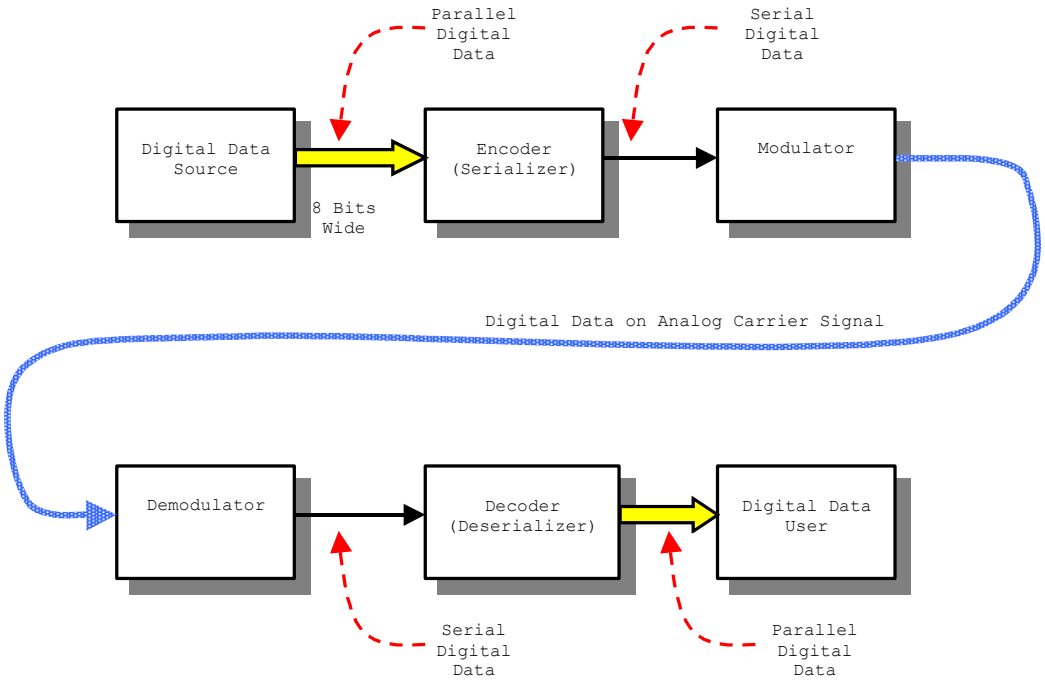

Figure 15-2: A Complete Digital Communication System

Digital data tends to be produced in *parallel* form. *Parallel data* is data where all the bits are sent at the same time. Computers work and communicate using eight bits, or a multiple of that number. It is common for computer data to be 8, 16, 32, 64, or even 128 bits wide. For a computer that communicates in 8-bit chunks, this poses a problem: At least eight wires are needed to carry the information, one for each data bit. For 16-bit and wider machines, the problem gets even worse. How can this problem be solved?

Figure 15-2 shows how it can be done. The *encoder* or *serializer* is responsible for transmitting the digital bits one at a time. We would say that the output of the serializer is *time division multiplexed digital data*, or *serial data*. Remember multiplexing? Multiplexing is the sending of more than one piece of information over a communications channel. FM and TV used *frequency division multiplexing* to carry different parts of their signals. Because each digital bit is assigned a different point in *time*, the serial output signal is considered to be time-division-multiplexed.

Parallel Transmission

Having one wire for each data bit is like having one lane for each car in Figure 15-3. This is called *parallel* transmission. For short distances (such as carrying a signal a few feet to a printer), this type of transmission is useful. Within a computer parallel transmission is used to carry address and data signals between the CPU, memory, and peripheral devices. It is simple and easy to manage.

Serial Transmission

However, it is possible to get all the cars to merge into a single lane. Although this will be slower (as only one car gets to use the road at a time), it is much cheaper to build, especially over a long distance. Notice that the cars line up in a particular order. This is important, since they will need to be assigned the proper lane again when they reach their destination. This is *serial* transmission. In serial transmission, one bit is sent at a time. Given equal channel conditions, serial transmission is always slower than parallel transmission -- but it much less expensive. The Universal Serial Bus (USB) is a good example of a serial data transmission standard.

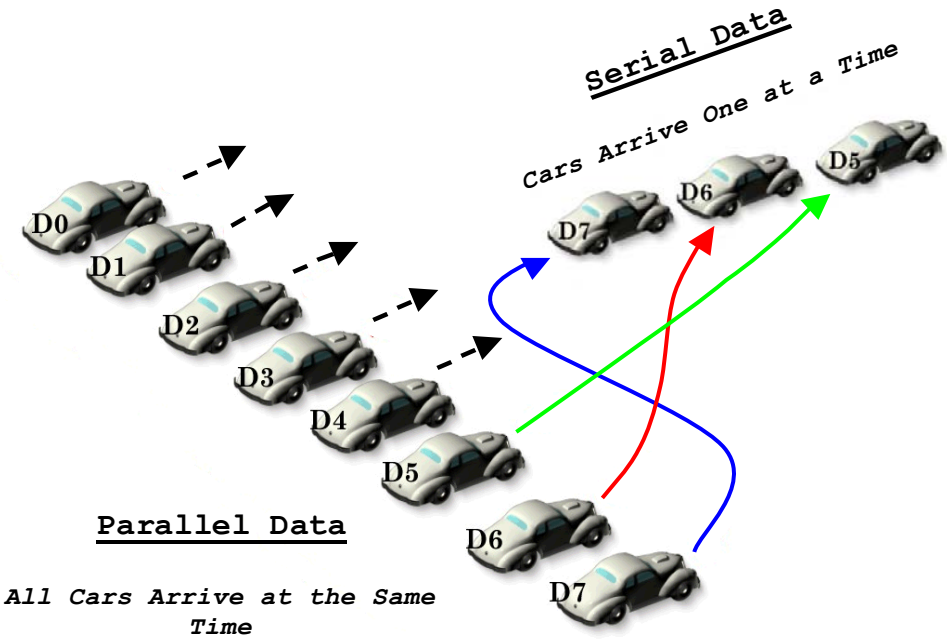

Figure 15-3: Parallel and Serial Traffic Flow

Modulation and Demodulation

The serial data signal of Figure 15-2 is almost ready for transmission. In order for the signal to be carried a long distance, it must be placed upon a carrier of some type. *In other words, the digital signal must be converted into an analog form*.

The analog carrier can be many things. In a high-speed communication system, the carrier might be a beam of laser light in a fiber optic "pipe" (discussed in Chapter 18). For communication over a phone line, the carrier may be a sine wave in the range of 300 Hz to 3000 Hz (the frequency range available over a standard dial-up telephone line). The carrier could also be a radio wave, using any of the standard modulation techniques (AM, FM, or PM). Radio wave carriers are often used where it is inconvenient or impossible to run wires, such as between two buildings on a college campus, or between an earth station and an orbiting space vehicle.

At the receiving end, the process is simply reversed. The analog carrier is received, and the *demodulator* recovers the serial digital data waveform. A *decoder* or *deserializer* reassembles the parallel data, which is then consumed by the data user (usually another computer).

Note that for short distances (a few meters or less), a digital signal can be sent directly without the use of modulation. The secret behind this is simple: For short-range transmission, the wiring must be treated as an RF transmission line (see Chapter 11) to avoid distortion of the digital waveforms. This why USB and other similar cables may only be a meter or so in length.

Measuring Digital Information Flow

Since the units of digital information are binary units or bits, the rate of digital information flow are *bits per second*, or bps. The higher the bps flow, the more information that can be transferred per unit time. Figure 15-4 shows how this works. This is a serial data waveform, with one time slot or *bit cell* assigned to each digital bit. For illustration purposes, the pattern "1010..." is being sent. This pattern is often used to test digital communications devices, since it can be easily generated by the square-wave output of a signal generator.

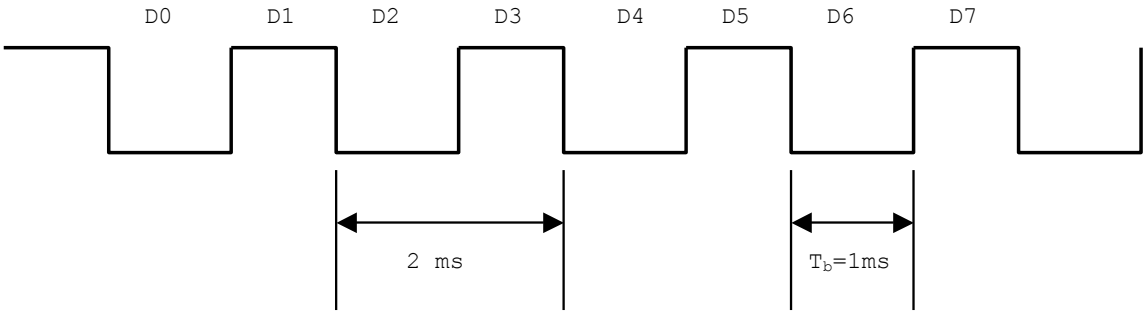

Figure 15-4: A Serial Data Waveform

A *bit cell* is simply the time interval assigned for the transmission of a bit. By knowing the time of each bit, we can calculate the rate of information flow:

(15-2) $$DR = \frac{1}{T_b}$$

where T_b is the time interval for one bit, and DR is the rate of information flow in bits per second.

Example 15-2

In Figure 15-4, compute the data rate in bps, and the signal generator frequency that will simulate the given data pattern (alternating 1s and 0s).

Solution
Equation 15-2 calculates data rate:

$$bps = \frac{1}{T_b} = \frac{1}{1ms} = \underline{\underline{1000\ bps}}$$

By examining the waveform, we see that it is merely a square wave with a period of 2ms. Therefore, the frequency of the waveform is:

$$f = \frac{1}{T} = \frac{1}{2ms} = \underline{\underline{500Hz}}$$

To simulate a 1000 bps serial data stream, a 500 Hz square wave should be used. Are you surprised that the frequency is only 500 Hz (and not 1 kHz)? If so, look carefully at the waveform. For each cycle of the square wave, there is both a logic 1 and logic 0. Each cycle therefore represents *two* bits.

Modes of Communication

The data communication system of Figure 15-2 only has the capability to talk in one direction. For many applications this is sufficient; we refer to this as *simplex* communication. A system that communicates in simplex mode can only work in one direction. Broadcasting is a good example of simplex communication; you can hear the DJ on your portable radio, but the DJ can't hear you (at least not through your radio!)

In order to attain two-way communications (which is needed for most applications), we need *two* sets of the blocks of Figure 15-2. The system will now be capable of sending information in both directions. Some two-way systems can only communicate in one direction at a time. For example, a hand-held walkie-talkie is designed to transmit only when the push-to-talk button is pressed, and during that time, it can't receive. A system that can communicate in both directions, but only one at a time is called a *half-duplex* system.

In some applications, it is necessary to have unrestricted two-way communications. A system that can communicate in both directions at the same time is called a *full-duplex* system. The telephone is an example of a full-duplex device; anyone who has argued over the phone can verify that! An example of an application where full duplex communication is necessary is the control of a robot in a manufacturing plant. An industrial robot works in a specialized environment called a *cell*, and receives commands from a nearby computer. The robot also talks back to that computer to return status information (for example, whether or

not an commanded operation executed successfully). The controlling computer is responsible for monitoring the safety of the cell containing the robot. If an unsafe condition is detected (for example, human presence in a dangerous area), the computer must immediately stop the robot. The robot must therefore always be able to "listen" for such a command, even if it is in the process of giving status information back to the computer ("talking.")

Section Checkpoint

15-1 What is meant by the term "digital data?"
15-2 What is a binary number?
15-3 Explain the difference between a *bit* and a *byte*.
15-4 How is the representation of information different for digital and analog circuits? Give an example.
15-5 What is a *sample*?
15-6 What is meant by the term "quantization?"
15-7 What is the difference between serial and parallel data?
15-8 Which is faster, serial or parallel data? Why?
15-9 What type of multiplexing is used to create serial data?
15-10 What are the units of digital information flow?
15-11 Explain the difference between simplex, half-duplex, and full-duplex communications.

15-2 Network Topologies

A *network* is defined as an organized system for computer communication. You may have already worked on a "networked computer," which usually means a PC connected to a local area network (LAN). There are many possible ways of connecting computers together. The physical layout of a computer network is called its *topology*. Networks can be classified in several ways; the topology (how the network is built) is one of the most fundamental.

Point-to-Point Network

Figure 15-5 shows the simplest possible network topology, a *point-to-point network*:

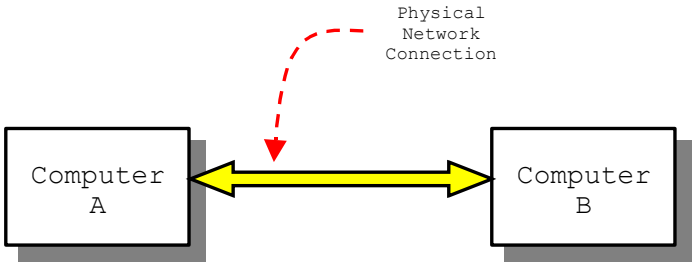

Figure 15-5: A Point to Point Network

A good example of a point-to-point connection is a dedicated line connecting two large mainframe computers, such as between two branches of a bank. A point-to-point network is sometimes used as the foundation for a *peer-to-peer network*, because the two units (marked A and B in the figure) may considered to be *equals*. A peer-to-peer network can be formed by connecting two or more computers with equivalent roles. For example, you may have connected two computers together in order to share files through a LAN.

However, just because a network is physically wired point-to-point doesn't mean that the computers are equal in function (peer-to-peer). For example, when you use a dial-up telephone line to connect to an Internet service provider (ISP), you are creating a virtual point-to-point circuit between your computer and the ISP computer. This is *not* a peer-to-peer relationship; the ISP is providing a service, and your computer is acting as a client (user of the service).

A point-to-point network is often used whenever two computers need to share resources such as printers, disks, and other storage devices. The primary advantage of this setup is its simplicity; we would say that the *protocol* requirements are very minimal. A *protocol* is a set of rules for communication. For example, a common part of the protocol for speaking in most human languages is to not speak while the other person is talking, to avoid interrupting them. The rules in a computer protocol control communications; they specify things such as when a unit may send a message onto a network, when a unit must "listen," and what a computer should do in case it detects an error in a message. The protocol's rules also determine how to form a message before transmitting it onto the network.

Although the point to point network is very simple and easy to implement, it suffers from two shortcomings. First, only two computers are allowed on the network. While this might be useful from a security standpoint, it severely limits the scope of communications. Second, if there are only two computers using the communications line, the cost of providing communications to each computer might be relatively high when compared to other network solutions. For example, suppose that the communications line in Figure 15-5 is a leased T1 line from the telephone company. This line provides 1.544 Mbps (mega-bit per second) capacity. A T1 line can cost over one thousand dollars per month to lease. If the volume of communications between the computers is sufficient, then the capacity of the line will be utilized fully, and the company will be getting its money's worth. It's more likely, though, that the line will be *underutilized* with only two computers on it. *A point to point configuration usually does not fully utilize the communications line.*

Note that a peer-to-peer connection of two (or more) personal computers is not exactly a *true* point-to-point connection. Peer-to-peer connection of PCs uses networking protocols that allow multiple units to share the same network wiring as a distributed *bus*, which will be discussed shortly.

Multidrop Network

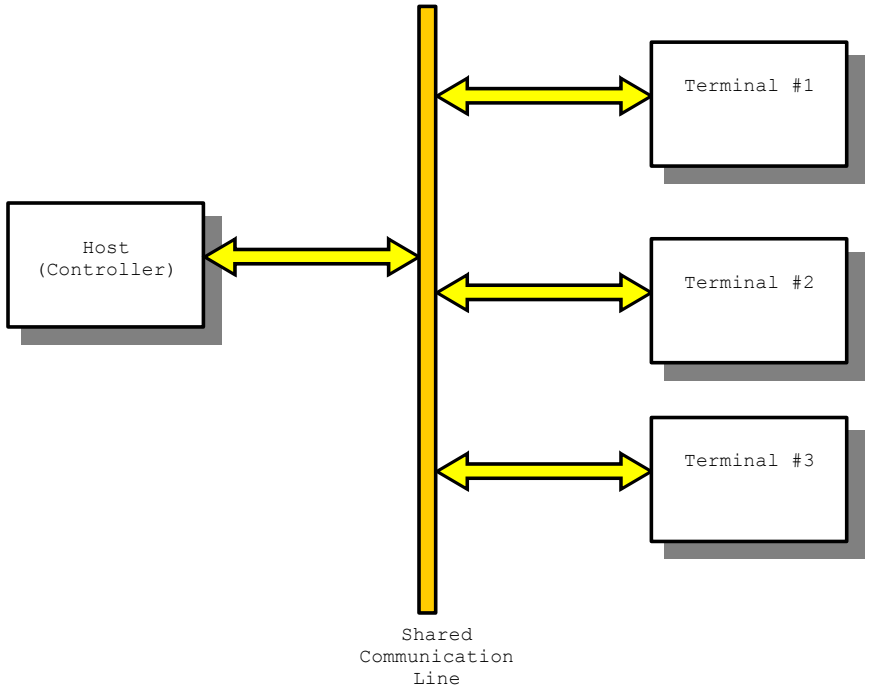

Figure 15-6: A Multidrop Network

An improvement over the point-to-point network is the multidrop of Figure 15-6. There are three differences between a point-to-point and multidrop network. First, more computers are allowed onto the same communications line. It now resembles a party-line telephone circuit, where several homes share the same physical telephone line. Second, because there are more units on the line, the rules of communication (protocol) must become more complex. The protocol must now account for more than two computers, and all of the computers must be able to communicate without receiving interference from other stations. The third change allows this. The third difference is the designation of one of the units as "boss" over all network communications. This "boss" computer is often referred to as a *host* computer. The host computer controls all communications on the multidrop network by *polling* or *selecting* each individual unit one at a time, in turn.

On a multidrop network, the *slave* units (having addresses 1, 2, and 3 above) can't transmit without permission from the host. The host *polls* each unit, one at a time, to check whether that unit has a message to send. Thus, the host will continuously seek out units 1, 2, 3, 1, 2, 3…in that order. All units except the one given permission by the host remain silent, so this arrangment prevents the collision of data from two different units. Note that data is exchanged only between the host and a slave; two slave units cannot directly communicate at all, for the host gives permission talk to only one unit at a time.

The host may have a pending message for one or more of the slave units. In that situation, the host simply sends a *select* message to that unit (notice again that each unit has a unique numerical address). Only the unit that has been selected pays attention to the message; the others ignore it.

The multidrop has as its primary advantage better utilization of the communications line. Many computers can now share the resource, which reduces the per-unit cost. However, from this brief discussion,

you can see that the rules of communication (protocol) have become much more complicated than before. Also, there is a hidden problem: What happens if the host computer breaks? Since this one computer has the responsibility of directing all the network activity, the network essentially stops working if the host fails! There is also the problem of a slave unit getting "stuck" in transmit mode. This occasionally happens. If any one unit stays in transmit, the entire network is down! To get a unit "unstuck," a technician normally must manually reset it.

Star Network

The star network of Figure 15-7 is a modification that is intended to improve reliability. Can you see how that is so?

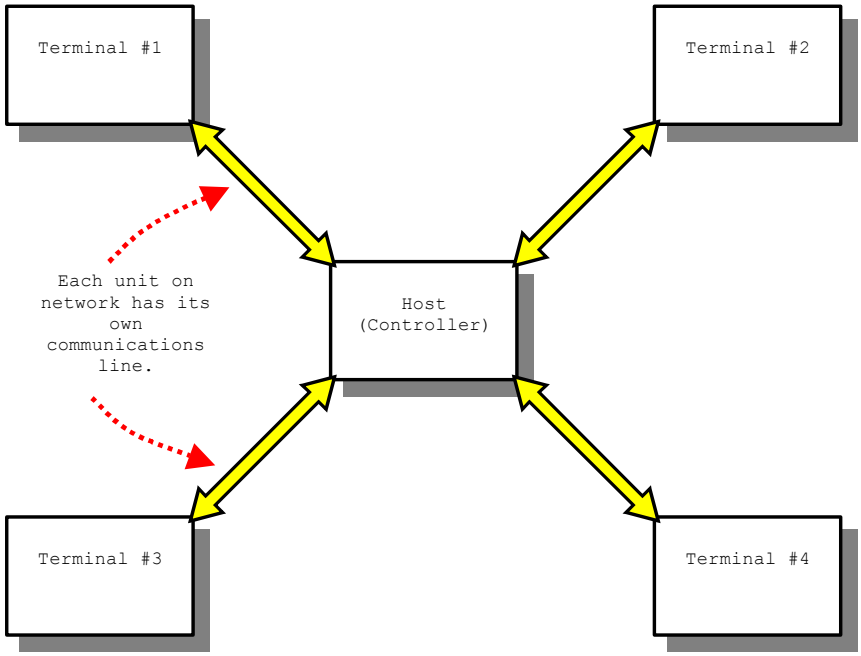

Figure 15-7: A star network

If you noticed that there is now one separate communications line for each unit on the network, you're right! On a multidrop network, any one unit can foul things up by hanging in transmit mode. In some applications, that is an unacceptable condition (such as in the control of military apparatus), so the addition of one line per unit solves that portion of the reliability problem. There is still a centralized host computer that serves all the slave units, so that if this computer fails, the network will stop functioning. However, it is possible to distribute the computing burden of the host among several centralized computers, so that if one of the central computers fails, most of the network will continue to function.

The primary advantage of the star network is its reliability. One renegade slave unit can't bring this network down. However, the cost of providing communications to each unit may still be high, as it was in the point-to-point network.

Ring Network

The *ring* network of Figure 15-8 represents a *distributed* approach to the control of communications over a network. By *distributed* we mean that the responsibility for controlling the operation of the network is no longer assigned to a centralized "boss" (host) computer like it was before. All the computers connected on the network equally share the control responsibility. The units on a ring network can be individual personal computers.

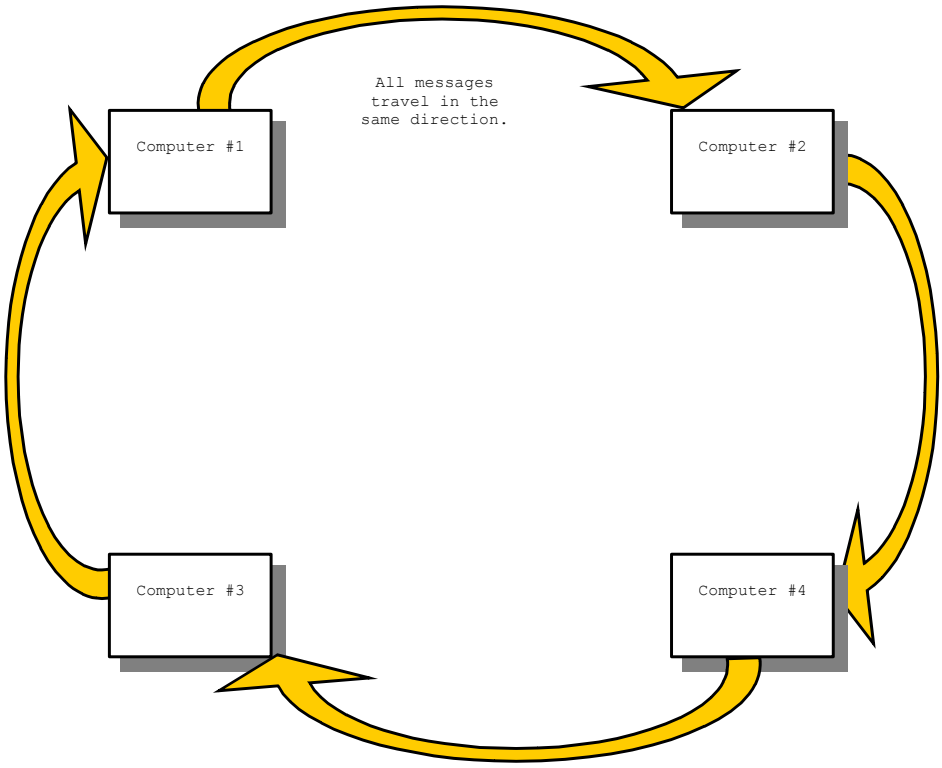

Figure 15-8: A ring network

The ring network gets its name from the overall shape of the communications path. It looks very much like a giant series circuit. Information always flows in one direction, from one unit to the next, around and around the ring. There is no "boss" on the ring; all of the units are self-responsible about when they can transmit. This is achieved through the use of an electronic "permission slip" called the *token*.

The *token* is a special data pattern (electronic object) that circulates the ring when no one is sending any messages. *The presence of the token on the ring indicates that the network is idle.* Any unit can send a message to any other unit as long as it can capture the token. For example, suppose that unit 1 needs to send a message to unit 4. The sequence of events would look like this:

1. Unit 1 waits until it receives the circulating token. Upon receiving the token, unit 1 removes the token from circulation, and transmits its message (which has address 4 marked as the recipient within the message body.)
2. Unit 2 sees the message from unit 1, and since the message is not a token, unit 2 knows it cannot originate any new messages at this time. Therefore, unit 2 repeats the message, which then passes on to unit 3.
3. Unit 3 treats the message in the same way, repeating it for unit 4.
4. Unit 4 recognizes the frame message as having "its" address (4), and proceeds to decipher the message content. Unit 4 also checks the message for errors. Unit 4 then *retransmits* the message back onto the ring, with the "received box" checked, and the "error status box" checked. (These two items are actually digital data bits that are reserved as part of the network message format.)
5. Unit 1 receives the message it just sent, and (you guessed it), examines the "received" and "error status" bits. If the message was received OK, unit 1 *releases the token* so that other units may communicate. If the message was not received correctly, unit 1 can retransmit it again.

The protocol used to operate a ring network is a little more complex than either of the previous networks, but it is very effective in operation. The primary disadvantage of this topology is reliability; if the network is broken at any point, all communications will cease, since a complete circle is needed for the units to talk back and forth.

The IBM token ring network (a local area network used by PCs in the 1980s and 1990s, and now only a historical curiosity) is actually a modified ring configuration. The network consists of a *main* and a *back* ring. If a unit detects a communications fault, it will attempt to connect through the redundant back ring. The failure of a single wire does not bother an IBM token ring network. Figure 15-9 shows how this is so.

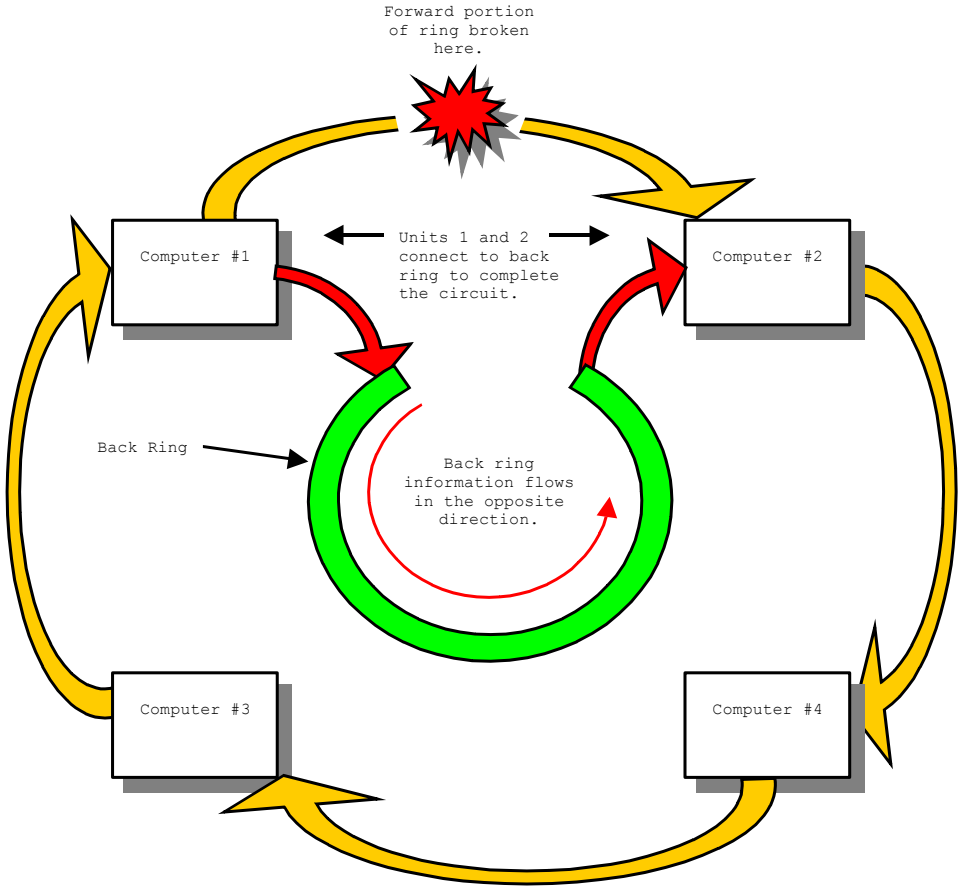

Figure 15-9: Operation of the Back Ring

In the figure, a wire has broken on the *main ring* between units 1 and 2. Momentarily, network communications ceases due to the broken connection. At this point, the units on the network begin a programmed strategy to get back on line. Unit 2 senses a loss of DC continuity on its input pin, and connects to the back ring. Unit 1 detects a similar problem, and connects its output to the back ring. Notice that the back ring contains short circuits for all of the units, except 1, so that the messages from 1 can now travel *counter-clockwise* on the back ring to unit 2. *The network has "healed" itself, at least temporarily!*

Bus Network

The *bus* network of Figure 15-10 is used in LANs. *Ethernet* is a bus network. The bus network is well suited to a distributed computing environment, like the ring network. In fact, the network above is much like the multidrop setup we discussed earlier, with one major difference: There is no "boss" or host computer for controlling the network communications. Like on the ring network, the individual units take full responsibility for their own activity, and use a specialized protocol to govern their communications.

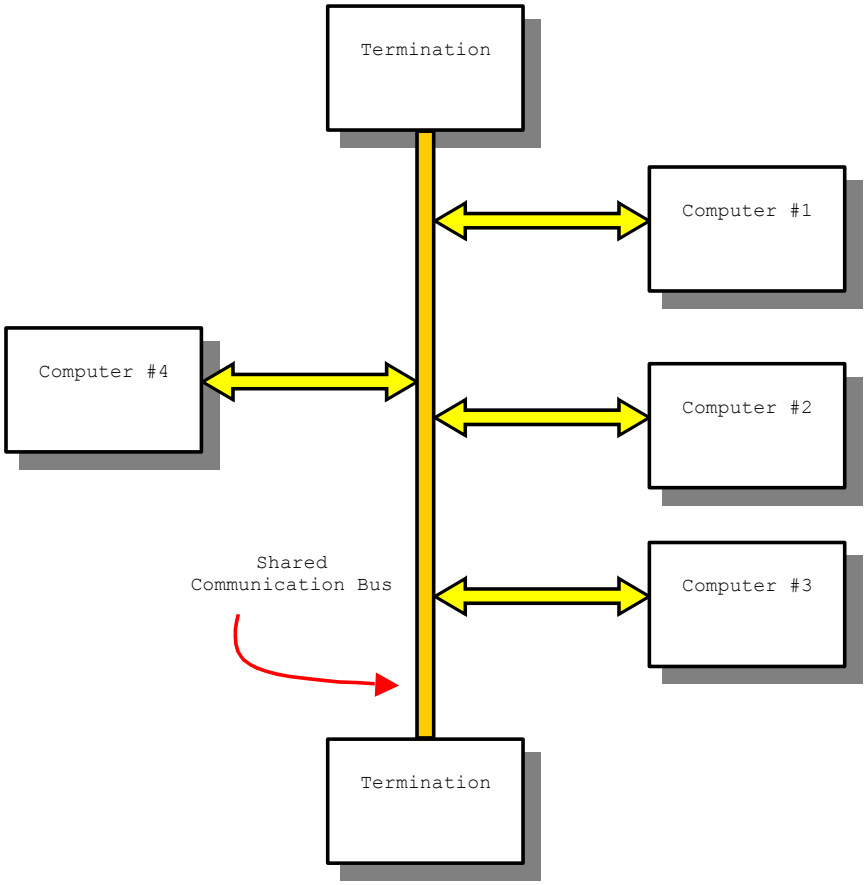

Figure 15-10: A Bus Network

The nature of communications on a bus network is very similar to that at a dinner table. At the table, only one person normally speaks at a time, and usually addresses one other person by name (though a person can certainly "broadcast" a message for all at the table to hear). The method used to achieve this on a bus network is called CSMA/CD, which stands for *carrier sense multiple access with collision detection*. The operation of CSMA/CD can best be summarized as "listen before you talk." Each unit on a bus network actively listens to communications in progress, and waits until no carrier is sensed (no one is transmitting) before transmitting a message onto the network. Since all the computers are on the same bus, they can all hear the outgoing message, but only the computer with the proper network address will respond; the others will remain silent. The receiving computer responds back to acknowledge the message, and the network operation then resumes.

It is possible that *two* or more units may attempt to send at the same time. When this happens, we say that a *collision* has occurred on the network. The transmitting units can detect a collision because they can listen while they are transmitting. The signals from the competing computer partially disrupt the outgoing signal, which is immediately detected by both of the competing computers. Both computers stop sending, and each picks a random number. The smallest random number wins, and its holder gets to communicate first.

The bus network is a very economical solution for LAN applications - in fact, Ethernet is actually a bus network. Its primary limitation is the number of computers that a single "backbone" can support. For many applications, 25 computers on a single backbone can become very slow, because the rate of collisions increases much faster than the number of computers on the network. For very large networks, groups of bus backbone networks are connected through *bridges*, *switches* or *routers*, depending upon the communication need.

A *bridge* is a device that connects two networks together, and allows them to act as one. It can bridge incompatible networks (such as a Token-ring and Ethernet) as well. A *switch* is a device that intelligently routes messages between computers on a network. It examines the data link layer address (or MAC address) of the message and sends it out only to the computer with a matching address. This reduces congestion on the bus and increases security. A *router* operates at a higher level than a bridge or switch. A router not only connects two or more networks, but actively controls how the messages will pass between them. A router can