PART 3

Data Link Layer

Objectives

The data link layer transforms the physical layer, a raw transmission facility, to a link responsible for node-to-node (hop-to-hop) communication. Specific responsibilities of the data link layer include *framing, addressing, flow control, error control,* and *media access control.* The data link layer divides the stream of bits received from the network layer into manageable data units called frames. The data link layer adds a header to the frame to define the addresses of the sender and receiver of the frame. If the rate at which the data are absorbed by the receiver is less than the rate at which data are produced in the sender, the data link layer imposes a flow control mechanism to avoid overwhelming the receiver. The data link layer also adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged, duplicate, or lost frames. When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

In Part 3 of the book, we first discuss services provided by the data link layer. We then discuss the implementation of these services in local area networks (LANs). Finally we discuss how wide area networks (WANs) use these services.

Part 3 of the book is devoted to the data link layer and the services provided by this layer.

Chapters

This part consists of nine chapters: Chapters 10 to 18.

Chapter 10

Chapter 10 discusses error detection and correction. Although the quality of devices and media have been improved during the last decade, we still need to check for errors and correct them in most applications.

Chapter 11

Chapter 11 is named data link control, which involves flow and error control. It discusses some protocols that are designed to handle the services required from the data link layer in relation to the network layer.

Chapter 12

Chapter 12 is devoted to access control, the duties of the data link layer that are related to the use of the physical layer.

Chapter 13

This chapter introduces wired local area networks. A wired LAN, viewed as a link, is mostly involved in the physical and data link layers. We have devoted the chapter to the discussion of Ethernet and its evolution, a dominant technology today.

Chapter 14

This chapter introduces wireless local area networks. The wireless LAN is a growing technology in the Internet. We devote one chapter to this topic.

Chapter 15

After discussing wired and wireless LANs, we show how they can be connected together using connecting devices.

Chapter 16

This is the first chapter on wide area networks (WANs). We start with wireless WANs and then move on to satellite networks and mobile telephone networks.

Chapter 17

To demonstrate the operation of a high-speed wide area network that can be used as a backbone for other WANs or for the Internet, we have chosen to devote all of Chapter 17 to SONET, a wide area network that uses fiber-optic technology.

Chapter 18

This chapter concludes our discussion on wide area networks. Two switched WANs, Frame Relay and ATM, are discussed here.

Down

CHAPTER 10

Error Detection and Correction

Networks must be able to transfer data from one device to another with acceptable accuracy. For most applications, a system must guarantee that the data received are identical to the data transmitted. Any time data are transmitted from one node to the next, they can become corrupted in passage. Many factors can alter one or more bits of a message. Some applications require a mechanism for detecting and correcting **errors.**

Data can be corrupted during transmission.
Some applications require that errors be detected and corrected.

Some applications can tolerate a small level of error. For example, random errors in audio or video transmissions may be tolerable, but when we transfer text, we expect a very high level of accuracy.

10.1 INTRODUCTION

Let us first discuss some issues related, directly or indirectly, to error detection and correcion.

Types of Errors

Whenever bits flow from one point to another, they are subject to unpredictable changes because of **interference.** This interference can change the shape of the signal. In a single-bit error, a 0 is changed to a 1 or a 1 to a 0. In a burst error, multiple bits are changed. For example, a 1/100 s burst of impulse noise on a transmission with a data rate of 1200 bps might change all or some of the12 bits of information.

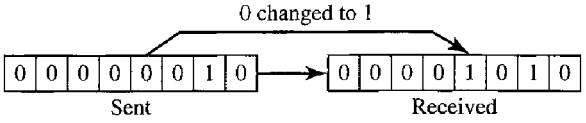
Single-Bit Error

The term ***single-bit error*** means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.

In a single-bit error, only 1 bit in the data unit has changed.

Figure 10.1 shows the effect of a single-bit error on a data unit. To understand the impact of the change, imagine that each group of 8 bits is an ASCII character with a 0 bit added to the left. In Figure 10.1, 00000010 (ASCII *STX*) was sent, meaning *start of text,* but 00001010 (ASCII *LF*) was received, meaning *line feed.* (For more information about ASCII code, see Appendix A.)

Figure 10.1 *Single-bit error*



Single-bit errors are the least likely type of error in serial data transmission. To understand why, imagine data sent at 1 Mbps. This means that each bit lasts only 1/1,000,000 s, or 1 μs. For a single-bit error to occur, the noise must have a duration of only 1 μs, which is very rare; noise normally lasts much longer than this.

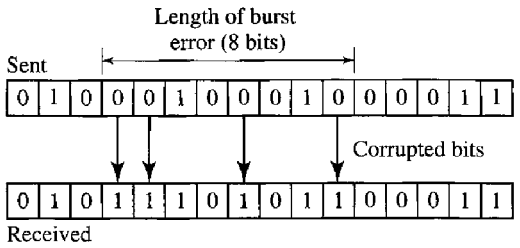
Burst Error

The term ***burst error*** means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

A burst error means that 2 or more bits in the data unit have changed.

Figure 10.2 shows the effect of a burst error on a data unit. In this case, 0100010001000011 was sent, but 0101110101100011 was received. Note that a burst error does not necessarily mean that the errors occur in consecutive bits. The length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

Figure 10.2 *Burst error of length 8*



A burst error is more likely to occur than a single-bit error. The duration of noise is normally longer than the duration of 1 bit, which means that when noise affects data, it affects a set of bits. The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at 1 kbps, a noise of 1/100 s can affect 10 bits; if we are sending data at 1 Mbps, the same noise can affect 10,000 bits.

Redundancy

The central concept in detecting or correcting errors is **redundancy.** To be able to detect or correct errors, we need to send some extra bits with our data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.

To detect or correct errors, we need to send extra (redundant) bits with data.

Detection Versus Correction

The correction of errors is more difficult than the detection. In **error detection,** we are looking only to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of errors. A single-bit error is the same for us as a burst error.

In **error correction,** we need to know the exact number of bits that are corrupted and more importantly, their location in the message. The number of the errors and the size of the message are important factors. If we need to correct one single error in an 8-bit data unit, we need to consider eight possible error locations; if we need to correct two errors in a data unit of the same size, we need to consider 28 possibilities. You can imagine the receiver's difficulty in finding 10 errors in a data unit of 1000 bits.

Forward Error Correction Versus Retransmission

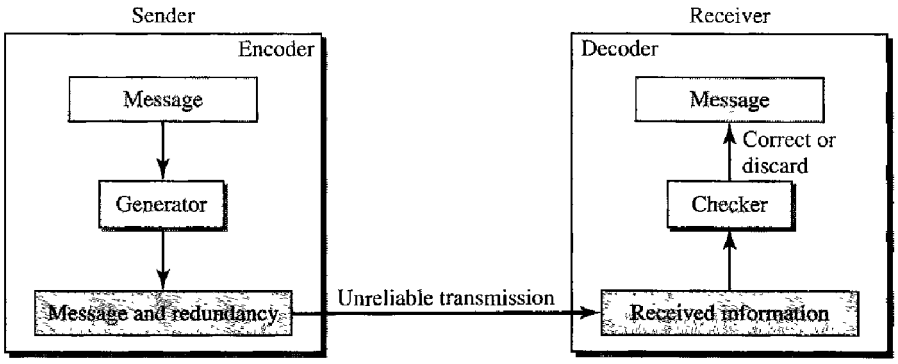
There are two main methods of error correction. **Forward error correction** is the process in which the receiver tries to guess the message by using redundant bits. This is possible, as we see later, if the number of errors is small. Correction by **retransmission** is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-free (usually, not all errors can be detected).

Coding

Redundancy is achieved through various coding schemes. The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits. The receiver checks the relationships between the two sets of bits to detect or correct the errors. The ratio of redundant bits to the data bits and the robustness of the process are important factors in any coding scheme. Figure 10.3 shows the general idea of coding.

We can divide coding schemes into two broad categories: **block coding** and **convolution coding.** In this book, we concentrate on block coding; convolution coding is more complex and beyond the scope of this book.

Figure 10.3 *The structure of encoder and decoder*



In this book, we concentrate on block codes; we leave convolution codes to advanced texts.

Modular Arithmetic

Before we finish this section, let us briefly discuss a concept basic to computer science in general and to error detection and correction in particular: modular arithmetic. Our intent here is not to delve deeply into the mathematics of this topic; we present just enough information to provide a background to materials discussed in this chapter.

In **modular arithmetic,** we use only a limited range of integers. We define an upper limit, called a **modulus** N. We then use only the integers 0 to $N - 1$, inclusive. This is modulo-N arithmetic. For example, if the modulus is 12, we use only the integers 0 to 11, inclusive. An example of modulo arithmetic is our clock system. It is based on modulo-12 arithmetic, substituting the number 12 for 0. In a modulo-N system, if a number is greater than N, it is divided by N and the remainder is the result. If it is negative, as many Ns as needed are added to make it positive. Consider our clock system again. If we start a job at 11 A.M. and the job takes 5 h, we can say that the job is to be finished at 16:00 if we are in the military, or we can say that it will be finished at 4 P.M. (the remainder of 16/12 is 4).

In modulo-N arithmetic, we use only the integers in the range 0 to $N - 1$, inclusive.

Addition and subtraction in modulo arithmetic are simple. There is no carry when you add two digits in a column. There is no carry when you subtract one digit from another in a column.

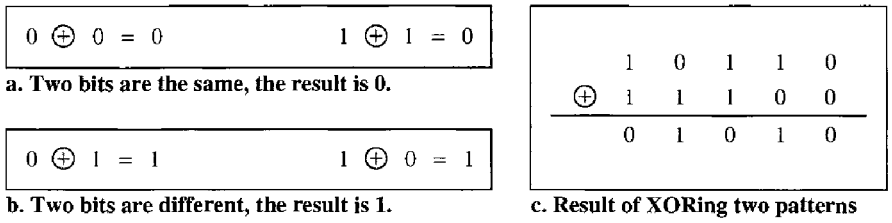
Modulo-2 Arithmetic

Of particular interest is modulo-2 arithmetic. In this arithmetic, the modulus N is 2. We can use only 0 and 1. Operations in this arithmetic are very simple. The following shows how we can add or subtract 2 bits.

Adding:	$0 + 0 = 0$	$0 + 1 = 1$	$1 + 0 = 1$	$1 + 1 = 0$
Subtracting:	$0 - 0 = 0$	$0 - 1 = 1$	$1 - 0 = 1$	$1 - 1 = 0$

Notice particularly that addition and subtraction give the same results. In this arithmetic we use the XOR (exclusive OR) operation for both addition and subtraction. The result of an XOR operation is 0 if two bits are the same; the result is 1 if two bits are different. Figure 10.4 shows this operation.

Figure 10.4 *XORing of two single bits or two words*



a. Two bits are the same, the result is 0.

b. Two bits are different, the result is 1.

c. Result of XORing two patterns

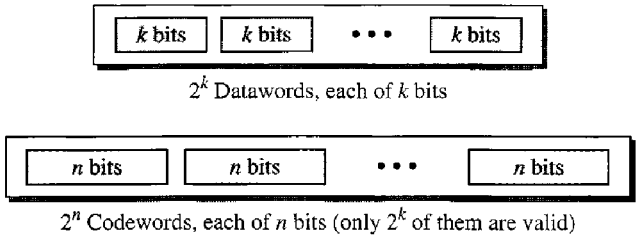
Other Modulo Arithmetic

We also use, modulo-N arithmetic through the book. The principle is the same; we use numbers between 0 and $N - 1$. If the modulus is not 2, addition and subtraction are distinct. If we get a negative result, we add enough multiples of N to make it positive.

10.2 BLOCK CODING

In block coding, we divide our message into blocks, each of k bits, called **datawords.** We add r redundant bits to each block to make the length $n = k + r$. The resulting n-bit blocks are called **codewords.** How the extra r bits is chosen or calculated is something we will discuss later. For the moment, it is important to know that we have a set of datawords, each of size k, and a set of codewords, each of size of n. With k bits, we can create a combination of 2^k datawords; with n bits, we can create a combination of 2^n codewords. Since $n > k$, the number of possible codewords is larger than the number of possible datawords. The block coding process is one-to-one; the same dataword is always encoded as the same codeword. This means that we have $2^n - 2^k$ codewords that are not used. We call these codewords invalid or illegal. Figure 10.5 shows the situation.

Figure 10.5 *Datawords and codewords in block coding*



Example 10.1

The 4B/5B block coding discussed in Chapter 4 is a good example of this type of coding. In this coding scheme, $k = 4$ and $n = 5$. As we saw, we have $2^k = 16$ datawords and $2^n = 32$ codewords. We saw that 16 out of 32 codewords are used for message transfer and the rest are either used for other purposes or unused.

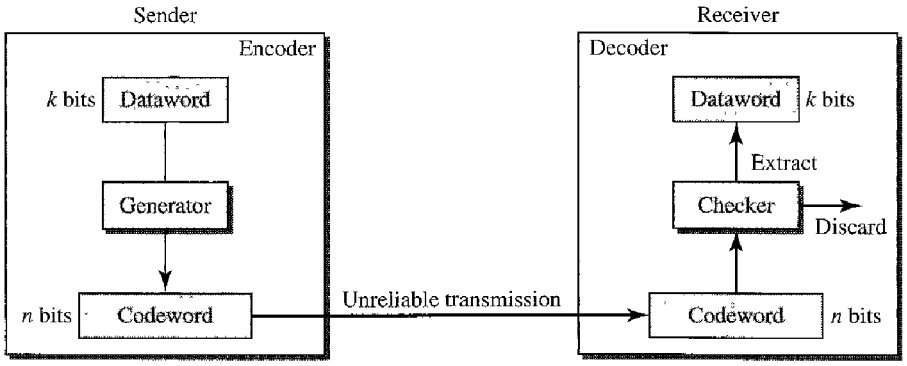
Error Detection

How can errors be detected by using block coding? If the following two conditions are met, the receiver can detect a change in the original codeword.

1. The receiver has (or can find) a list of valid codewords.
2. The original codeword has changed to an invalid one.

Figure 10.6 shows the role of block coding in error detection.

Figure 10.6 *Process of error detection in block coding*



The sender creates codewords out of datawords by using a generator that applies the rules and procedures of encoding (discussed later). Each codeword sent to the receiver may change during transmission. If the received codeword is the same as one of the valid codewords, the word is accepted; the corresponding dataword is extracted for use. If the received codeword is not valid, it is discarded. However, if the codeword is corrupted during transmission but the received word still matches a valid codeword, the error remains undetected. This type of coding can detect only single errors. Two or more errors may remain undetected.

Example 10.2

Let us assume that $k = 2$ and $n = 3$. Table 10.1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword.

Table 10.1 *A code for error detection (Example 10.2)*

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

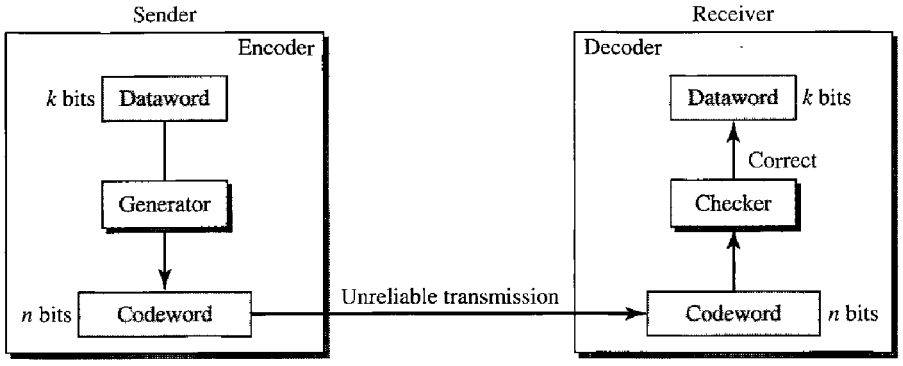
1. The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.
2. The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted). This is not a valid codeword and is discarded.
3. The codeword is corrupted during transmission, and 000 is received (the right two bits are corrupted). This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected.

Error Correction

As we said before, error correction is much more difficult than error detection. In error detection, the receiver needs to know only that the received codeword is invalid; in error correction the receiver needs to find (or guess) the original codeword sent. We can say that we need more redundant bits for error correction than for error detection. Figure 10.7 shows the role of block coding in error correction. We can see that the idea is the same as error detection but the checker functions are much more complex.

Figure 10.7 *Structure of encoder and decoder in error correction*



Example 10.3

Let us add more redundant bits to Example 10.2 to see if the receiver can correct an error without knowing what was actually sent. We add 3 redundant bits to the 2-bit dataword to make 5-bit codewords. Again, later we will show how we chose the redundant bits. For the moment let us concentrate on the error correction concept. Table 10.2 shows the datawords and codewords.

Assume the dataword is 01. The sender consults the table (or uses an algorithm) to create the codeword 01011. The codeword is corrupted during transmission, and 01001 is received (error in the second bit from the right). First, the receiver finds that the received codeword is not in the table. This means an error has occurred. (Detection must come before correction.) The receiver, assuming that there is only 1 bit corrupted, uses the following strategy to guess the correct dataword.

Table 10.2 *A code for error correction (Example 10.3)*

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

1. Comparing the received codeword with the first codeword in the table (01001 versus 00000), the receiver decides that the first codeword is not the one that was sent because there are two different bits.
2. By the same reasoning, the original codeword cannot be the third or fourth one in the table.
3. The original codeword must be the second one in the table because this is the only one that differs from the received codeword by 1 bit. The receiver replaces 01001 with 01011 and consults the table to find the dataword 01.

Hamming Distance

One of the central concepts in coding for error control is the idea of the Hamming distance. The **Hamming distance** between two words (of the same size) is the number of differences between the corresponding bits. We show the Hamming distance between two words x and y as $d(x, y)$.

The Hamming distance can easily be found if we apply the XOR operation ($\oplus$) on the two words and count the number of 1s in the result. Note that the Hamming distance is a value greater than zero.

The Hamming distance between two words is the number of differences between corresponding bits.

Example 10.4

Let us find the Hamming distance between two pairs of words.

1. The Hamming distance $d(000, 011)$ is 2 because $000 \oplus 011$ is 011 (two 1s).
2. The Hamming distance $d(10101, 11110)$ is 3 because $10101 \oplus 11110$ is 01011 (three 1s).

Minimum Hamming Distance

Although the concept of the Hamming distance is the central point in dealing with error detection and correction codes, the measurement that is used for designing a code is the minimum Hamming distance. In a set of words, the **minimum Hamming distance** is the smallest Hamming distance between all possible pairs. We use d_{min} to define the minimum Hamming distance in a coding scheme. To find this value, we find the Hamming distances between all words and select the smallest one.

The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

Example 10.5

Find the minimum Hamming distance of the coding scheme in Table 10.1.

Solution

We first find all Hamming distances.

$$d(000, 011) = 2 \quad d(000, 101) = 2 \quad d(000, 110) = 2 \quad d(011, 101) = 2$$
$$d(011, 110) = 2 \quad d(101, 110) = 2$$

The d_{min} in this case is 2.

Example 10.6

Find the minimum Hamming distance of the coding scheme in Table 10.2.

Solution

We first find all the Hamming distances.

$$d(00000, 01011) = 3 \quad d(00000, 10101) = 3 \quad d(00000, 11110) = 4$$
$$d(01011, 10101) = 4 \quad d(01011, 11110) = 3 \quad d(10101, 11110) = 3$$

The d_{min} in this case is 3.

Three Parameters

Before we continue with our discussion, we need to mention that any coding scheme needs to have at least three parameters: the codeword size n, the dataword size k, and the minimum Hamming distance d_{min}. A coding scheme C is written as $C(n, k)$ with a separate expression for d_{min}. For example, we can call our first coding scheme $C(3, 2)$ with $d_{min} = 2$ and our second coding scheme $C(5, 2)$ with $d_{min} = 3$.

Hamming Distance and Error

Before we explore the criteria for error detection or correction, let us discuss the relationship between the Hamming distance and errors occurring during transmission. When a codeword is corrupted during transmission, the Hamming distance between the sent and received codewords is the number of bits affected by the error. In other words, the Hamming distance between the received codeword and the sent codeword is the number of bits that are corrupted during transmission. For example, if the codeword 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is $d(00000, 01101) = 3$.

Minimum Distance for Error Detection

Now let us find the minimum Hamming distance in a code if we want to be able to detect up to s errors. If s errors occur during transmission, the Hamming distance between the sent codeword and received codeword is s. If our code is to detect up to s errors, the minimum distance between the valid codes must be $s + 1$, so that the received codeword does not match a valid codeword. In other words, if the minimum distance between all valid codewords is $s + 1$, the received codeword cannot be erroneously mistaken for another codeword. The distances are not enough $(s + 1)$ for the receiver to accept it as valid. The error will be detected. We need to clarify a point here: Although a code with $d_{min} = s + 1$

may be able to detect more than s errors in some special cases, only s or fewer errors are guaranteed to be detected.

To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.

Example 10.7

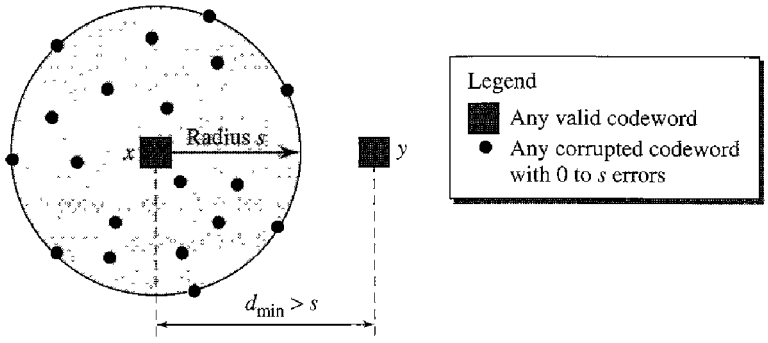
The minimum Hamming distance for our first code scheme (Table 10.1) is 2. This code guarantees detection of only a single error. For example, if the third codeword (101) is sent and one error occurs, the received codeword does not match any valid codeword. If two errors occur, however, the received codeword may match a valid codeword and the errors are not detected.

Example 10.8

Our second block code scheme (Table 10.2) has $d_{min} = 3$. This code can detect up to two errors. Again, we see that when any of the valid codewords is sent, two errors create a codeword which is not in the table of valid codewords. The receiver cannot be fooled. However, some combinations of three errors change a valid codeword to another valid codeword. The receiver accepts the received codeword and the errors are undetected.

We can look at this geometrically. Let us assume that the sent codeword x is at the center of a circle with radius s. All other received codewords that are created by 1 to s errors are points inside the circle or on the perimeter of the circle. All other valid codewords must be outside the circle, as shown in Figure 10.8.

Figure 10.8 *Geometric concept for finding d_{min} in error detection*



In Figure 10.8, d_{min} must be an integer greater than s; that is, $d_{min} = s + 1$.

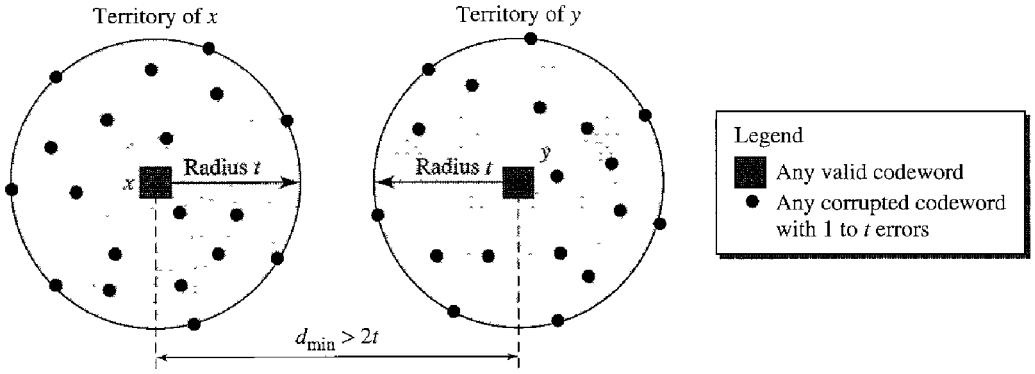
Minimum Distance for Error Correction

Error correction is more complex than error detection; a decision is involved. When a received codeword is not a valid codeword, the receiver needs to decide which valid codeword was actually sent. The decision is based on the concept of territory, an exclusive area surrounding the codeword. Each valid codeword has its own territory.

We use a geometric approach to define each territory. We assume that each valid codeword has a circular territory with a radius of t and that the valid codeword is at the

center. For example, suppose a codeword x is corrupted by t bits or less. Then this corrupted codeword is located either inside or on the perimeter of this circle. If the receiver receives a codeword that belongs to this territory, it decides that the original codeword is the one at the center. Note that we assume that only up to t errors have occurred; otherwise, the decision is wrong. Figure 10.9 shows this geometric interpretation. Some texts use a sphere to show the distance between all valid block codes.

Figure 10.9 *Geometric concept for finding d_{min} in error correction*



In Figure 10.9, $d_{min} > 2t$; since the next integer increment is 1, we can say that $d_{min} = 2t + 1$.

To guarantee correction of up to t errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.

Example 10.9

A code scheme has a Hamming distance $d_{min} = 4$. What is the error detection and correction capability of this scheme?

Solution

This code guarantees the detection of up to three errors ($s = 3$), but it can correct up to one error. In other words, if this code is used for error correction, part of its capability is wasted. Error correction codes need to have an odd minimum distance (3, 5, 7, . . .).

10.3 LINEAR BLOCK CODES

Almost all block codes used today belong to a subset called **linear block codes.** The use of nonlinear block codes for error detection and correction is not as widespread because their structure makes theoretical analysis and implementation difficult. We therefore concentrate on linear block codes.

The formal definition of linear block codes requires the knowledge of abstract algebra (particularly Galois fields), which is beyond the scope of this book. We therefore give an informal definition. For our purposes, a linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

In a linear block code, the exclusive OR (XOR) of any two valid codewords creates another valid codeword.

Example 10.10

Let us see if the two codes we defined in Table 10.1 and Table 10.2 belong to the class of linear block codes.

1. The scheme in Table 10.1 is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third codewords creates the fourth one.
2. The scheme in Table 10.2 is also a linear block code. We can create all four codewords by XORing two other codewords.

Minimum Distance for Linear Block Codes

It is simple to find the minimum Hamming distance for a linear block code. The minimum Hamming distance is the number of 1s in the nonzero valid codeword with the smallest number of 1s.

Example 10.11

In our first code (Table 10.1), the numbers of 1s in the nonzero codewords are 2, 2, and 2. So the minimum Hamming distance is $d_{min} = 2$. In our second code (Table 10.2), the numbers of 1s in the nonzero codewords are 3, 3, and 4. So in this code we have $d_{min} = 3$.

Some Linear Block Codes

Let us now show some linear block codes. These codes are trivial because we can easily find the encoding and decoding algorithms and check their performances.

Simple Parity-Check Code

Perhaps the most familiar error-detecting code is the **simple parity-check code.** In this code, a k-bit dataword is changed to an n-bit codeword where $n = k + 1$. The extra bit, called the parity bit, is selected to make the total number of 1s in the codeword even. Although some implementations specify an odd number of 1s, we discuss the even case. The minimum Hamming distance for this category is $d_{min} = 2$, which means that the code is a single-bit error-detecting code; it cannot correct any error.

A simple parity-check code is a single-bit error-detecting code in which $n = k + 1$ with $d_{min} = 2$.

Our first code (Table 10.1) is a parity-check code with $k = 2$ and $n = 3$. The code in Table 10.3 is also a parity-check code with $k = 4$ and $n = 5$.

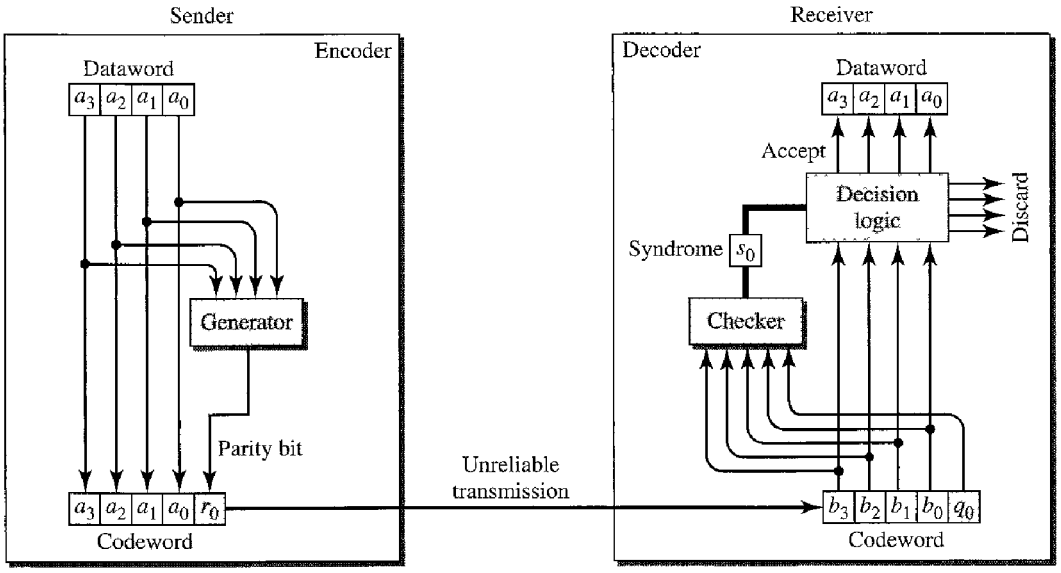
Figure 10.10 shows a possible structure of an encoder (at the sender) and a decoder (at the receiver).

The encoder uses a generator that takes a copy of a 4-bit dataword (a_0, a_1, a_2, and a_3) and generates a parity bit r_0. The dataword bits and the **parity bit** create the 5-bit codeword. The parity bit that is added makes the number of 1s in the codeword even.

Table 10.3 *Simple parity-check code C(5, 4)*

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Figure 10.10 *Encoder and decoder for simple parity-check code*



This is normally done by adding the 4 bits of the dataword (modulo-2); the result is the parity bit. In other words,

$$r_0 = a_3 + a_2 + a_1 + a_0 \qquad \text{(modulo-2)}$$

If the number of 1s is even, the result is 0; if the number of 1s is odd, the result is 1. In both cases, the total number of 1s in the codeword is even.

The sender sends the codeword which may be corrupted during transmission. The receiver receives a 5-bit word. The checker at the receiver does the same thing as the generator in the sender with one exception: The addition is done over all 5 bits. The result, which is called the **syndrome,** is just 1 bit. The syndrome is 0 when the number of 1s in the received codeword is even; otherwise, it is 1.

$$s_0 = b_3 + b_2 + b_1 + b_0 + q_0 \qquad \text{(modulo-2)}$$

The syndrome is passed to the decision logic analyzer. If the syndrome is 0, there is no error in the received codeword; the data portion of the received codeword is accepted as the dataword; if the syndrome is 1, the data portion of the received codeword is discarded. The dataword is not created.

Example 10.12

Let us look at some transmission scenarios. Assume the sender sends the dataword 1011. The codeword created from this dataword is 10111, which is sent to the receiver. We examine five cases:

1. No error occurs; the received codeword is 10111. The syndrome is 0. The dataword 1011 is created.
2. One single-bit error changes a_1. The received codeword is 10011. The syndrome is 1. No dataword is created.
3. One single-bit error changes r_0. The received codeword is 10110. The syndrome is 1. No dataword is created. Note that although none of the dataword bits are corrupted, no dataword is created because the code is not sophisticated enough to show the position of the corrupted bit.
4. An error changes r_0 and a second error changes a_3. The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value. The simple parity-check decoder cannot detect an even number of errors. The errors cancel each other out and give the syndrome a value of 0.
5. Three bits—a_3, a_2, and a_1—are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of errors.

A simple parity-check code can detect an odd number of errors.

A better approach is the **two-dimensional parity check.** In this method, the dataword is organized in a table (rows and columns). In Figure 10.11, the data to be sent, five 7-bit bytes, are put in separate rows. For each row and each column, 1 parity-check bit is calculated. The whole table is then sent to the receiver, which finds the syndrome for each row and each column. As Figure 10.11 shows, the two-dimensional parity check can detect up to three errors that occur anywhere in the table (arrows point to the locations of the created nonzero syndromes). However, errors affecting 4 bits may not be detected.

Hamming Codes

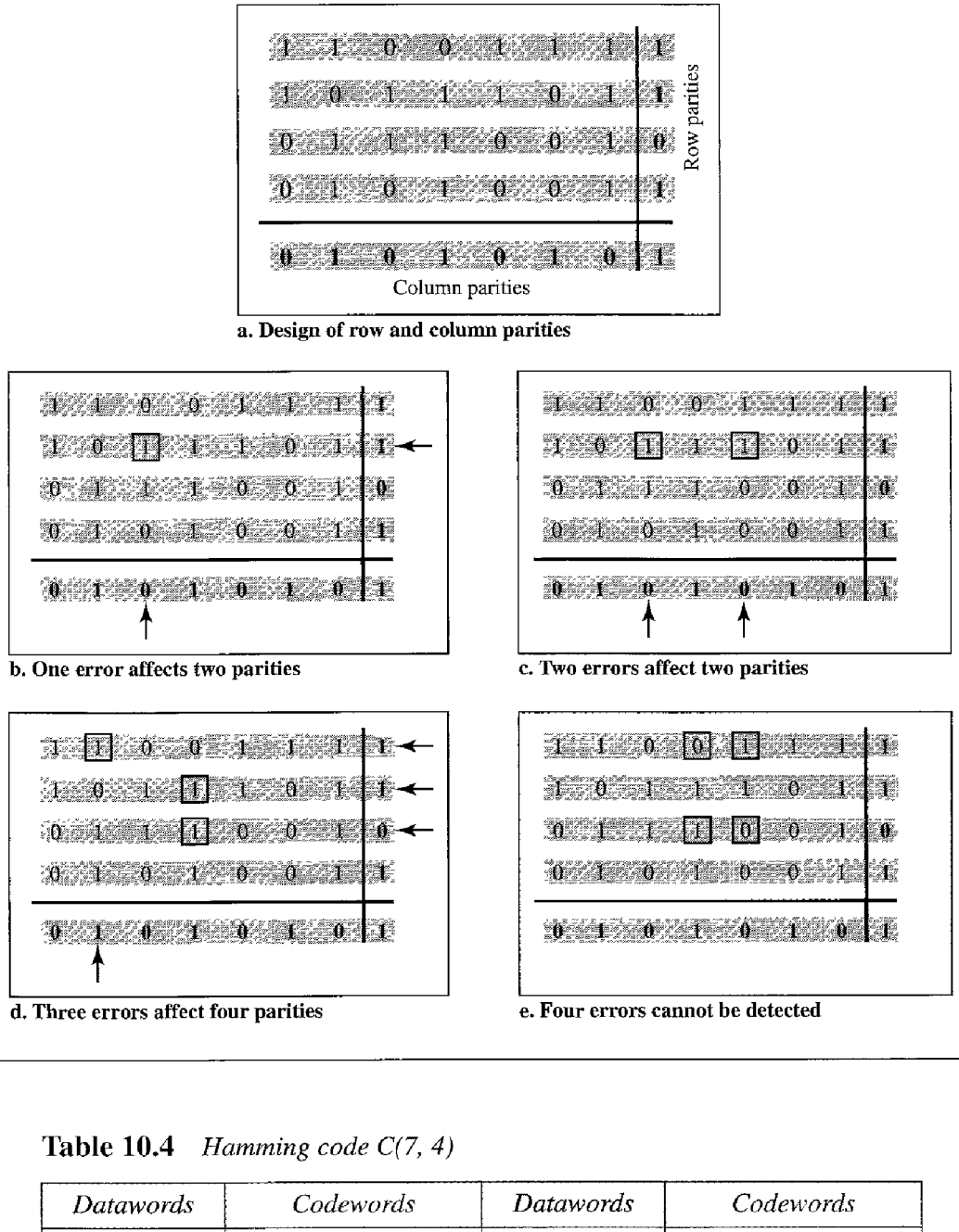
Now let us discuss a category of error-correcting codes called **Hamming codes.** These codes were originally designed with $d_{min} = 3$, which means that they can detect up to two errors or correct one single error. Although there are some Hamming codes that can correct more than one error, our discussion focuses on the single-bit error-correcting code.

First let us find the relationship between n and k in a Hamming code. We need to choose an integer $m >= 3$. The values of n and k are then calculated from m as $n = 2^m - 1$ and $k = n - m$. The number of check bits $r = m$.

All Hamming codes discussed in this book have $d_{min} = 3$.
The relationship between m and n in these codes is $n = 2^m - 1$.

For example, if $m = 3$, then $n = 7$ and $k = 4$. This is a Hamming code $C(7, 4)$ with $d_{min} = 3$. Table 10.4 shows the datawords and codewords for this code.

Figure 10.11 *Two-dimensional parity-check code*



a. Design of row and column parities

b. One error affects two parities

c. Two errors affect two parities

d. Three errors affect four parities

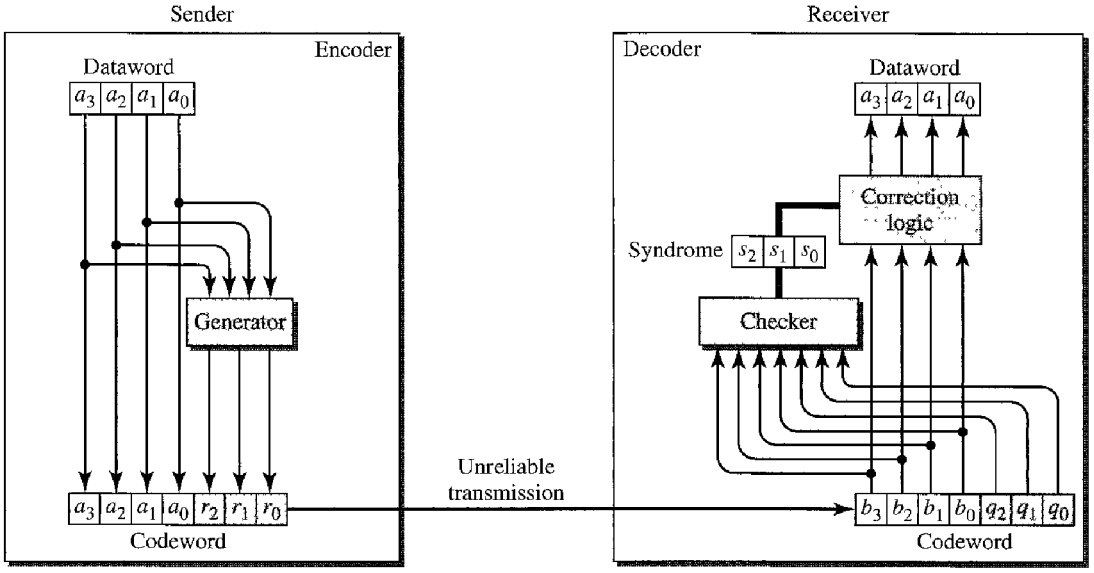
e. Four errors cannot be detected

Table 10.4 *Hamming code C(7, 4)*

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	1011100
0100	0100011	1100	1100101
0101	0101110	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111

Figure 10.12 shows the structure of the encoder and decoder for this example.

Figure 10.12 *The structure of the encoder and decoder for a Hamming code*



A copy of a 4-bit dataword is fed into the generator that creates three parity checks r_0, r_1, and r_2, as shown below:

$$r_0 = a_2 + a_1 + a_0 \quad \text{modulo-2}$$
$$r_1 = a_3 + a_2 + a_1 \quad \text{modulo-2}$$
$$r_2 = a_1 + a_0 + a_3 \quad \text{modulo-2}$$

In other words, each of the parity-check bits handles 3 out of the 4 bits of the dataword. The total number of 1s in each 4-bit combination (3 dataword bits and 1 parity bit) must be even. We are not saying that these three equations are unique; any three equations that involve 3 of the 4 bits in the dataword and create independent equations (a combination of two cannot create the third) are valid.

The checker in the decoder creates a 3-bit syndrome ($s_2 s_1 s_0$) in which each bit is the parity check for 4 out of the 7 bits in the received codeword:

$$s_0 = b_2 + b_1 + b_0 + q_0 \quad \text{modulo-2}$$
$$s_1 = b_3 + b_2 + b_1 + q_1 \quad \text{modulo-2}$$
$$s_2 = b_1 + b_0 + b_3 + q_2 \quad \text{modulo-2}$$

The equations used by the checker are the same as those used by the generator with the parity-check bits added to the right-hand side of the equation. The 3-bit syndrome creates eight different bit patterns (000 to 111) that can represent eight different conditions. These conditions define a lack of error or an error in 1 of the 7 bits of the received codeword, as shown in Table 10.5.

Table 10.5 *Logical decision made by the correction logic analyzer of the decoder*

Syndrome	000	001	010	011	100	101	110	111
Error	None	q_0	q_1	b_2	q_2	b_0	b_3	b_1

Note that the generator is not concerned with the four cases shaded in Table 10.5 because there is either no error or an error in the parity bit. In the other four cases, 1 of the bits must be flipped (changed from 0 to 1 or 1 to 0) to find the correct dataword.

The syndrome values in Table 10.5 are based on the syndrome bit calculations. For example, if q_0 is in error, s_0 is the only bit affected; the syndrome, therefore, is 001. If b_2 is in error, s_0 and s_1 are the bits affected; the syndrome, therefore is 011. Similarly, if b_1 is in error, all 3 syndrome bits are affected and the syndrome is 111.

There are two points we need to emphasize here. First, if two errors occur during transmission, the created dataword might not be the right one. Second, if we want to use the above code for error detection, we need a different design.

Example 10.13

Let us trace the path of three datawords from the sender to the destination:

1. The dataword 0100 becomes the codeword 0100011. The codeword 0100011 is received. The syndrome is 000 (no error), the final dataword is 0100.
2. The dataword 0111 becomes the codeword 0111001. The codeword 0011001 is received. The syndrome is 011. According to Table 10.5, b_2 is in error. After flipping b_2 (changing the 1 to 0), the final dataword is 0111.
3. The dataword 1101 becomes the codeword 1101000. The codeword 0001000 is received (two errors). The syndrome is 101, which means that b_0 is in error. After flipping b_0, we get 0000, the wrong dataword. This shows that our code cannot correct two errors.

Example 10.14

We need a dataword of at least 7 bits. Calculate values of k and n that satisfy this requirement.

Solution

We need to make $k = n - m$ greater than or equal to 7, or $2^m - 1 - m \geq 7$.

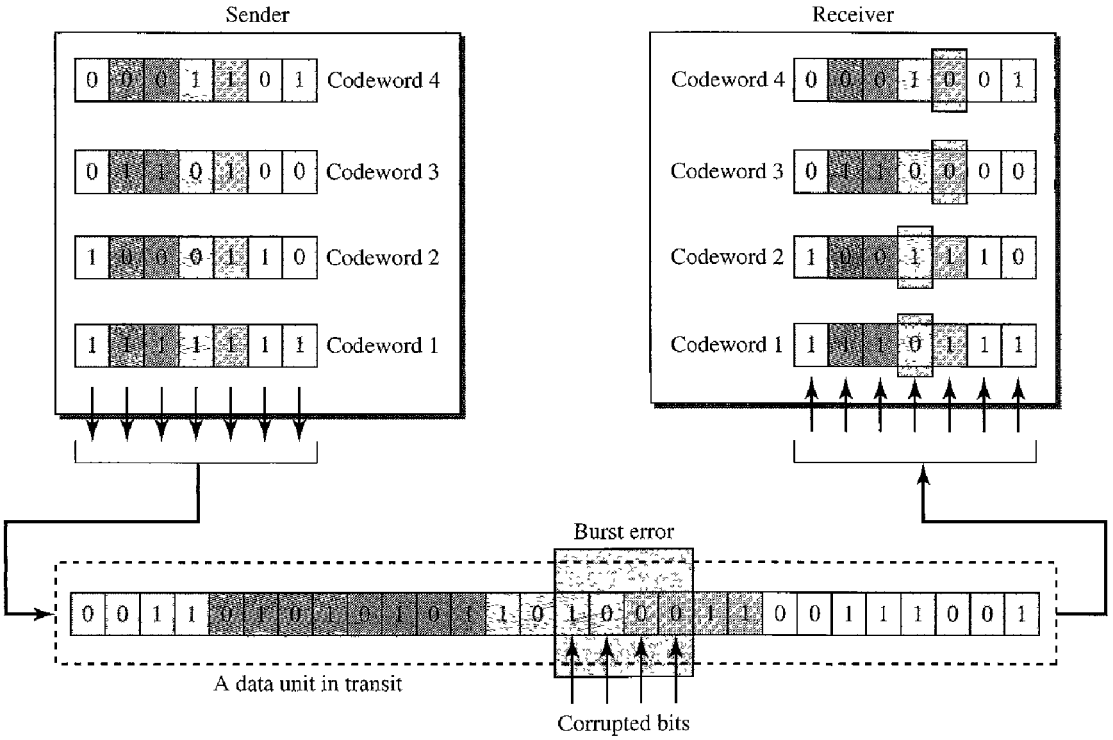
1. If we set $m = 3$, the result is $n = 2^3 - 1$ and $k = 7 - 3$, or 4, which is not acceptable.
2. If we set $m = 4$, then $n = 2^4 - 1 = 15$ and $k = 15 - 4 = 11$, which satisfies the condition. So the code is $C(15, 11)$. There are methods to make the dataword a specific size, but the discussion and implementation are beyond the scope of this book.

Performance

A Hamming code can only correct a single error or detect a double error. However, there is a way to make it detect a burst error, as shown in Figure 10.13.

The key is to split a burst error between several codewords, one error for each codeword. In data communications, we normally send a packet or a frame of data. To make the Hamming code respond to a burst error of size N, we need to make N codewords out of our frame. Then, instead of sending one codeword at a time, we arrange the codewords in a table and send the bits in the table a column at a time. In Figure 10.13, the bits are sent column by column (from the left). In each column, the bits are sent from the bottom to the top. In this way, a frame is made out of the four codewords and sent to the receiver. Figure 10.13 shows

Figure 10.13 *Burst error correction using Hamming code*



that when a burst error of size 4 corrupts the frame, only 1 bit from each codeword is corrupted. The corrupted bit in each codeword can then easily be corrected at the receiver.

10.4 CYCLIC CODES

Cyclic codes are special linear block codes with one extra property. In a **cyclic code,** if a codeword is cyclically shifted (rotated), the result is another codeword. For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword. In this case, if we call the bits in the first word a_0 to a_6, and the bits in the second word b_0 to b_6, we can shift the bits by using the following:

$$b_1 = a_0 \quad b_2 = a_1 \quad b_3 = a_2 \quad b_4 = a_3 \quad b_5 = a_4 \quad b_6 = a_5 \quad b_0 = a_6$$

In the rightmost equation, the last bit of the first word is wrapped around and becomes the first bit of the second word.

Cyclic Redundancy Check

We can create cyclic codes to correct errors. However, the theoretical background required is beyond the scope of this book. In this section, we simply discuss a category of cyclic codes called the **cyclic redundancy check (CRC)** that is used in networks such as LANs and WANs.

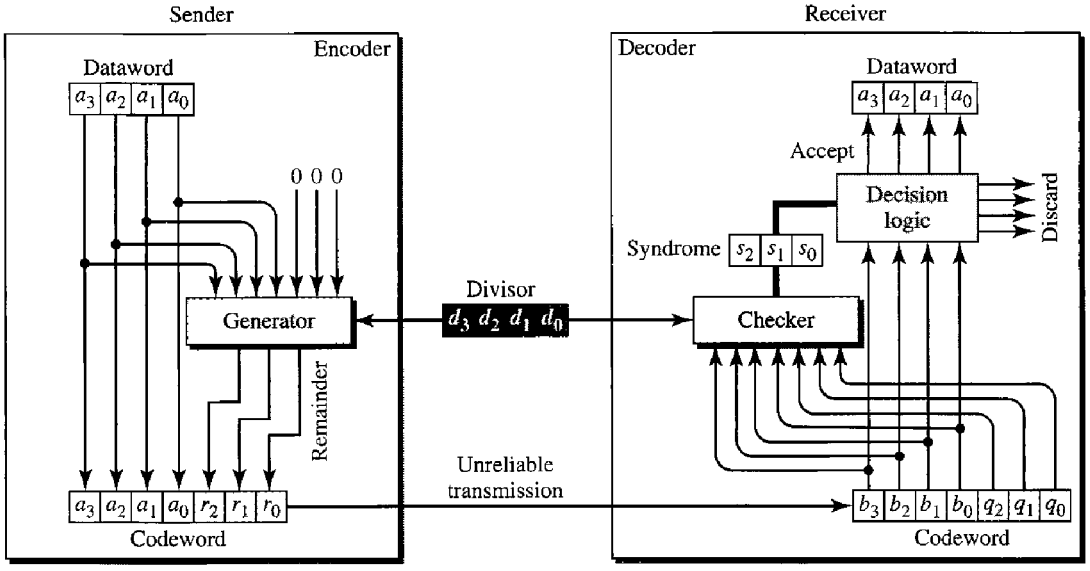
Table 10.6 shows an example of a CRC code. We can see both the linear and cyclic properties of this code.

Table 10.6 *A CRC code with C(7, 4)*

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

Figure 10.14 shows one possible design for the encoder and decoder.

Figure 10.14 *CRC encoder and decoder*



In the encoder, the dataword has k bits (4 here); the codeword has n bits (7 here). The size of the dataword is augmented by adding $n - k$ (3 here) 0s to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size $n - k + 1$ (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder ($r_2 r_1 r_0$) is appended to the dataword to create the codeword.

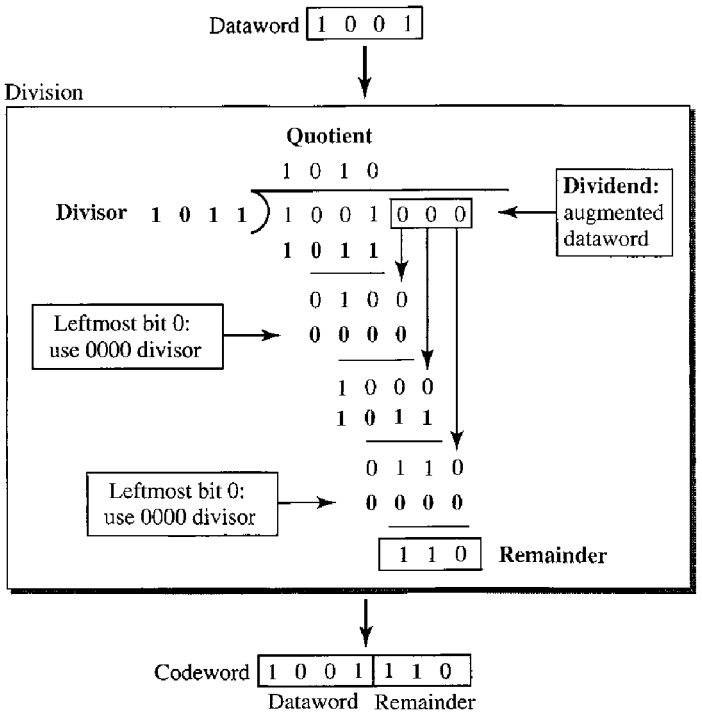
The decoder receives the possibly corrupted codeword. A copy of all n bits is fed to the checker which is a replica of the generator. The remainder produced by the checker

is a syndrome of $n - k$ (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all 0s, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).

Encoder

Let us take a closer look at the encoder. The encoder takes the dataword and augments it with $n - k$ number of 0s. It then divides the augmented dataword by the divisor, as shown in Figure 10.15.

Figure 10.15 *Division in CRC encoder*



The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers. However, as mentioned at the beginning of the chapter, in this case addition and subtraction are the same. We use the XOR operation to do both.

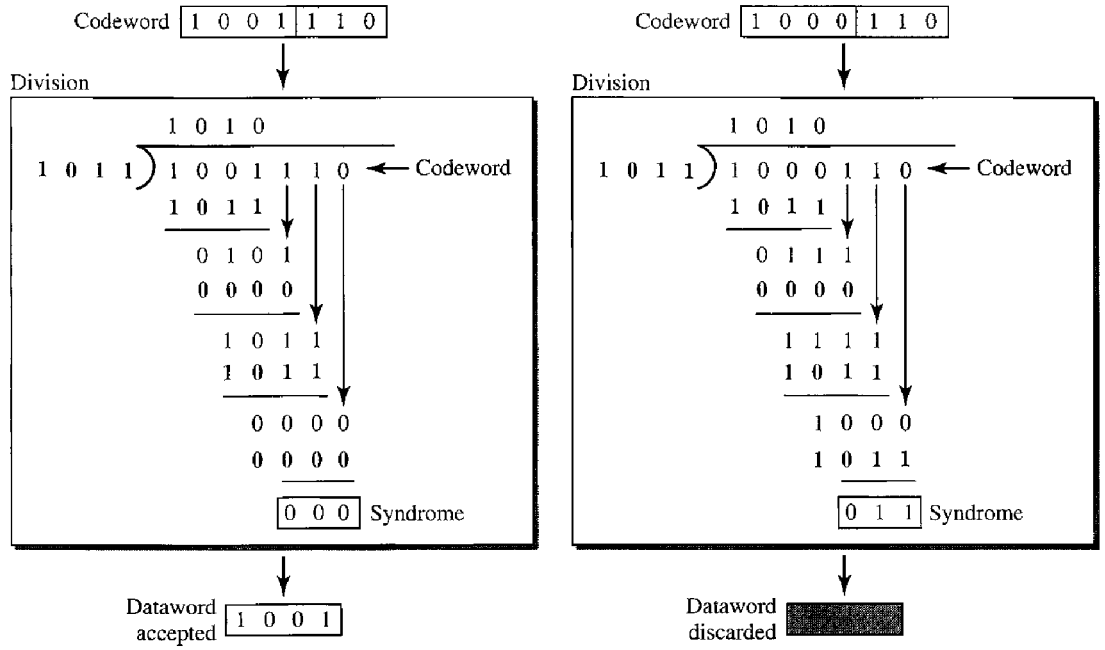
As in decimal division, the process is done step by step. In each step, a copy of the divisor is XORed with the 4 bits of the dividend. The result of the XOR operation (remainder) is 3 bits (in this case), which is used for the next step after 1 extra bit is pulled down to make it 4 bits long. There is one important point we need to remember in this type of division. If the leftmost bit of the dividend (or the part used in each step) is 0, the step cannot use the regular divisor; we need to use an all-0s divisor.

When there are no bits left to pull down, we have a result. The 3-bit remainder forms the check bits (r_2, r_1, and r_0). They are appended to the dataword to create the codeword.

Decoder

The codeword can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. If the syndrome is all 0s, there is no error; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded. Figure 10.16 shows two cases: The left-hand figure shows the value of syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is one single error. The syndrome is not all 0s (it is 011).

Figure 10.16 *Division in the CRC decoder for two cases*



Divisor

You may be wondering how the divisor 1011 is chosen. Later in the chapter we present some criteria, but in general it involves abstract algebra.

Hardware Implementation

One of the advantages of a cyclic code is that the encoder and decoder can easily and cheaply be implemented in hardware by using a handful of electronic devices. Also, a hardware implementation increases the rate of check bit and syndrome bit calculation. In this section, we try to show, step by step, the process. The section, however, is optional and does not affect the understanding of the rest of the chapter.

Divisor

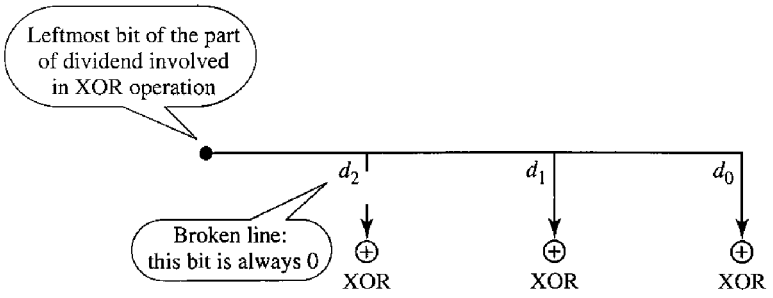
Let us first consider the divisor. We need to note the following points:

1. The divisor is repeatedly XORed with part of the dividend.

2. The divisor has $n - k + 1$ bits which either are predefined or are all 0s. In other words, the bits do not change from one dataword to another. In our previous example, the divisor bits were either 1011 or 0000. The choice was based on the leftmost bit of the part of the augmented data bits that are active in the XOR operation.
3. A close look shows that only $n - k$ bits of the divisor is needed in the XOR operation. The leftmost bit is not needed because the result of the operation is always 0, no matter what the value of this bit. The reason is that the inputs to this XOR operation are either both 0s or both 1s. In our previous example, only 3 bits, not 4, is actually used in the XOR operation.

Using these points, we can make a fixed (hardwired) divisor that can be used for a cyclic code if we know the divisor pattern. Figure 10.17 shows such a design for our previous example. We have also shown the XOR devices used for the operation.

Figure 10.17 *Hardwired design of the divisor in CRC*



Note that if the leftmost bit of the part of dividend to be used in this step is 1, the divisor bits ($d_2 d_1 d_0$) are 011; if the leftmost bit is 0, the divisor bits are 000. The design provides the right choice based on the leftmost bit.

Augmented Dataword

In our paper-and-pencil division process in Figure 10.15, we show the augmented dataword as fixed in position with the divisor bits shifting to the right, 1 bit in each step. The divisor bits are aligned with the appropriate part of the augmented dataword. Now that our divisor is fixed, we need instead to shift the bits of the augmented dataword to the left (opposite direction) to align the divisor bits with the appropriate part. There is no need to store the augmented dataword bits.

Remainder

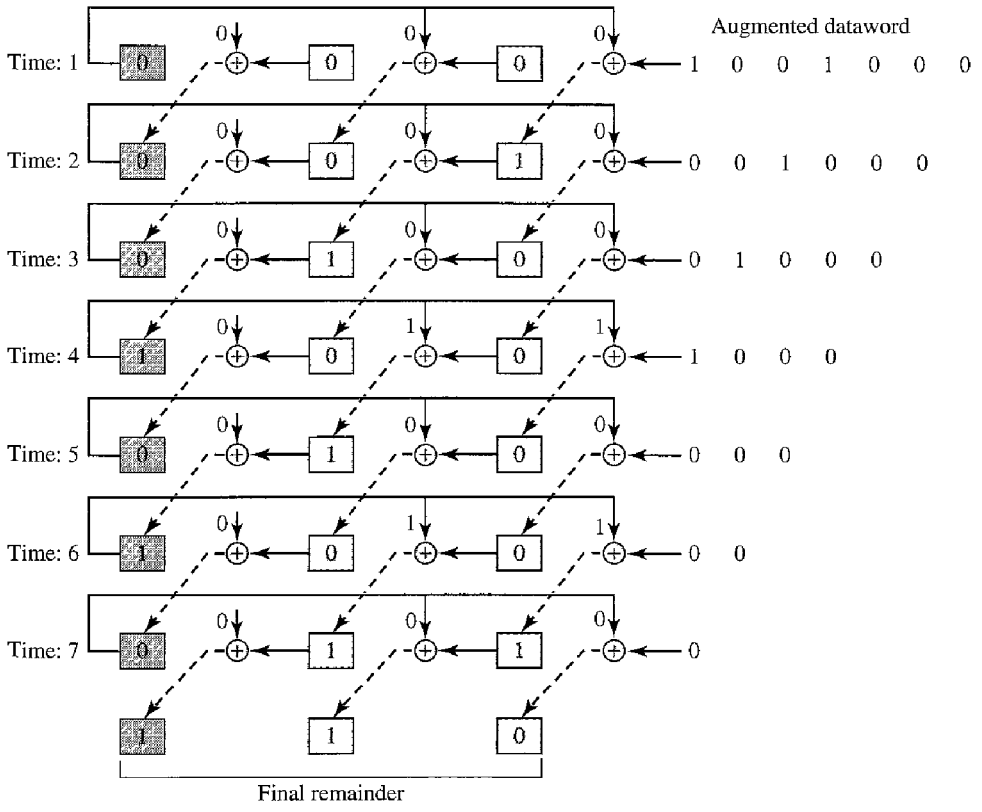
In our previous example, the remainder is 3 bits ($n - k$ bits in general) in length. We can use three **registers** (single-bit storage devices) to hold these bits. To find the final remainder of the division, we need to modify our division process. The following is the step-by-step process that can be used to simulate the division process in hardware (or even in software).

1. We assume that the remainder is originally all 0s (000 in our example).

2. At each time click (arrival of 1 bit from an augmented dataword), we repeat the following two actions:
   a. We use the leftmost bit to make a decision about the divisor (011 or 000).
   b. The other 2 bits of the remainder and the next bit from the augmented dataword (total of 3 bits) are XORed with the 3-bit divisor to create the next remainder.

Figure 10.18 shows this simulator, but note that this is not the final design; there will be more improvements.

Figure 10.18 *Simulation of division in CRC encoder*

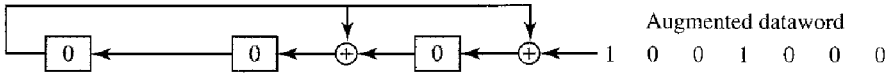


At each clock tick, shown as different times, one of the bits from the augmented dataword is used in the XOR process. If we look carefully at the design, we have seven steps here, while in the paper-and-pencil method we had only four steps. The first three steps have been added here to make each step equal and to make the design for each step the same. Steps 1, 2, and 3 push the first 3 bits to the remainder registers; steps 4, 5, 6, and 7 match the paper-and-pencil design. Note that the values in the remainder register in steps 4 to 7 exactly match the values in the paper-and-pencil design. The final remainder is also the same.

The above design is for demonstration purposes only. It needs simplification to be practical. First, we do not need to keep the intermediate values of the remainder bits; we need only the final bits. We therefore need only 3 registers instead of 24. After the XOR operations, we do not need the bit values of the previous remainder. Also, we do

not need 21 XOR devices; two are enough because the output of an XOR operation in which one of the bits is 0 is simply the value of the other bit. This other bit can be used as the output. With these two modifications, the design becomes tremendously simpler and less expensive, as shown in Figure 10.19.

Figure 10.19 *The CRC encoder design using shift registers*

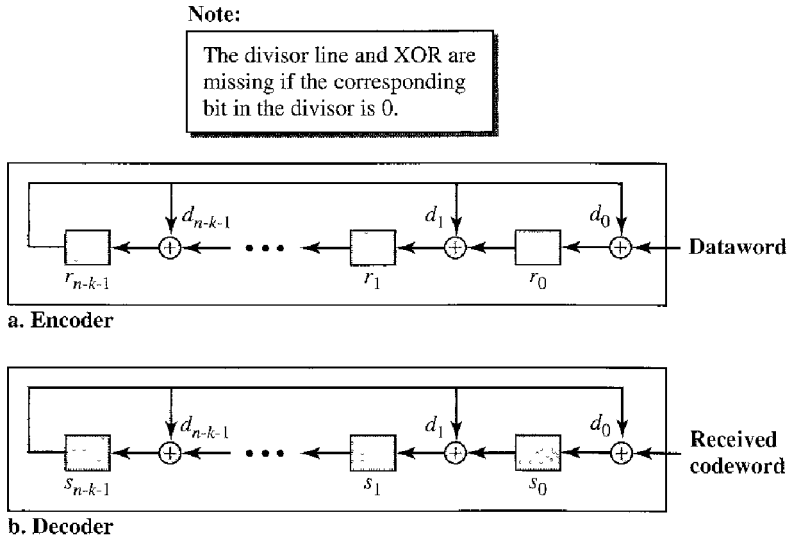


We need, however, to make the registers shift registers. A 1-bit shift register holds a bit for a duration of one clock time. At a time click, the shift register accepts the bit at its input port, stores the new bit, and displays it on the output port. The content and the output remain the same until the next input arrives. When we connect several 1-bit shift registers together, it looks as if the contents of the register are shifting.

General Design

A general design for the encoder and decoder is shown in Figure 10.20.

Figure 10.20 *General design of encoder and decoder of a CRC code*



Note that we have $n - k$ 1-bit shift registers in both the encoder and decoder. We have up to $n - k$ XOR devices, but the divisors normally have several 0s in their pattern, which reduces the number of devices. Also note that, instead of augmented datawords, we show the dataword itself as the input because after the bits in the dataword are all fed into the encoder, the extra bits, which all are 0s, do not have any effect on the rightmost XOR. Of course, the process needs to be continued for another $n - k$ steps before

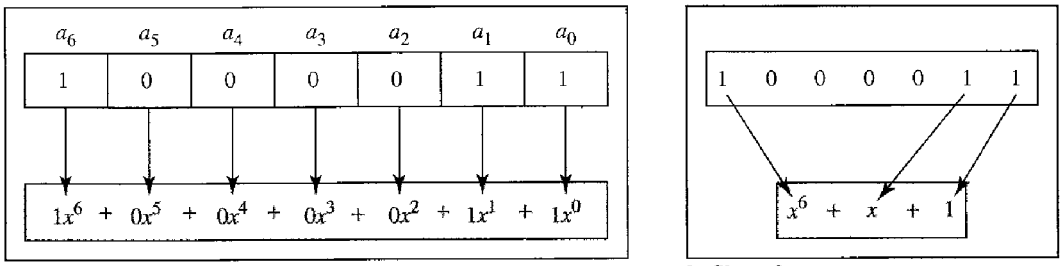
the check bits are ready. This fact is one of the criticisms of this design. Better schemes have been designed to eliminate this waiting time (the check bits are ready after k steps), but we leave this as a research topic for the reader. In the decoder, however, the entire codeword must be fed to the decoder before the syndrome is ready.

Polynomials

A better way to understand cyclic codes and how they can be analyzed is to represent them as polynomials. Again, this section is optional.

A pattern of 0s and 1s can be represented as a **polynomial** with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit. Figure 10.21 shows a binary pattern and its polynomial representation. In Figure 10.21a we show how to translate a binary pattern to a polynomial; in Figure 10.21b we show how the polynomial can be shortened by removing all terms with zero coefficients and replacing x^1 by x and x^0 by 1.

Figure 10.21 *A polynomial to represent a binary word*



a. Binary pattern and polynomial

b. Short form

Figure 10.21 shows one immediate benefit; a 7-bit pattern can be replaced by three terms. The benefit is even more conspicuous when we have a polynomial such as $x^{23} + x^3 + 1$. Here the bit pattern is 24 bits in length (three 1s and twenty-one 0s) while the polynomial is just three terms.

Degree of a Polynomial

The degree of a polynomial is the highest power in the polynomial. For example, the degree of the polynomial $x^6 + x + 1$ is 6. Note that the degree of a polynomial is 1 less that the number of bits in the pattern. The bit pattern in this case has 7 bits.

Adding and Subtracting Polynomials

Adding and subtracting polynomials in mathematics are done by adding or subtracting the coefficients of terms with the same power. In our case, the coefficients are only 0 and 1, and adding is in modulo-2. This has two consequences. First, addition and subtraction are the same. Second, adding or subtracting is done by combining terms and deleting pairs of identical terms. For example, adding $x^5 + x^4 + x^2$ and $x^6 + x^4 + x^2$ gives just $x^6 + x^5$. The terms x^4 and x^2 are deleted. However, note that if we add, for example, three polynomials and we get x^2 three times, we delete a pair of them and keep the third.

Multiplying or Dividing Terms

In this arithmetic, multiplying a term by another term is very simple; we just add the powers. For example, $x^3 \times x^4$ is x^7. For dividing, we just subtract the power of the second term from the power of the first. For example, x^5/x^2 is x^3.

Multiplying Two Polynomials

Multiplying a polynomial by another is done term by term. Each term of the first polynomial must be multiplied by all terms of the second. The result, of course, is then simplified, and pairs of equal terms are deleted. The following is an example:

$$\begin{aligned}(x^5 + x^3 + x^2 + x)(x^2 + x + 1) \\ &= x^7 + x^6 + x^5 + x^5 + x^4 + x^3 + x^4 + x^3 + x^2 + x^3 + x^2 + x \\ &= x^7 + x^6 + x^3 + x\end{aligned}$$

Dividing One Polynomial by Another

Division of polynomials is conceptually the same as the binary division we discussed for an encoder. We divide the first term of the dividend by the first term of the divisor to get the first term of the quotient. We multiply the term in the quotient by the divisor and subtract the result from the dividend. We repeat the process until the dividend degree is less than the divisor degree. We will show an example of division later in this chapter.

Shifting

A binary pattern is often shifted a number of bits to the right or left. Shifting to the left means adding extra 0s as rightmost bits; shifting to the right means deleting some rightmost bits. Shifting to the left is accomplished by multiplying each term of the polynomial by x^m, where m is the number of shifted bits; shifting to the right is accomplished by dividing each term of the polynomial by x^m. The following shows shifting to the left and to the right. Note that we do not have negative powers in the polynomial representation.

Shifting left 3 bits:	10011 becomes 10011000	$x^4 + x + 1$ becomes $x^7 + x^4 + x^3$
Shifting right 3 bits:	10011 becomes 10	$x^4 + x + 1$ becomes x

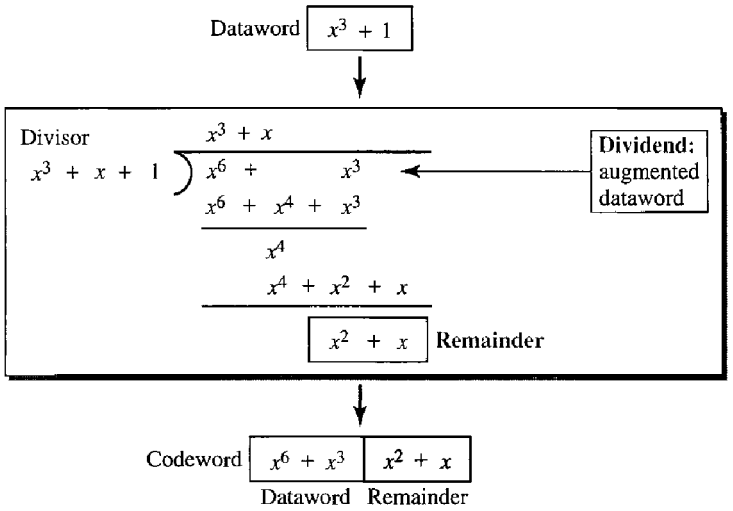
When we augmented the dataword in the encoder of Figure 10.15, we actually shifted the bits to the left. Also note that when we concatenate two bit patterns, we shift the first polynomial to the left and then add the second polynomial.

Cyclic Code Encoder Using Polynomials

Now that we have discussed operations on polynomials, we show the creation of a codeword from a dataword. Figure 10.22 is the polynomial version of Figure 10.15. We can see that the process is shorter. The dataword 1001 is represented as $x^3 + 1$. The divisor 1011 is represented as $x^3 + x + 1$. To find the augmented dataword, we have left-shifted the dataword 3 bits (multiplying by x^3). The result is $x^6 + x^3$. Division is straightforward. We divide the first term of the dividend, x^6, by the first term of the divisor, x^3. The first term of the quotient is then x^6/x^3, or x^3. Then we multiply x^3 by the divisor and subtract (according to our previous definition of subtraction) the result from the dividend. The

result is x^4, with a degree greater than the divisor's degree; we continue to divide until the degree of the remainder is less than the degree of the divisor.

Figure 10.22 *CRC division using polynomials*



It can be seen that the polynomial representation can easily simplify the operation of division in this case, because the two steps involving all-0s divisors are not needed here. (Of course, one could argue that the all-0s divisor step can also be eliminated in binary division.) In a polynomial representation, the divisor is normally referred to as the **generator polynomial** $t(x)$.

The divisor in a cyclic code is normally called the generator polynomial or simply the generator.

Cyclic Code Analysis

We can analyze a cyclic code to find its capabilities by using polynomials. We define the following, where $f(x)$ is a polynomial with binary coefficients.

Dataword: $d(x)$ Codeword: $c(x)$ Generator: $g(x)$
Syndrome: $s(x)$ Error: $e(x)$

If $s(x)$ is not zero, then one or more bits is corrupted. However, if $s(x)$ is zero, either no bit is corrupted or the decoder failed to detect any errors.

In a cyclic code,

1. If $s(x) \neq 0$, one or more bits is corrupted.
2. If $s(x) = 0$, either
   a. No bit is corrupted. or
   b. Some bits are corrupted, but the decoder failed to detect them.

In our analysis we want to find the criteria that must be imposed on the generator, $g(x)$ to detect the type of error we especially want to be detected. Let us first find the relationship among the sent codeword, error, received codeword, and the generator. We can say

$$\text{Received codeword} = c(x) + e(x)$$

In other words, the received codeword is the sum of the sent codeword and the error. The receiver divides the received codeword by $g(x)$ to get the syndrome. We can write this as

$$\frac{\text{Received codeword}}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

The first term at the right-hand side of the equality does not have a remainder (according to the definition of codeword). So the syndrome is actually the remainder of the second term on the right-hand side. If this term does not have a remainder (syndrome = 0), either $e(x)$ is 0 or $e(x)$ is divisible by $g(x)$. We do not have to worry about the first case (there is no error); the second case is very important. Those errors that are divisible by $g(x)$ are not caught.

In a cyclic code, those $e(x)$ errors that are divisible by $g(x)$ are not caught.

Let us show some specific errors and see how they can be caught by a well-designed $g(x)$.

Single-Bit Error

What should be the structure of $g(x)$ to guarantee the detection of a single-bit error? A single-bit error is $e(x) = x^i$, where i is the position of the bit. If a single-bit error is caught, then x^i is not divisible by $g(x)$. (Note that when we say *not divisible,* we mean that there is a remainder.) If $g(x)$ has at least two terms (which is normally the case) and the coefficient of x^0 is not zero (the rightmost bit is 1), then $e(x)$ cannot be divided by $g(x)$.

If the generator has more than one term and the coefficient of x^0 is 1, all single errors can be caught.

Example 10.15

Which of the following $g(x)$ values guarantees that a single-bit error is caught? For each case, what is the error that cannot be caught?

a. $x + 1$

b. x^3

c. 1

Solution

a. No x^i can be divisible by $x + 1$. In other words, $x^i/(x + 1)$ always has a remainder. So the syndrome is nonzero. Any single-bit error can be caught.

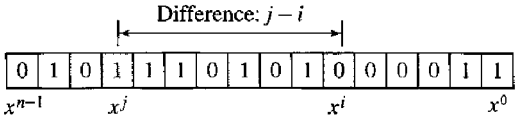
b. If i is equal to or greater than 3, x^i is divisible by $g(x)$. The remainder of x^i/x^3 is zero, and the receiver is fooled into believing that there is no error, although there might be one. Note that in this case, the corrupted bit must be in position 4 or above. All single-bit errors in positions 1 to 3 are caught.

c. All values of i make x^i divisible by $g(x)$. No single-bit error can be caught. In addition, this $g(x)$ is useless because it means the codeword is just the dataword augmented with $n - k$ zeros.

Two Isolated Single-Bit Errors

Now imagine there are two single-bit isolated errors. Under what conditions can this type of error be caught? We can show this type of error as $e(x) = x^j + x^i$. The values of i and j define the positions of the errors, and the difference $j - i$ defines the distance between the two errors, as shown in Figure 10.23.

Figure 10.23 *Representation of two isolated single-bit errors using polynomials*



We can write $e(x) = x^i(x^{j-i} + 1)$. If $g(x)$ has more than one term and one term is x^0, it cannot divide x^i, as we saw in the previous section. So if $g(x)$ is to divide $e(x)$, it must divide $x^{j-i} + 1$. In other words, $g(x)$ must not divide $x^t + 1$, where t is between 0 and $n - 1$. However, $t = 0$ is meaningless and $t = 1$ is needed as we will see later. This means t should be between 2 and $n - 1$.

If a generator cannot divide $x^t + 1$ (t between 0 and $n - 1$), then all isolated double errors can be detected.

Example 10.16

Find the status of the following generators related to two isolated, single-bit errors.

a. $x + 1$

b. $x^4 + 1$

c. $x^7 + x^6 + 1$

d. $x^{15} + x^{14} + 1$

Solution

a. This is a very poor choice for a generator. Any two errors next to each other cannot be detected.

b. This generator cannot detect two errors that are four positions apart. The two errors can be anywhere, but if their distance is 4, they remain undetected.

c. This is a good choice for this purpose.

d. This polynomial cannot divide any error of type $x^t + 1$ if t is less than 32,768. This means that a codeword with two isolated errors that are next to each other or up to 32,768 bits apart can be detected by this generator.

Odd Numbers of Errors

A generator with a factor of $x + 1$ can catch all odd numbers of errors. This means that we need to make $x + 1$ a factor of any generator. Note that we are not saying that the generator itself should be $x + 1$; we are saying that it should have a factor of $x + 1$. If it is only $x + 1$, it cannot catch the two adjacent isolated errors (see the previous section). For example, $x^4 + x^2 + x + 1$ can catch all odd-numbered errors since it can be written as a product of the two polynomials $x + 1$ and $x^3 + x^2 + 1$.

A generator that contains a factor of $x + 1$ can detect all odd-numbered errors.

Burst Errors

Now let us extend our analysis to the burst error, which is the most important of all. A burst error is of the form $e(x) = (x^j + \cdots + x^i)$. Note the difference between a burst error and two isolated single-bit errors. The first can have two terms or more; the second can only have two terms. We can factor out x^i and write the error as $x^i(x^{j-i} + \cdots + 1)$. If our generator can detect a single error (minimum condition for a generator), then it cannot divide x^i. What we should worry about are those generators that divide $x^{j-i} + \cdots + 1$. In other words, the remainder of $(x^{j-i} + \cdots + 1)/(x^r + \cdots + 1)$ must not be zero. Note that the denominator is the generator polynomial. We can have three cases:

1. If $j - i < r$, the remainder can never be zero. We can write $j - i = L - 1$, where L is the length of the error. So $L - 1 < r$ or $L < r + 1$ or $L \leq r$. This means all burst errors with length smaller than or equal to the number of check bits r will be detected.
2. In some rare cases, if $j - i = r$, or $L = r + 1$, the syndrome is 0 and the error is undetected. It can be proved that in these cases, the probability of undetected burst error of length $r + 1$ is $(1/2)^{r-1}$. For example, if our generator is $x^{14} + x^3 + 1$, in which $r = 14$, a burst error of length $L = 15$ can slip by undetected with the probability of $(1/2)^{14-1}$ or almost 1 in 10,000.
3. In some rare cases, if $j - i > r$, or $L > r + 1$, the syndrome is 0 and the error is undetected. It can be proved that in these cases, the probability of undetected burst error of length greater than $r + 1$ is $(1/2)^r$. For example, if our generator is $x^{14} + x^3 + 1$, in which $r = 14$, a burst error of length greater than 15 can slip by undetected with the probability of $(1/2)^{14}$ or almost 1 in 16,000 cases.

- ❑ **All burst errors with $L \leq r$ will be detected.**
- ❑ **All burst errors with $L = r + 1$ will be detected with probability $1 - (1/2)^{r-1}$.**
- ❑ **All burst errors with $L > r + 1$ will be detected with probability $1 - (1/2)^r$.**

Example 10.17

Find the suitability of the following generators in relation to burst errors of different lengths.

a. $x^6 + 1$

b. $x^{18} + x^7 + x + 1$

c. $x^{32} + x^{23} + x^7 + 1$

Solution

a. This generator can detect all burst errors with a length less than or equal to 6 bits; 3 out of 100 burst errors with length 7 will slip by; 16 out of 1000 burst errors of length 8 or more will slip by.

b. This generator can detect all burst errors with a length less than or equal to 18 bits; 8 out of 1 million burst errors with length 19 will slip by; 4 out of 1 million burst errors of length 20 or more will slip by.

c. This generator can detect all burst errors with a length less than or equal to 32 bits; 5 out of 10 billion burst errors with length 33 will slip by; 3 out of 10 billion burst errors of length 34 or more will slip by.

Summary

We can summarize the criteria for a good polynomial generator:

A good polynomial generator needs to have the following characteristics:

1. **It should have at least two terms.**
2. **The coefficient of the term x^0 should be 1.**
3. **It should not divide $x^t + 1$, for t between 2 and $n - 1$.**
4. **It should have the factor $x + 1$.**

Standard Polynomials

Some standard polynomials used by popular protocols for CRC generation are shown in Table 10.7.

Table 10.7 *Standard polynomials*

Name	*Polynomial*	*Application*
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

Advantages of Cyclic Codes

We have seen that cyclic codes have a very good performance in detecting single-bit errors, double errors, an odd number of errors, and burst errors. They can easily be implemented in hardware and software. They are especially fast when implemented in hardware. This has made cyclic codes a good candidate for many networks.

Other Cyclic Codes

The cyclic codes we have discussed in this section are very simple. The check bits and syndromes can be calculated by simple algebra. There are, however, more powerful polynomials that are based on abstract algebra involving Galois fields. These are beyond

the scope of this book. One of the most interesting of these codes is the **Reed-Solomon code** used today for both detection and correction.

10.5 CHECKSUM

The last error detection method we discuss here is called the **checksum.** The checksum is used in the Internet by several protocols although not at the data link layer. However, we briefly discuss it here to complete our discussion on error checking.

Like linear and cyclic codes, the checksum is based on the concept of redundancy. Several protocols still use the checksum for error detection as we will see in future chapters, although the tendency is to replace it with a CRC. This means that the CRC is also used in layers other than the data link layer.

Idea

The concept of the checksum is not difficult. Let us illustrate it with a few examples.

Example 10.18

Suppose our data is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, **36**), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the data are not accepted.

Example 10.19

We can make the job of the receiver easier if we send the negative (complement) of the sum, called the *checksum*. In this case, we send (7, 11, 12, 0, 6, **−36**). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

One's Complement

The previous example has one major drawback. All of our data can be written as a 4-bit word (they are less than 15) except for the checksum. One solution is to use **one's complement** arithmetic. In this arithmetic, we can represent unsigned numbers between 0 and $2^n - 1$ using only n bits.[†] If the number has more than n bits, the extra leftmost bits need to be added to the n rightmost bits (wrapping). In one's complement arithmetic, a negative number can be represented by inverting all bits (changing a 0 to a 1 and a 1 to a 0). This is the same as subtracting the number from $2^n - 1$.

Example 10.20

How can we represent the number 21 in one's complement arithmetic using only four bits?

[†]Although one's complement can represent both positive and negative numbers, we are concerned only with unsigned representation here.

Solution

The number 21 in binary is 10101 (it needs five bits). We can wrap the leftmost bit and add it to the four rightmost bits. We have (0101 + 1) = 0110 or 6.

Example 10.21

How can we represent the number −6 in one's complement arithmetic using only four bits?

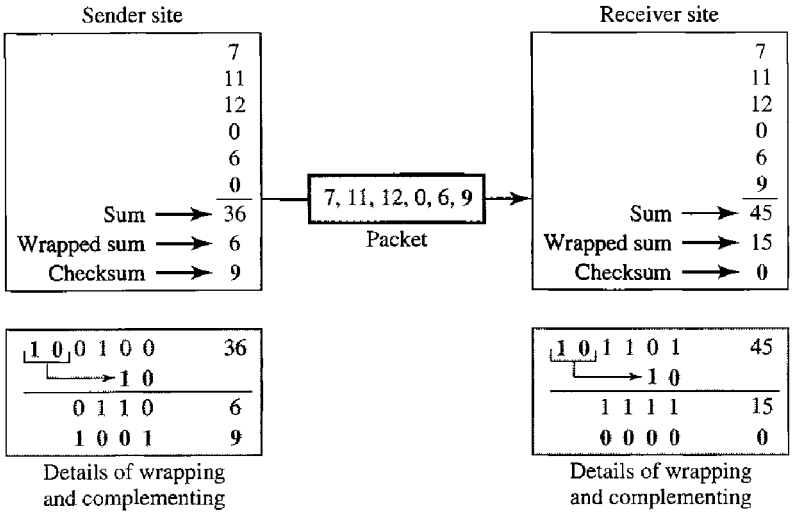
Solution

In one's complement arithmetic, the negative or complement of a number is found by inverting all bits. Positive 6 is 0110; negative 6 is 1001. If we consider only unsigned numbers, this is 9. In other words, the complement of 6 is 9. Another way to find the complement of a number in one's complement arithmetic is to subtract the number from $2^n - 1$ ($16 - 1$ in this case).

Example 10.22

Let us redo Exercise 10.19 using one's complement arithmetic. Figure 10.24 shows the process at the sender and at the receiver. The sender initializes the checksum to 0 and adds all data items and the checksum (the checksum is considered as one data item and is shown in color). The result is 36. However, 36 cannot be expressed in 4 bits. The extra two bits are wrapped and added with the sum to create the wrapped sum value 6. In the figure, we have shown the details in binary. The sum is then complemented, resulting in the checksum value 9 ($15 - 6 = 9$). The sender now sends six data items to the receiver including the checksum 9. The receiver follows the same procedure as the sender. It adds all data items (including the checksum); the result is 45. The sum is wrapped and becomes 15. The wrapped sum is complemented and becomes 0. Since the value of the checksum is 0, this means that the data is not corrupted. The receiver drops the checksum and keeps the other data items. If the checksum is not zero, the entire packet is dropped.

Figure 10.24



Internet Checksum

Traditionally, the Internet has been using a 16-bit checksum. The sender calculates the checksum by following these steps.

Sender site:

1. **The message is divided into 16-bit words.**
2. **The value of the checksum word is set to 0.**
3. **All words including the checksum are added using one's complement addition.**
4. **The sum is complemented and becomes the checksum.**
5. **The checksum is sent with the data.**

The receiver uses the following steps for error detection.

Receiver site:

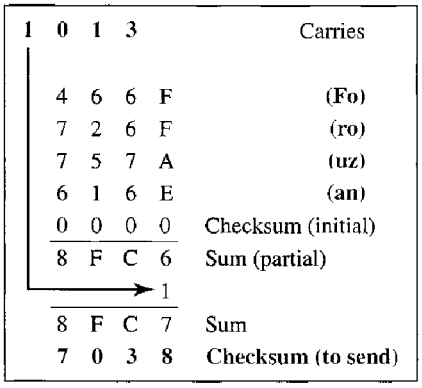
1. **The message (including checksum) is divided into 16-bit words.**
2. **All words are added using one's complement addition.**
3. **The sum is complemented and becomes the new checksum.**
4. **If the value of checksum is 0, the message is accepted; otherwise, it is rejected.**

The nature of the checksum (treating words as numbers and adding and complementing them) is well-suited for software implementation. Short programs can be written to calculate the checksum at the receiver site or to check the validity of the message at the receiver site.

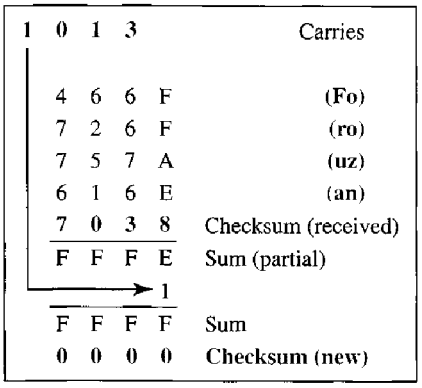
Example 10.23

Let us calculate the checksum for a text of 8 characters ("Forouzan"). The text needs to be divided into 2-byte (16-bit) words. We use ASCII (see Appendix A) to change each byte to a 2-digit hexadecimal number. For example, F is represented as 0x46 and o is represented as 0x6F. Figure 10.25 shows how the checksum is calculated at the sender and receiver sites. In part a of the figure, the value of partial sum for the first column is 0x36. We keep the rightmost digit (6) and insert the

Figure 10.25



a. Checksum at the sender site



b. Checksum at the receiver site

leftmost dight (3) as the carry in the second column. The process is repeated for each column. Hexadecimal numbers are reviewed in Appendix B.

Note that if there is any corruption, the checksum recalculated by the receiver is not all 0s. We leave this an exercise.

Performance

The traditional checksum uses a small number of bits (16) to detect errors in a message of any size (sometimes thousands of bits). However, it is not as strong as the CRC in error-checking capability. For example, if the value of one word is incremented and the value of another word is decremented by the same amount, the two errors cannot be detected because the sum and checksum remain the same. Also if the values of several words are incremented but the total change is a multiple of 65535, the sum and the checksum does not change, which means the errors are not detected. Fletcher and Adler have proposed some weighted checksums, in which each word is multiplied by a number (its weight) that is related to its position in the text. This will eliminate the first problem we mentioned. However, the tendency in the Internet, particularly in designing new protocols, is to replace the checksum with a CRC.

10.6 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

Several excellent book are devoted to error coding. Among them we recommend [Ham80], [Zar02], [Ror96], and [SWE04].

RFCs

A discussion of the use of the checksum in the Internet can be found in RFC 1141.

10.7 KEY TERMS

block code
burst error
check bit
checksum
codeword
convolution code
cyclic code
cyclic redundancy check (CRC)
dataword
error
error correction
error detection
forward error correction
generator polynomial
Hamming code
Hamming distance
interference
linear block code
minimum Hamming distance
modular arithmetic

modulus
one's complement
parity bit
parity-check code
polynomial
redundancy
Reed-Solomon
register
retransmission
shift register
single-bit error
syndrome
two-dimensional parity check

10.8 SUMMARY

- Data can be corrupted during transmission. Some applications require that errors be detected and corrected.
- In a single-bit error, only one bit in the data unit has changed. A burst error means that two or more bits in the data unit have changed.
- To detect or correct errors, we need to send extra (redundant) bits with data.
- There are two main methods of error correction: forward error correction and correction by retransmission.
- We can divide coding schemes into two broad categories: *block coding* and *convolution* coding.
- In coding, we need to use modulo-2 arithmetic. Operations in this arithmetic are very simple; addition and subtraction give the same results. we use the XOR (exclusive OR) operation for both addition and subtraction.
- In block coding, we divide our message into blocks, each of k bits, called datawords. We add r redundant bits to each block to make the length $n = k + r$. The resulting n-bit blocks are called codewords.
- In block coding, errors be detected by using the following two conditions:
  a. The receiver has (or can find) a list of valid codewords.
  b. The original codeword has changed to an invalid one.
- The Hamming distance between two words is the number of differences between corresponding bits. The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.
- To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$. To guarantee correction of up to t errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.
- In a linear block code, the exclusive OR (XOR) of any two valid codewords creates another valid codeword.
- A simple parity-check code is a single-bit error-detecting code in which $n = k + 1$ with $d_{min} = 2$. A simple parity-check code can detect an odd number of errors.
- All Hamming codes discussed in this book have $d_{min} = 3$. The relationship between m and n in these codes is $n = 2m - 1$.
- Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.

- A category of cyclic codes called the cyclic redundancy check (CRC) is used in networks such as LANs and WANs.
- A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1.
- Traditionally, the Internet has been using a 16-bit checksum, which uses *one's complement* arithmetic. In this arithmetic, we can represent unsigned numbers between 0 and $2^n - 1$ using only n bits.

10.9 PRACTICE SET

Review Questions

1. How does a single-bit error differ from a burst error?
2. Discuss the concept of redundancy in error detection and correction.
3. Distinguish between forward error correction versus error correction by retransmission.
4. What is the definition of a linear block code? What is the definition of a cyclic code?
5. What is the Hamming distance? What is the minimum Hamming distance?
6. How is the simple parity check related to the two-dimensional parity check?
7. In CRC, show the relationship between the following entities (size means the number of bits):
   a. The size of the dataword and the size of the codeword
   b. The size of the divisor and the remainder
   c. The degree of the polynomial generator and the size of the divisor
   d. The degree of the polynomial generator and the size of the remainder
8. What kind of arithmetic is used to add data items in checksum calculation?
9. What kind of error is undetectable by the checksum?
10. Can the value of a checksum be all 0s (in binary)? Defend your answer. Can the value be all 1s (in binary)? Defend your answer.

Exercises

11. What is the maximum effect of a 2-ms burst of noise on data transmitted at the following rates?
    a. 1500 bps
    b. 12 kbps
    c. 100 kbps
    d. 100 Mbps
12. Apply the exclusive-or operation on the following pair of patterns (the symbol $\oplus$ means XOR):
    a. (10001) $\oplus$ (10000)
    b. (10001) $\oplus$ (10001) (What do you infer from the result?)
    c. (11100) $\oplus$ (00000) (What do you infer from the result?)
    d. (10011) $\oplus$ (11111) (What do you infer from the result?)

13. In Table 10.1, the sender sends dataword 10. A 3-bit burst error corrupts the codeword. Can the receiver detect the error? Defend your answer.
14. In Table 10.2, the sender sends dataword 10. If a 3-bit burst error corrupts the first three bits of the codeword, can the receiver detect the error? Defend your answer.
15. What is the Hamming distance for each of the following codewords:
    a. d (10000, 00000)
    b. d (10101, 10000)
    c. d (11111, 11111)
    d. d (000, 000)
16. Find the minimum Hamming distance for the following cases:
    a. Detection of two errors.
    b. Correction of two errors.
    c. Detection of 3 errors or correction of 2 errors.
    d. Detection of 6 errors or correction of 2 errors.
17. Using the code in Table 10.2, what is the dataword if one of the following codewords is received?
    a. 01011
    b. 11111
    c. 00000
    d. 11011
18. Prove that the code represented by Table 10.8 is not a linear code. You need to find only one case that violates the linearity.

Table 10.8 *Table for Exercise 18*

Dataword	*Codeword*
00	00000
01	01011
10	10111
11	11111

19. Although it can mathematically be proved that a simple parity check code is a linear code, use manual testing of linearity for five pairs of the codewords in Table 10.3 to partially prove this fact.
20. Show that the Hamming code C(7,4) of Table 10.4 can detect two-bit errors but not necessarily three-bit error by testing the code in the following cases. The character "∇" in the burst error means no error; the character "**E**" means an error.
    a. Dataword: 0100 Burst error: ∇**E**∇∇∇∇
    b. Dataword: 0111 Burst error: **E**∇∇∇∇∇**E**
    c. Dataword: 1111 Burst error: **E**∇**E**∇∇∇**E**
    d. Dataword: 0000 Burst error: **EE**∇**E**∇∇∇

21. Show that the Hamming code C(7,4) of Table 10.4 can correct one-bit errors but not more by testing the code in the following cases. The character "∇" in the burst error means no error; the character "**E**" means an error.
    a. Dataword: 0100 Burst error: **E**∇∇∇∇∇∇
    b. Dataword: 0111 Burst error: ∇**E**∇∇∇∇∇
    c. Dataword: 1111 Burst error: **E**∇∇∇∇∇**E**
    d. Dataword: 0000 Burst error: **EE**∇∇∇∇**E**
22. Although it can be proved that code in Table 10.6 is both linear and cyclic, use only two tests to partially prove the fact:
    a. Test the cyclic property on codeword 0101100.
    b. Test the linear property on codewords 0010110 and 1111111.
23. We need a dataword of at least 11 bits. Find the values of k and n in the Hamming code C(n, k) with $d_{min} = 3$.
24. Apply the following operations on the corresponding polynomials:
    a. $(x^3 + x^2 + x + 1) + (x^4 + x^2 + x + 1)$
    b. $(x^3 + x^2 + x + 1) - (x^4 + x^2 + x + 1)$
    c. $(x^3 + x^2) \times (x^4 + x^2 + x + 1)$
    d. $(x^3 + x^2 + x + 1) / (x^2 + 1)$
25. Answer the following questions:
    a. What is the polynomial representation of 101110?
    b. What is the result of shifting 101110 three bits to the left?
    c. Repeat part b using polynomials.
    d. What is the result of shifting 101110 four bits to the right?
    e. Repeat part d using polynomials.
26. Which of the following CRC generators guarantee the detection of a single bit error?
    a. $x^3 + x + 1$
    b. $x^4 + x^2$
    c. 1
    d. $x^2 + 1$
27. Referring to the CRC-8 polynomial in Table 10.7, answer the following questions:
    a. Does it detect a single error? Defend your answer.
    b. Does it detect a burst error of size 6? Defend your answer.
    c. What is the probability of detecting a burst error of size 9?
    d. What is the probability of detecting a burst error of size 15?
28. Referring to the CRC-32 polynomial in Table 10.7, answer the following questions:
    a. Does it detect a single error? Defend your answer.
    b. Does it detect a burst error of size 16? Defend your answer.
    c. What is the probability of detecting a burst error of size 33?
    d. What is the probability of detecting a burst error of size 55?

29. Assuming even parity, find the parity bit for each of the following data units.
    a. 1001011
    b. 0001100
    c. 1000000
    d. 1110111
30. Given the dataword 1010011110 and the divisor 10111,
    a. Show the generation of the codeword at the sender site (using binary division).
    b. Show the checking of the codeword at the receiver site (assume no error).
31. Repeat Exercise 30 using polynomials.
32. A sender needs to send the four data items 0x3456, 0xABCC, 0x02BC, and 0xEEEE. Answer the following:
    a. Find the checksum at the sender site.
    b. Find the checksum at the receiver site if there is no error.
    c. Find the checksum at the receiver site if the second data item is changed to 0xABCE.
    d. Find the checksum at the receiver site if the second data item is changed to 0xABCE and the third data item is changed to 0x02BA.
33. This problem shows a special case in checksum handling. A sender has two data items to send: 0x4567 and 0xBA98. What is the value of the checksum?

CHAPTER 11

Data Link Control

The two main functions of the data link layer are data link control and media access control. The first, data link control, deals with the design and procedures for communication between two adjacent nodes: node-to-node communication. We discuss this functionality in this chapter. The second function of the data link layer is media access control, or how to share the link. We discuss this functionality in Chapter 12.

Data link control functions include framing, flow and error control, and software-implemented protocols that provide smooth and reliable transmission of frames between nodes. In this chapter, we first discuss framing, or how to organize the bits that are carried by the physical layer. We then discuss flow and error control. A subset of this topic, techniques for error detection and correction, was discussed in Chapter 10.

To implement data link control, we need protocols. Each protocol is a set of rules that need to be implemented in software and run by the two nodes involved in data exchange at the data link layer. We discuss five protocols: two for noiseless (ideal) channels and three for noisy (real) channels. Those in the first category are not actually implemented, but provide a foundation for understanding the protocols in the second category.

After discussing the five protocol designs, we show how a bit-oriented protocol is actually implemented by using the High-level Data Link Control (HDLC) Protocol as an example. We also discuss a popular byte-oriented protocol, Point-to-Point Protocol (PPP).

11.1 FRAMING

Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination. The physical layer provides bit synchronization to ensure that the sender and receiver use the same bit durations and timing.

The data link layer, on the other hand, needs to pack bits into frames, so that each frame is distinguishable from another. Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter. In addition, each envelope defines the sender and receiver addresses since the postal system is a many-to-many carrier facility.

Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address. The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.

Although the whole message could be packed in one frame, that is not normally done. One reason is that a frame can be very large, making flow and error control very inefficient. When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole message. When a message is divided into smaller frames, a single-bit error affects only that small frame.

Fixed-Size Framing

Frames can be of fixed or variable size. In **fixed-size framing,** there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. An example of this type of framing is the ATM wide-area network, which uses frames of fixed size called cells. We discuss ATM in Chapter 18.

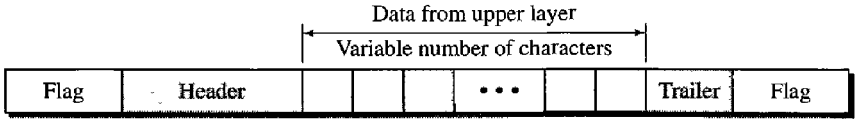
Variable-Size Framing

Our main discussion in this chapter concerns **variable-size framing,** prevalent in local-area networks. In variable-size framing, we need a way to define the end of the frame and the beginning of the next. Historically, two approaches were used for this purpose: a character-oriented approach and a bit-oriented approach.

Character-Oriented Protocols

In a **character-oriented protocol,** data to be carried are 8-bit characters from a coding system such as ASCII (see Appendix A). The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection or error correction redundant bits, are also multiples of 8 bits. To separate one frame from the next, an 8-bit (1-byte) **flag** is added at the beginning and the end of a frame. The flag, composed of protocol-dependent special characters, signals the start or end of a frame. Figure 11.1 shows the format of a frame in a character-oriented protocol.

Figure 11.1 *A frame in a character-oriented protocol*

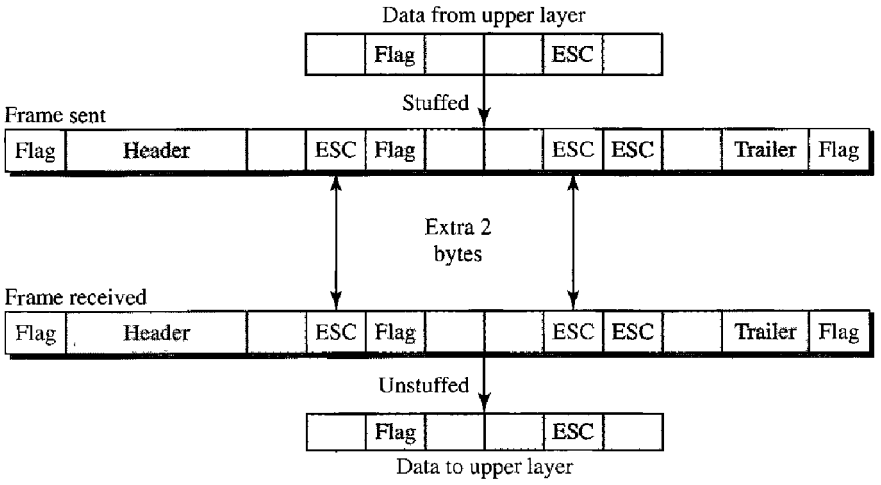


Character-oriented framing was popular when only text was exchanged by the data link layers. The flag could be selected to be any character not used for text communication. Now, however, we send other types of information such as graphs, audio, and video. Any pattern used for the flag could also be part of the information. If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame. To fix this problem, a **byte-stuffing** strategy was added to

character-oriented framing. In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the **escape character (ESC),** which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag.

Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. What happens if the text contains one or more escape characters followed by a flag? The receiver removes the escape character, but keeps the flag, which is incorrectly interpreted as the end of the frame. To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words, if the escape character is part of the text, an extra one is added to show that the second one is part of the text. Figure 11.2 shows the situation.

Figure 11.2 *Byte stuffing and unstuffing*



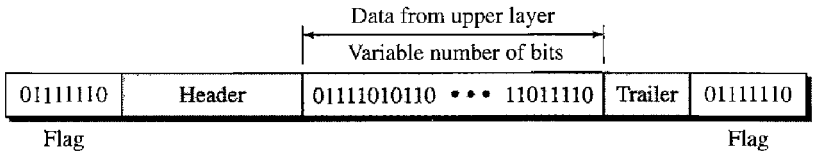
Byte stuffing is the process of adding 1 extra byte whenever there is a flag or escape character in the text.

Character-oriented protocols present another problem in data communications. The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters. We can say that in general, the tendency is moving toward the bit-oriented protocols that we discuss next.

Bit-Oriented Protocols

In a **bit-oriented protocol,** the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame, as shown in Figure 11.3.

Figure 11.3 *A frame in a bit-oriented protocol*

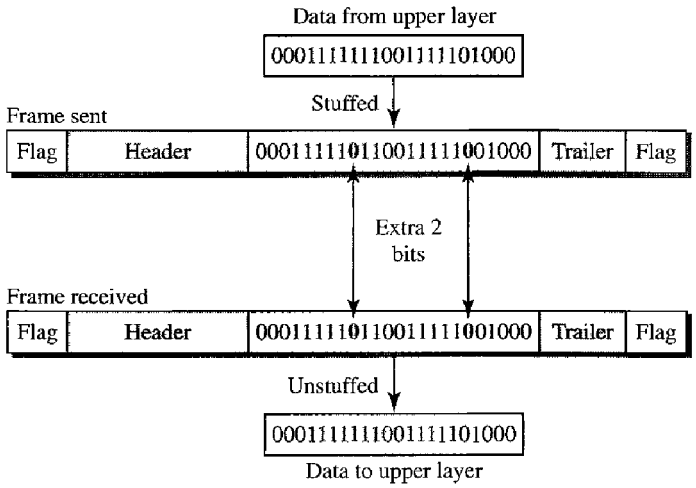


This flag can create the same type of problem we saw in the byte-oriented protocols. That is, if the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame. We do this by stuffing 1 single bit (instead of 1 byte) to prevent the pattern from looking like a flag. The strategy is called **bit stuffing.** In bit stuffing, if a 0 and five consecutive 1 bits are encountered, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver. Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. This guarantees that the flag field sequence does not inadvertently appear in the frame.

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

Figure 11.4 shows bit stuffing at the sender and bit removal at the receiver. Note that even if we have a 0 after five 1s, we still stuff a 0. The 0 will be removed by the receiver.

Figure 11.4 *Bit stuffing and unstuffing*



This means that if the flaglike pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken as a flag by the receiver. The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

11.2 FLOW AND ERROR CONTROL

Data communication requires at least two devices working together, one to send and the other to receive. Even such a basic arrangement requires a great deal of coordination for an intelligible exchange to occur. The most important responsibilities of the data link layer are **flow control** and **error control.** Collectively, these functions are known as **data link control.**

Flow Control

Flow control coordinates the amount of data that can be sent before receiving an acknowledgment and is one of the most important duties of the data link layer. In most protocols, flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgment from the receiver. The flow of data must not be allowed to overwhelm the receiver. Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data. The receiving device must be able to inform the sending device before those limits are reached and to request that the transmitting device send fewer frames or stop temporarily. Incoming data must be checked and processed before they can be used. The rate of such processing is often slower than the rate of transmission. For this reason, each receiving device has a block of memory, called a *buffer,* reserved for storing incoming data until they are processed. If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

Error Control

Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term *error control* refers primarily to methods of error detection and retransmission. Error control in the data link layer is often implemented simply: Any time an error is detected in an exchange, specified frames are retransmitted. This process is called **automatic repeat request (ARQ).**

Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.

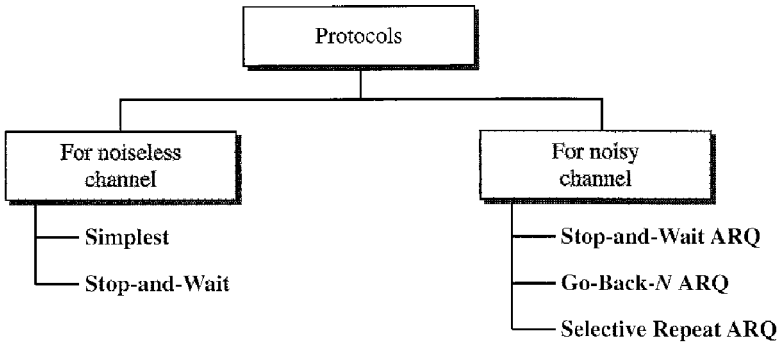
11.3 PROTOCOLS

Now let us see how the data link layer can combine framing, flow control, and error control to achieve the delivery of data from one node to another. The protocols are normally implemented in software by using one of the common programming languages. To make our

discussions language-free, we have written in pseudocode a version of each protocol that concentrates mostly on the procedure instead of delving into the details of language rules.

We divide the discussion of protocols into those that can be used for noiseless (error-free) channels and those that can be used for noisy (error-creating) channels. The protocols in the first category cannot be used in real life, but they serve as a basis for understanding the protocols of noisy channels. Figure 11.5 shows the classifications.

Figure 11.5 *Taxonomy of protocols discussed in this chapter*



There is a difference between the protocols we discuss here and those used in real networks. All the protocols we discuss are unidirectional in the sense that the data frames travel from one node, called the sender, to another node, called the receiver. Although special frames, called **acknowledgment (ACK)** and **negative acknowledgment (NAK)** can flow in the opposite direction for flow and error control purposes, data flow in only one direction.

In a real-life network, the data link protocols are implemented as bidirectional; data flow in both directions. In these protocols the flow and error control information such as ACKs and NAKs is included in the data frames in a technique called **piggybacking.** Because bidirectional protocols are more complex than unidirectional ones, we chose the latter for our discussion. If they are understood, they can be extended to bidirectional protocols. We leave this extension as an exercise.

11.4 NOISELESS CHANNELS

Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted. We introduce two protocols for this type of channel. The first is a protocol that does not use flow control; the second is the one that does. Of course, neither has error control because we have assumed that the channel is a perfect **noiseless channel.**

Simplest Protocol

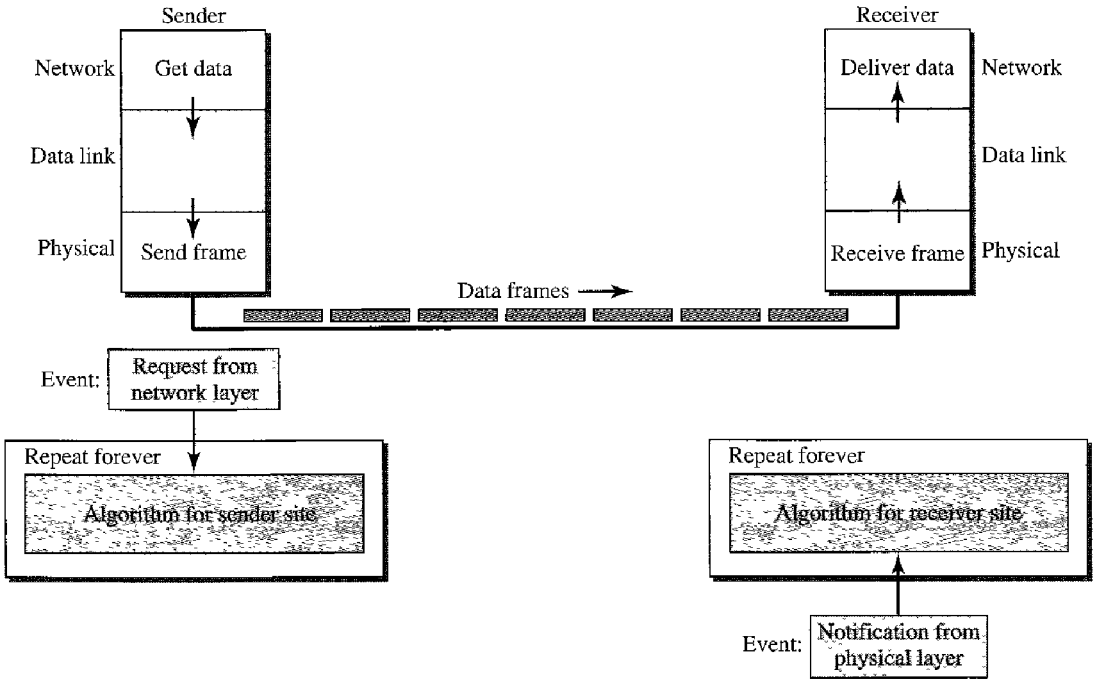
Our first protocol, which we call the **Simplest Protocol** for lack of any other name, is one that has no flow or error control. Like other protocols we will discuss in this chapter, it is a unidirectional protocol in which data frames are traveling in only one direction—from the

sender to receiver. We assume that the receiver can immediately handle any frame it receives with a processing time that is small enough to be negligible. The data link layer of the receiver immediately removes the header from the frame and hands the data packet to its network layer, which can also accept the packet immediately. In other words, the receiver can never be overwhelmed with incoming frames.

Design

There is no need for flow control in this scheme. The data link layer at the sender site gets data from its network layer, makes a frame out of the data, and sends it. The data link layer at the receiver site receives a frame from its physical layer, extracts data from the frame, and delivers the data to its network layer. The data link layers of the sender and receiver provide transmission services for their network layers. The data link layers use the services provided by their physical layers (such as signaling, multiplexing, and so on) for the physical transmission of bits. Figure 11.6 shows a design.

Figure 11.6 *The design of the simplest protocol with no flow or error control*



We need to elaborate on the procedure used by both data link layers. The sender site cannot send a frame until its network layer has a data packet to send. The receiver site cannot deliver a data packet to its network layer until a frame arrives. If the protocol is implemented as a procedure, we need to introduce the idea of **events** in the protocol. The procedure at the sender site is constantly running; there is no action until there is a request from the network layer. The procedure at the receiver site is also constantly running, but there is no action until notification from the physical layer arrives. Both procedures are constantly running because they do not know when the corresponding events will occur.

Algorithms

Algorithm 11.1 shows the procedure at the sender site.

Algorithm 11.1 ***Sender-site algorithm for the simplest protocol***

```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(RequestToSend))           //There is a packet to send
5    {
6       GetData();
7       MakeFrame();
8       SendFrame();                    //Send the frame
9    }
10 }
```

Analysis The algorithm has an infinite loop, which means lines 3 to 9 are repeated forever once the program starts. The algorithm is an event-driven one, which means that it *sleeps* (line 3) until an event *wakes* it *up* (line 4). This means that there may be an undefined span of time between the execution of line 3 and line 4; there is a gap between these actions. When the event, a request from the network layer, occurs, lines 6 though 8 are executed. The program then repeats the loop and again sleeps at line 3 until the next occurrence of the event. We have written pseudocode for the main process. We do not show any details for the modules GetData, MakeFrame, and SendFrame. GetData() takes a data packet from the network layer, MakeFrame() adds a header and delimiter flags to the data packet to make a frame, and SendFrame() delivers the frame to the physical layer for transmission.

Algorithm 11.2 shows the procedure at the receiver site.

Algorithm 11.2 ***Receiver-site algorithm for the simplest protocol***

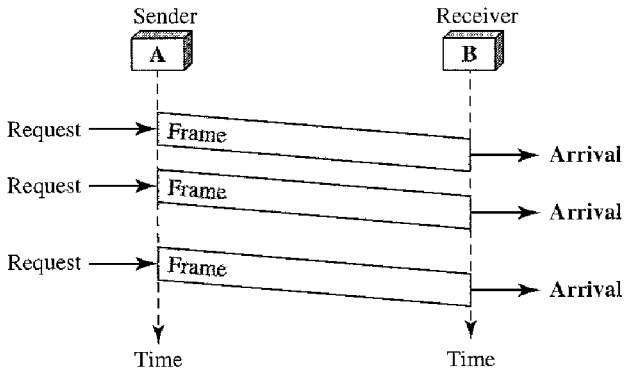
```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(ArrivalNotification))     //Data frame arrived
5    {
6       ReceiveFrame();
7       ExtractData();
8       DeliverData();                  //Deliver data to network layer
9    }
10 }
```

Analysis This algorithm has the same format as Algorithm 11.1, except that the direction of the frames and data is upward. The event here is the arrival of a data frame. After the event occurs, the data link layer receives the frame from the physical layer using the ReceiveFrame() process, extracts the data from the frame using the ExtractData() process, and delivers the data to the network layer using the DeliverData() process. Here, we also have an event-driven algorithm because the algorithm never knows when the data frame will arrive.

Example 11.1

Figure 11.7 shows an example of communication using this protocol. It is very simple. The sender sends a sequence of frames without even thinking about the receiver. To send three frames, three events occur at the sender site and three events at the receiver site. Note that the data frames are shown by tilted boxes; the height of the box defines the transmission time difference between the first bit and the last bit in the frame.

Figure 11.7 *Flow diagram for Example 11.1*



Stop-and-Wait Protocol

If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use. Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service. To prevent the receiver from becoming overwhelmed with frames,we somehow need to tell the sender to slow down. There must be feedback from the receiver to the sender.

The protocol we discuss now is called the **Stop-and-Wait Protocol** because the sender sends one frame, stops until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame. We still have unidirectional communication for data frames, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction. We add flow control to our previous protocol.

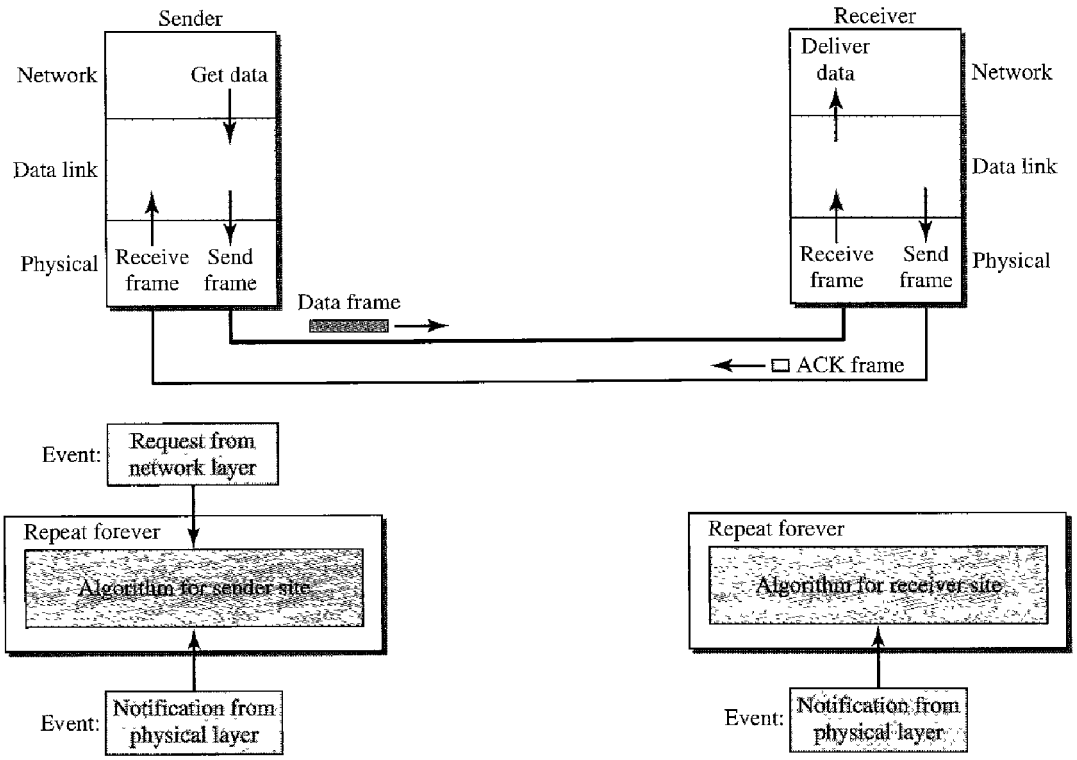
Design

Figure 11.8 illustrates the mechanism. Comparing this figure with Figure 11.6, we can see the traffic on the forward channel (from sender to receiver) and the reverse channel. At any time, there is either one data frame on the forward channel or one ACK frame on the reverse channel. We therefore need a half-duplex link.

Algorithms

Algorithm 11.3 is for the sender site.

Figure 11.8 *Design of Stop-and-Wait Protocol*



Algorithm 11.3 *Sender-site algorithm for Stop-and-Wait Protocol*

```
1  while(true)                          //Repeat forever
2  canSend = true                       //Allow the first frame to go
3  {
4    WaitForEvent();                    // Sleep until an event occurs
5    if(Event(RequestToSend) AND canSend)
6    {
7       GetData();
8       MakeFrame();
9       SendFrame();                    //Send the data frame
10      canSend = false;                //Cannot send until ACK arrives
11   }
12   WaitForEvent();                    // Sleep until an event occurs
13   if(Event(ArrivalNotification) // An ACK has arrived
14    {
15      ReceiveFrame();                 //Receive the ACK frame
16      canSend = true;
17    }
18 }
```

Analysis Here two events can occur: a request from the network layer or an arrival notification from the physical layer. The responses to these events must alternate. In other words, after a frame is sent, the algorithm must ignore another network layer request until that frame is

acknowledged. We know that two arrival events cannot happen one after another because the channel is error-free and does not duplicate the frames. The requests from the network layer, however, may happen one after another without an arrival event in between. We need somehow to prevent the immediate sending of the data frame. Although there are several methods, we have used a simple *canSend* variable that can either be true or false. When a frame is sent, the variable is set to false to indicate that a new network request cannot be sent until *canSend* is true. When an ACK is received, canSend is set to true to allow the sending of the next frame.

Algorithm 11.4 shows the procedure at the receiver site.

Algorithm 11.4 *Receiver-site algorithm for Stop-and-Wait Protocol*

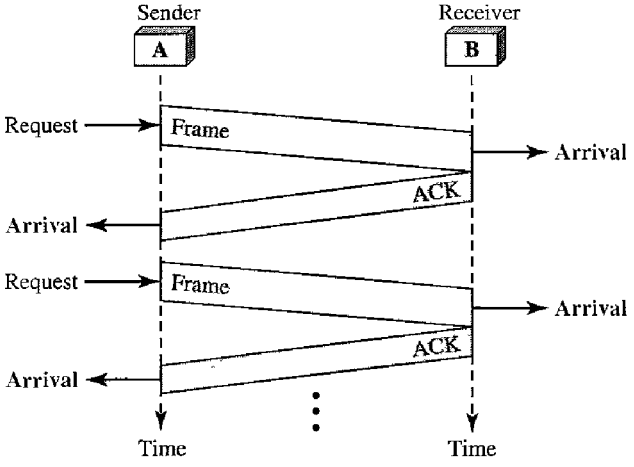
```
1  while(true)                          //Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(ArrivalNotification))     //Data frame arrives
5    {
6       ReceiveFrame();
7       ExtractData();
8       Deliver(data);                  //Deliver data to network layer
9       SendFrame();                    //Send an ACK frame
10   }
11 }
```

Analysis This is very similar to Algorithm 11.2 with one exception. After the data frame arrives, the receiver sends an ACK frame (line 9) to acknowledge the receipt and allow the sender to send the next frame.

Example 11.2

Figure 11.9 shows an example of communication using this protocol. It is still very simple. The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame. Note that sending two frames in the protocol involves the sender in four events and the receiver in two events.

Figure 11.9 *Flow diagram for Example 11.2*



11.5 NOISY CHANNELS

Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent. We can ignore the error (as we sometimes do), or we need to add error control to our protocols. We discuss three protocols in this section that use error control.

Stop-and-Wait Automatic Repeat Request

Our first protocol, called the **Stop-and-Wait Automatic Repeat Request (Stop-and-Wait ARQ),** adds a simple error control mechanism to the Stop-and-Wait Protocol. Let us see how this protocol detects and corrects errors.

To detect and correct corrupted frames, we need to add redundancy bits to our data frame (see Chapter 10). When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded. The detection of errors in this protocol is manifested by the silence of the receiver.

Lost frames are more difficult to handle than corrupted ones. In our previous protocols, there was no way to identify a frame. The received frame could be the correct one, or a duplicate, or a frame out of order. The solution is to number the frames. When the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated.

The corrupted and lost frames need to be resent in this protocol. If the receiver does not respond when there is an error, how can the sender know which frame to resend? To remedy this problem, the sender keeps a copy of the sent frame. At the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK even though several copies of the same frame can be in the network.

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

Since an ACK frame can also be corrupted and lost, it too needs redundancy bits and a sequence number. The ACK frame for this protocol has a sequence number field. In this protocol, the sender simply discards a corrupted ACK frame or ignores an out-of-order one.

Sequence Numbers

As we discussed, the protocol specifies that frames need to be numbered. This is done by using **sequence numbers.** A field is added to the data frame to hold the sequence number of that frame.

One important consideration is the range of the sequence numbers. Since we want to minimize the frame size, we look for the smallest range that provides unambiguous

communication. The sequence numbers of course can wrap around. For example, if we decide that the field is m bits long, the sequence numbers start from 0, go to $2^m - 1$, and then are repeated.

Let us reason out the range of sequence numbers we need. Assume we have used x as a sequence number; we only need to use $x + 1$ after that. There is no need for $x + 2$. To show this, assume that the sender has sent the frame numbered x. Three things can happen.

1. The frame arrives safe and sound at the receiver site; the receiver sends an acknowledgment. The acknowledgment arrives at the sender site, causing the sender to send the next frame numbered $x + 1$.
2. The frame arrives safe and sound at the receiver site; the receiver sends an acknowledgment, but the acknowledgment is corrupted or lost. The sender resends the frame (numbered x) after the time-out. Note that the frame here is a duplicate. The receiver can recognize this fact because it expects frame $x + 1$ but frame x was received.
3. The frame is corrupted or never arrives at the receiver site; the sender resends the frame (numbered x) after the time-out.

We can see that there is a need for sequence numbers x and $x + 1$ because the receiver needs to distinguish between case 1 and case 2. But there is no need for a frame to be numbered $x + 2$. In case 1, the frame can be numbered x again because frames x and $x + 1$ are acknowledged and there is no ambiguity at either site. In cases 2 and 3, the new frame is $x + 1$, not $x + 2$. If only x and $x + 1$ are needed, we can let $x = 0$ and $x + 1 = 1$. This means that the sequence is 0, 1, 0, 1, 0, and so on. Is this pattern familiar? This is modulo-2 arithmetic as we saw in Chapter 10.

In Stop-and-Wait ARQ, we use sequence numbers to number the frames. The sequence numbers are based on modulo-2 arithmetic.

Acknowledgment Numbers

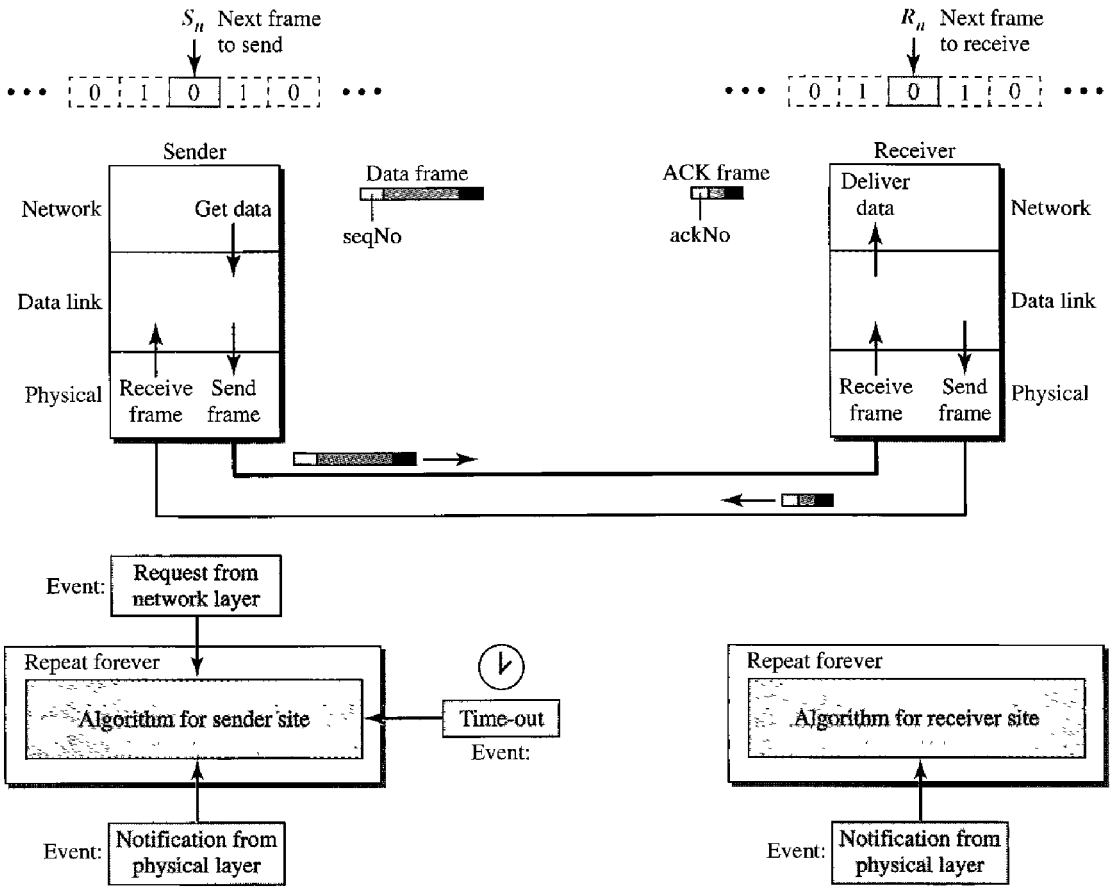
Since the sequence numbers must be suitable for both data frames and ACK frames, we use this convention: The acknowledgment numbers always announce the sequence number of the next frame expected by the receiver. For example, if frame 0 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 1 (meaning frame 1 is expected next). If frame 1 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 0 (meaning frame 0 is expected).

In Stop-and-Wait ARQ, the acknowledgment number always announces in modulo-2 arithmetic the sequence number of the next frame expected.

Design

Figure 11.10 shows the design of the Stop-and-Wait ARQ Protocol. The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame. A data frames uses a seqNo (sequence number); an ACK frame uses an ackNo (acknowledgment number). The sender has a control variable, which we call S_n (sender, next frame to send), that holds the sequence number for the next frame to be sent (0 or 1).

Figure 11.10 *Design of the Stop-and-Wait ARQ Protocol*



The receiver has a control variable, which we call R_n (receiver, next frame expected), that holds the number of the next frame expected. When a frame is sent, the value of S_n is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. When a frame is received, the value of R_n is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. Three events can happen at the sender site; one event can happen at the receiver site. Variable S_n points to the slot that matches the sequence number of the frame that has been sent, but not acknowledged; R_n points to the slot that matches the sequence number of the expected frame.

Algorithms

Algorithm 11.5 is for the sender site.

Algorithm 11.5 ***Sender-site algorithm for Stop-and-Wait ARQ***

```
1  Sn = 0;                          // Frame 0 should be sent first
2  canSend = true;                  // Allow the first request to go
3  while(true)                      // Repeat forever
4  {
5    WaitForEvent();                // Sleep until an event occurs
```

Algorithm 11.5 *Sender-site algorithm for Stop-and-Wait ARQ (continued)*

```
 6   if(Event(RequestToSend) AND canSend)
 7   {
 8      GetData();
 9      MakeFrame(Sn);                    //The seqNo is Sn
10      StoreFrame(Sn);                   //Keep copy
11      SendFrame(Sn);
12      StartTimer();
13      Sn = Sn + 1;
14      canSend = false;
15   }
16   WaitForEvent();                      // Sleep
17     if(Event(ArrivalNotification)      // An ACK has arrived
18     {
19       ReceiveFrame(ackNo);             //Receive the ACK frame
20       if(not corrupted AND ackNo == Sn) //Valid ACK
21         {
22           Stoptimer();
23           PurgeFrame(Sn-1);            //Copy is not needed
24           canSend = true;
25         }
26      }
27
28     if(Event(TimeOut)                  // The timer expired
29     {
30      StartTimer();
31      ResendFrame(Sn-1);                //Resend a copy check
32     }
33 }
```

Analysis We first notice the presence of S_n, the sequence number of the next frame to be sent. This variable is initialized once (line 1), but it is incremented every time a frame is sent (line 13) in preparation for the next frame. However, since this is modulo-2 arithmetic, the sequence numbers are 0, 1, 0, 1, and so on. Note that the processes in the first event (SendFrame, StoreFrame, and PurgeFrame) use an S_n defining the frame sent out. We need at least one buffer to hold this frame until we are sure that it is received safe and sound. Line 10 shows that before the frame is sent, it is stored. The copy is used for resending a corrupt or lost frame. We are still using the canSend variable to prevent the network layer from making a request before the previous frame is received safe and sound. If the frame is not corrupted and the ackNo of the ACK frame matches the sequence number of the next frame to send, we stop the timer and purge the copy of the data frame we saved. Otherwise, we just ignore this event and wait for the next event to happen. After each frame is sent, a timer is started. When the timer expires (line 28), the frame is resent and the timer is restarted.

Algorithm 11.6 shows the procedure at the receiver site.

Algorithm 11.6 *Receiver-site algorithm for Stop-and-Wait ARQ Protocol*

```
1  Rn = 0;                          // Frame 0 expected to arrive first
2  while(true)
3  {
4    WaitForEvent();                // Sleep until an event occurs
```

Algorithm 11.6 *Receiver-site algorithm for Stop-and-Wait ARQ Protocol (continued)*

```
5   if(Event(ArrivalNotification))  //Data frame arrives
6   {
7      ReceiveFrame();
8      if(corrupted(frame));
9         sleep();
10     if(seqNo == Rn)              //Valid data frame
11     {
12      ExtractData();
13       DeliverData();             //Deliver data
14       Rn = Rn + 1;
15     }
16      SendFrame(Rn);              //Send an ACK
17  }
18 }
```

Analysis This is noticeably different from Algorithm 11.4. First, all arrived data frames that are corrupted are ignored. If the seqNo of the frame is the one that is expected (R_n), the frame is accepted, the data are delivered to the network layer, and the value of R_n is incremented. However, there is one subtle point here. Even if the sequence number of the data frame does not match the next frame expected, an ACK is sent to the sender. This ACK, however, just reconfirms the previous ACK instead of confirming the frame received. This is done because the receiver assumes that the previous ACK might have been lost; the receiver is sending a duplicate frame. The resent ACK may solve the problem before the time-out does it.

Example 11.3

Figure 11.11 shows an example of Stop-and-Wait ARQ. Frame 0 is sent and acknowledged. Frame 1 is lost and resent after the time-out. The resent frame 1 is acknowledged and the timer stops. Frame 0 is sent and acknowledged, but the acknowledgment is lost. The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.

Efficiency

The Stop-and-Wait ARQ discussed in the previous section is very inefficient if our channel is *thick* and *long*. By *thick,* we mean that our channel has a large bandwidth; by *long,* we mean the round-trip delay is long. The product of these two is called the **bandwidth-delay product,** as we discussed in Chapter 3. We can think of the channel as a pipe. The bandwidth-delay product then is the volume of the pipe in bits. The pipe is always there. If we do not use it, we are inefficient. The bandwidth-delay product is a measure of the number of bits we can send out of our system while waiting for news from the receiver.

Example 11.4

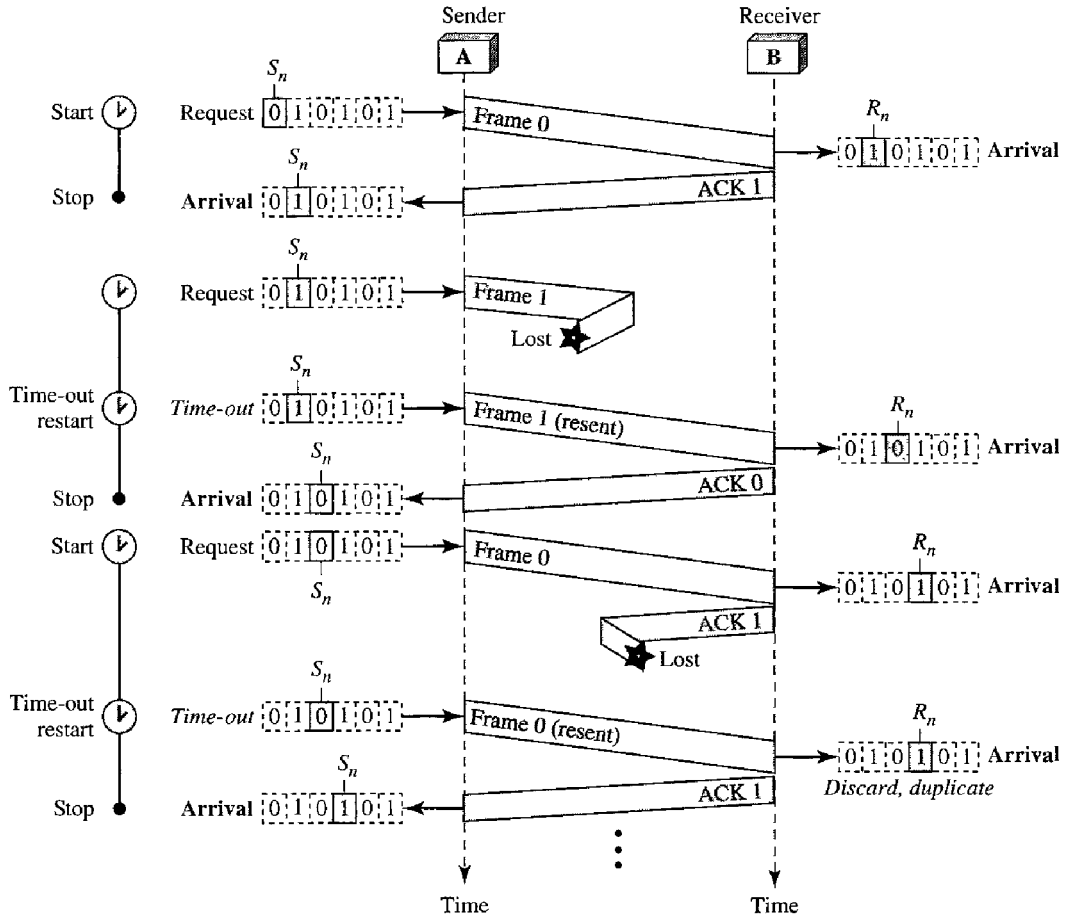
Assume that, in a Stop-and-Wait ARQ system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 ms to make a round trip. What is the bandwidth-delay product? If the system data frames are 1000 bits in length, what is the utilization percentage of the link?

Solution

The bandwidth-delay product is

$$(1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000 \text{ bits}$$

Figure 11.11 *Flow diagram for Example 11.3*



The system can send 20,000 bits during the time it takes for the data to go from the sender to the receiver and then back again. However, the system sends only 1000 bits. We can say that the link utilization is only 1000/20,000, or 5 percent. For this reason, for a link with a high bandwidth or long delay, the use of Stop-and-Wait ARQ wastes the capacity of the link.

Example 11.5

What is the utilization percentage of the link in Example 11.4 if we have a protocol that can send up to 15 frames before stopping and worrying about the acknowledgments?

Solution

The bandwidth-delay product is still 20,000 bits. The system can send up to 15 frames or 15,000 bits during a round trip. This means the utilization is 15,000/20,000, or 75 percent. Of course, if there are damaged frames, the utilization percentage is much less because frames have to be resent.

Pipelining

In networking and in other areas, a task is often begun before the previous task has ended. This is known as **pipelining.** There is no pipelining in Stop-and-Wait ARQ because we need to wait for a frame to reach the destination and be acknowledged before the next frame can be sent. However, pipelining does apply to our next two protocols because

several frames can be sent before we receive news about the previous frames. Pipelining improves the efficiency of the transmission if the number of bits in transition is large with respect to the bandwidth-delay product.

Go-Back-*N* Automatic Repeat Request

To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be outstanding to keep the channel busy while the sender is waiting for acknowledgment. In this section, we discuss one protocol that can achieve this goal; in the next section, we discuss a second.

The first is called **Go-Back-*N* Automatic Repeat Request** (the rationale for the name will become clear later). In this protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

Sequence Numbers

Frames from a sending station are numbered sequentially. However, because we need to include the sequence number of each frame in the header, we need to set a limit. If the header of the frame allows m bits for the sequence number, the sequence numbers range from 0 to $2^m - 1$. For example, if m is 4, the only sequence numbers are 0 through 15 inclusive. However, we can repeat the sequence. So the sequence numbers are

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, . . .**

In other words, the sequence numbers are modulo-2^m.

In the Go-Back-*N* Protocol, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.

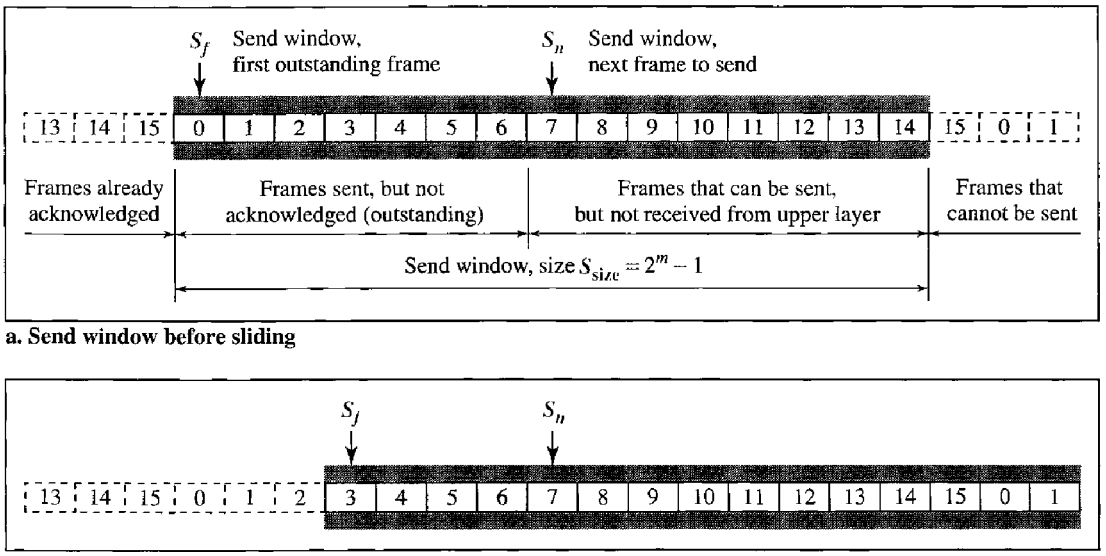
Sliding Window

In this protocol (and the next), the **sliding window** is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, the sender and receiver need to deal with only part of the possible sequence numbers. The range which is the concern of the sender is called the **send sliding window;** the range that is the concern of the receiver is called the **receive sliding window.** We discuss both here.

The send window is an imaginary box covering the sequence numbers of the data frames which can be in transit. In each window position, some of these sequence numbers define the frames that have been sent; others define those that can be sent. The maximum size of the window is $2^m - 1$ for reasons that we discuss later. In this chapter, we let the size be fixed and set to the maximum value, but we will see in future chapters that some protocols may have a variable window size. Figure 11.12 shows a sliding window of size 15 ($m = 4$).

The window at any time divides the possible sequence numbers into four regions. The first region, from the far left to the left wall of the window, defines the sequence

Figure 11.12 *Send window for Go-Back-N ARQ*



a. Send window before sliding

b. Send window after sliding

numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them. The second region, colored in Figure 11.12a, defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. We call these outstanding frames. The third range, white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network layer. Finally, the fourth region defines sequence numbers that cannot be used until the window slides, as we see next.

The window itself is an abstraction; three variables define its size and location at any time. We call these variables S_f (send window, the first outstanding frame), S_n (send window, the next frame to be sent), and S_{size} (send window, size). The variable S_f defines the sequence number of the first (oldest) outstanding frame. The variable S_n holds the sequence number that will be assigned to the next frame to be sent. Finally, the variable S_{size} defines the size of the window, which is fixed in our protocol.

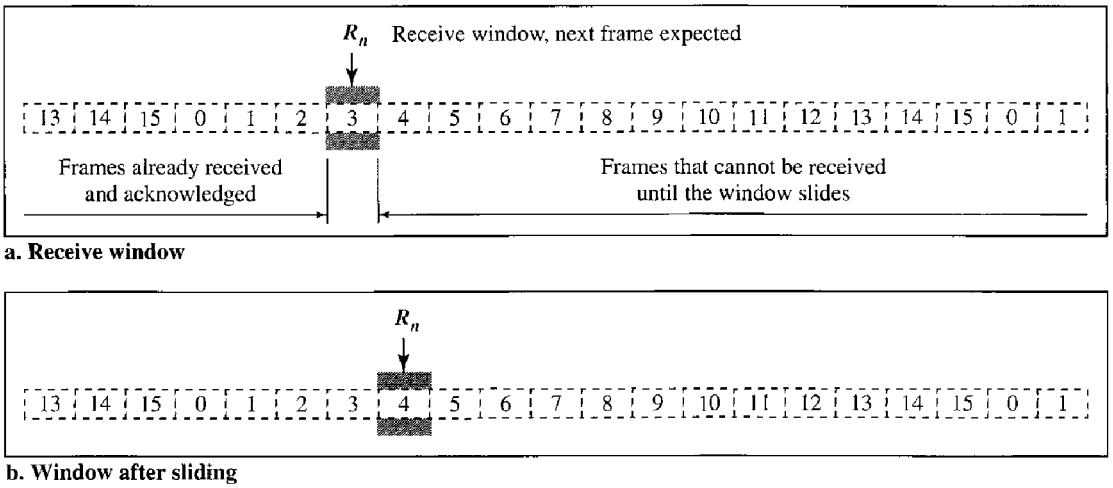
The send window is an abstract concept defining an imaginary box of size $2^m - 1$ with three variables: S_f, S_n, and S_{size}.

Figure 11.12b shows how a send window can slide one or more slots to the right when an acknowledgment arrives from the other end. As we will see shortly, the acknowledgments in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame. In Figure 11.12b, frames 0, 1, and 2 are acknowledged, so the window has slid to the right three slots. Note that the value of S_f is 3 because frame 3 is now the first outstanding frame.

The send window can slide one or more slots when a valid acknowledgment arrives.

The receive window makes sure that the correct data frames are received and that the correct acknowledgments are sent. The size of the receive window is always 1. The receiver is always looking for the arrival of a specific frame. Any frame arriving out of order is discarded and needs to be resent. Figure 11.13 shows the receive window.

Figure 11.13 *Receive window for Go-Back-N ARQ*



The receive window is an abstract concept defining an imaginary box of size 1 with one single variable R_n. The window slides when a correct frame has arrived; sliding occurs one slot at a time.

Note that we need only one variable R_n (receive window, next frame expected) to define this abstraction. The sequence numbers to the left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this window define the frames that cannot be received. Any received frame with a sequence number in these two regions is discarded. Only a frame with a sequence number matching the value of R_n is accepted and acknowledged.

The receive window also slides, but only one slot at a time. When a correct frame is received (and a frame is received only one at a time), the window slides.

Timers

Although there can be a timer for each frame that is sent, in our protocol we use only one. The reason is that the timer for the first outstanding frame always expires first; we send all outstanding frames when this timer expires.

Acknowledgment

The receiver sends a positive acknowledgment if a frame has arrived safe and sound and in order. If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting. The silence of

the receiver causes the timer of the unacknowledged frame at the sender site to expire. This, in turn, causes the sender to go back and resend all frames, beginning with the one with the expired timer. The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgment for several frames.

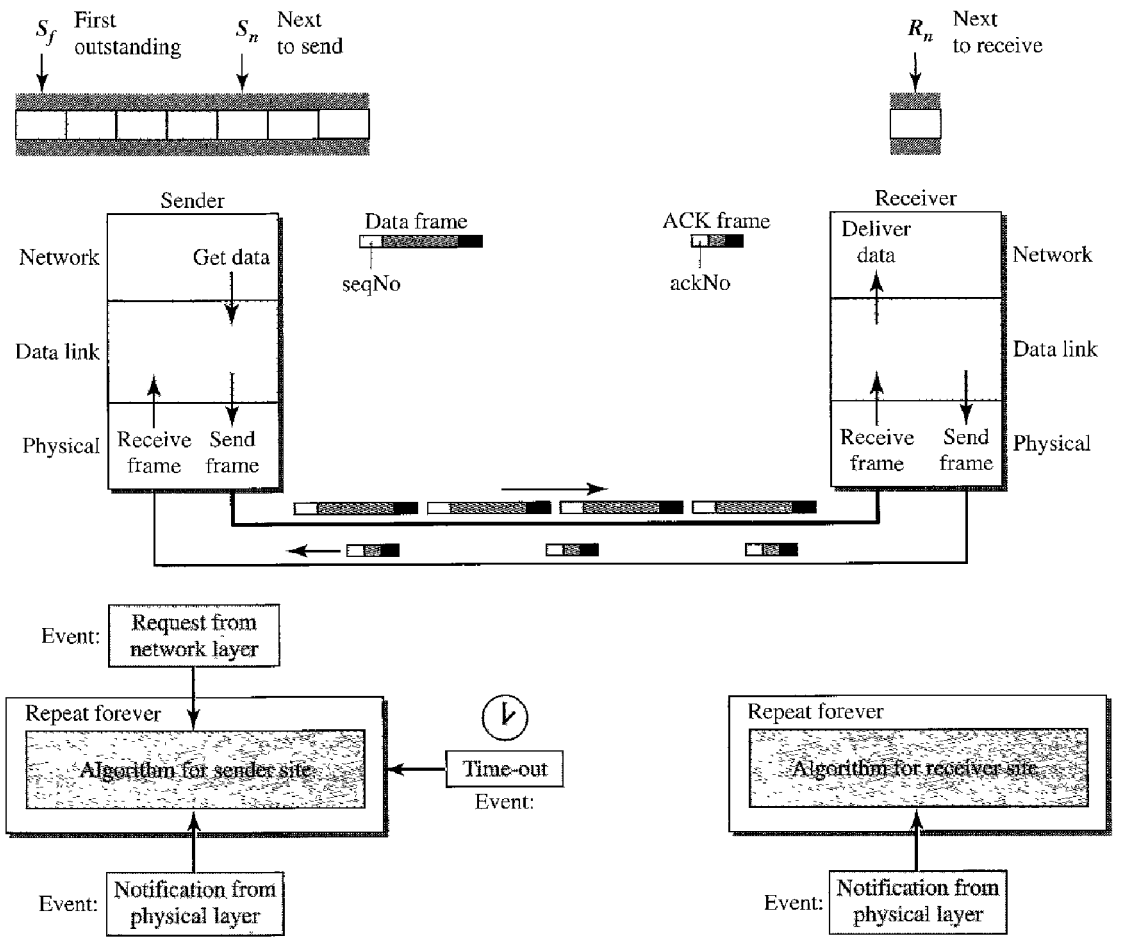
Resending a Frame

When the timer expires, the sender resends all outstanding frames. For example, suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged; the sender goes back and sends frames 3, 4, 5, and 6 again. That is why the protocol is called Go-Back-*N* ARQ.

Design

Figure 11.14 shows the design for this protocol. As we can see, multiple frames can be in transit in the forward direction, and multiple acknowledgments in the reverse direction. The idea is similar to Stop-and-Wait ARQ; the difference is that the send

Figure 11.14 *Design of Go-Back-N ARQ*

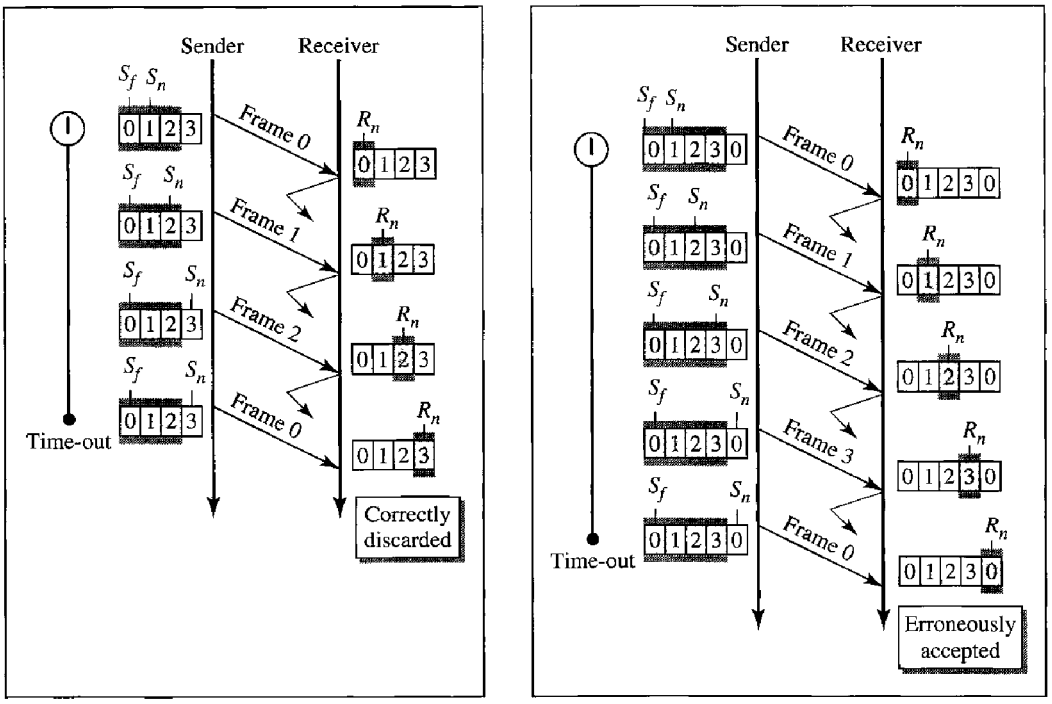


window allows us to have as many frames in transition as there are slots in the send window.

Send Window Size

We can now show why the size of the send window must be less than 2^m. As an example, we choose $m = 2$, which means the size of the window can be $2^m - 1$, or 3. Figure 11.15 compares a window size of 3 against a window size of 4. If the size of the window is 3 (less than 2^2) and all three acknowledgments are lost, the frame 0 timer expires and all three frames are resent. The receiver is now expecting frame 3, not frame 0, so the duplicate frame is correctly discarded. On the other hand, if the size of the window is 4 (equal to 2^2) and all acknowledgments are lost, the sender will send a duplicate of frame 0. However, this time the window of the receiver expects to receive frame 0, so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is an error.

Figure 11.15 *Window size for Go-Back-N ARQ*



a. Window size < 2^m b. Window size = 2^m

In Go-Back-*N* ARQ, the size of the send window must be less than 2^m; the size of the receiver window is always 1.

Algorithms

Algorithm 11.7 shows the procedure for the sender in this protocol.

Algorithm 11.7 *Go-Back-N sender algorithm*

```
1  Sw = 2^m - 1;
2  Sf = 0;
3  Sn = 0;
4
5  while (true)                          //Repeat forever
6  {
7   WaitForEvent();
8    if(Event(RequestToSend))            //A packet to send
9    {
10      if(Sn-Sf >= Sw)                  //If window is full
11            Sleep();
12      GetData();
13      MakeFrame(Sn);
14      StoreFrame(Sn);
15      SendFrame(Sn);
16      Sn = Sn + 1;
17      if(timer not running)
18           StartTimer();
19   }
20
21   if(Event(ArrivalNotification)) //ACK arrives
22   {
23      Receive(ACK);
24      if(corrupted(ACK))
25           Sleep();
26      if((ackNo>Sf)&&(ackNo<=Sn))  //If a valid ACK
27      While(Sf <= ackNo)
28       {
29        PurgeFrame(Sf);
30        Sf = Sf + 1;
31       }
32       StopTimer();
33   }
34
35   if(Event(TimeOut))                 //The timer expires
36   {
37    StartTimer();
38    Temp = Sf;
39    while(Temp < Sn);
40     {
41      SendFrame(Sf);
42      Sf = Sf + 1;
43     }
44   }
45 }
```

Analysis This algorithm first initializes three variables. Unlike Stop-and-Wait ARQ, this protocol allows several requests from the network layer without the need for other events to occur; we just need to be sure that the window is not full (line 12). In our approach, if the window is full,

the request is just ignored and the network layer needs to try again. Some implementations use other methods such as enabling or disabling the network layer. The handling of the arrival event is more complex than in the previous protocol. If we receive a corrupted ACK, we ignore it. If the ackNo belongs to one of the outstanding frames, we use a loop to purge the buffers and move the left wall to the right. The time-out event is also more complex. We first start a new timer. We then resend all outstanding frames.

Algorithm 11.8 is the procedure at the receiver site.

Algorithm 11.8 *Go-Back-N receiver algorithm*

```
1  Rn = 0;
2
3  while (true)                          //Repeat forever
4  {
5    WaitForEvent();
6
7    if(Event(ArrivalNotification)) /Data frame arrives
8    {
9       Receive(Frame);
10      if(corrupted(Frame))
11           Sleep();
12      if(seqNo == Rn)                  //If expected frame
13      {
14        DeliverData();                 //Deliver data
15        Rn = Rn + 1;                   //Slide window
16        SendACK(Rn);
17      }
18   }
19 }
```

Analysis This algorithm is simple. We ignore a corrupt or out-of-order frame. If a frame arrives with an expected sequence number, we deliver the data, update the value of R_n, and send an ACK with the ackNo showing the next frame expected.

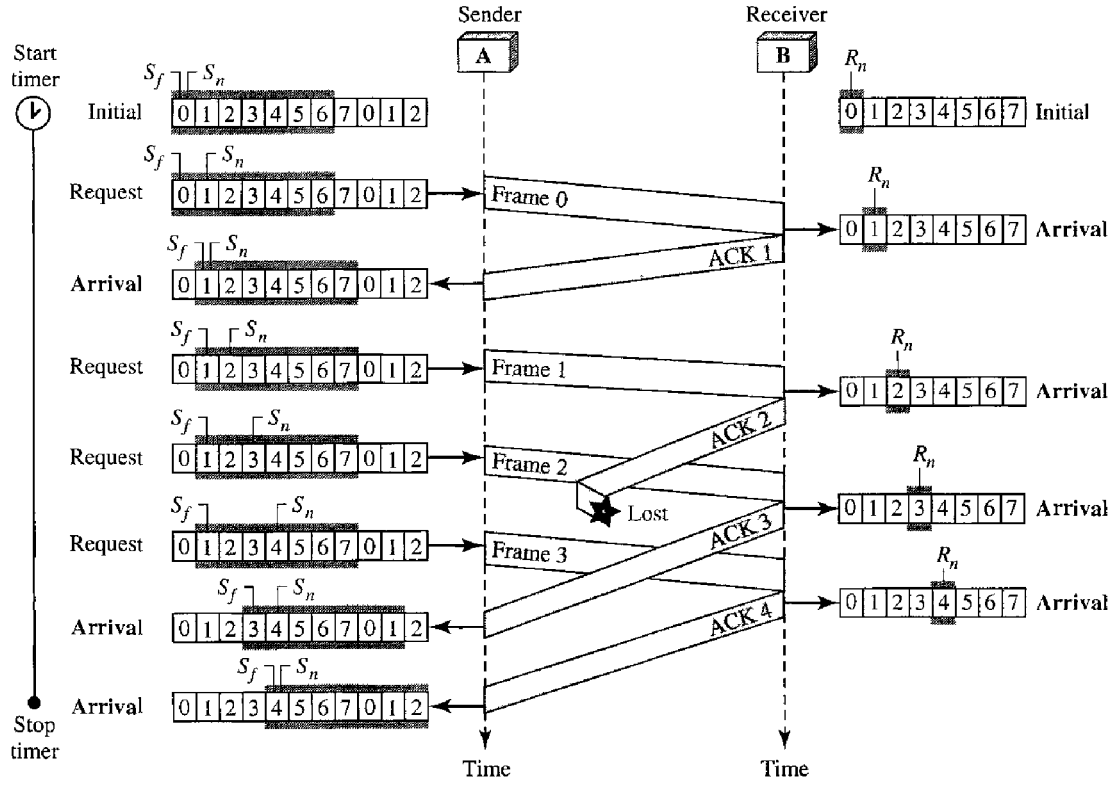
Example 11.6

Figure 11.16 shows an example of Go-Back-*N*. This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost.

After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK 3.

There are four receiver events, all triggered by the arrival of frames from the physical layer.

Figure 11.16 *Flow diagram for Example 11.6*



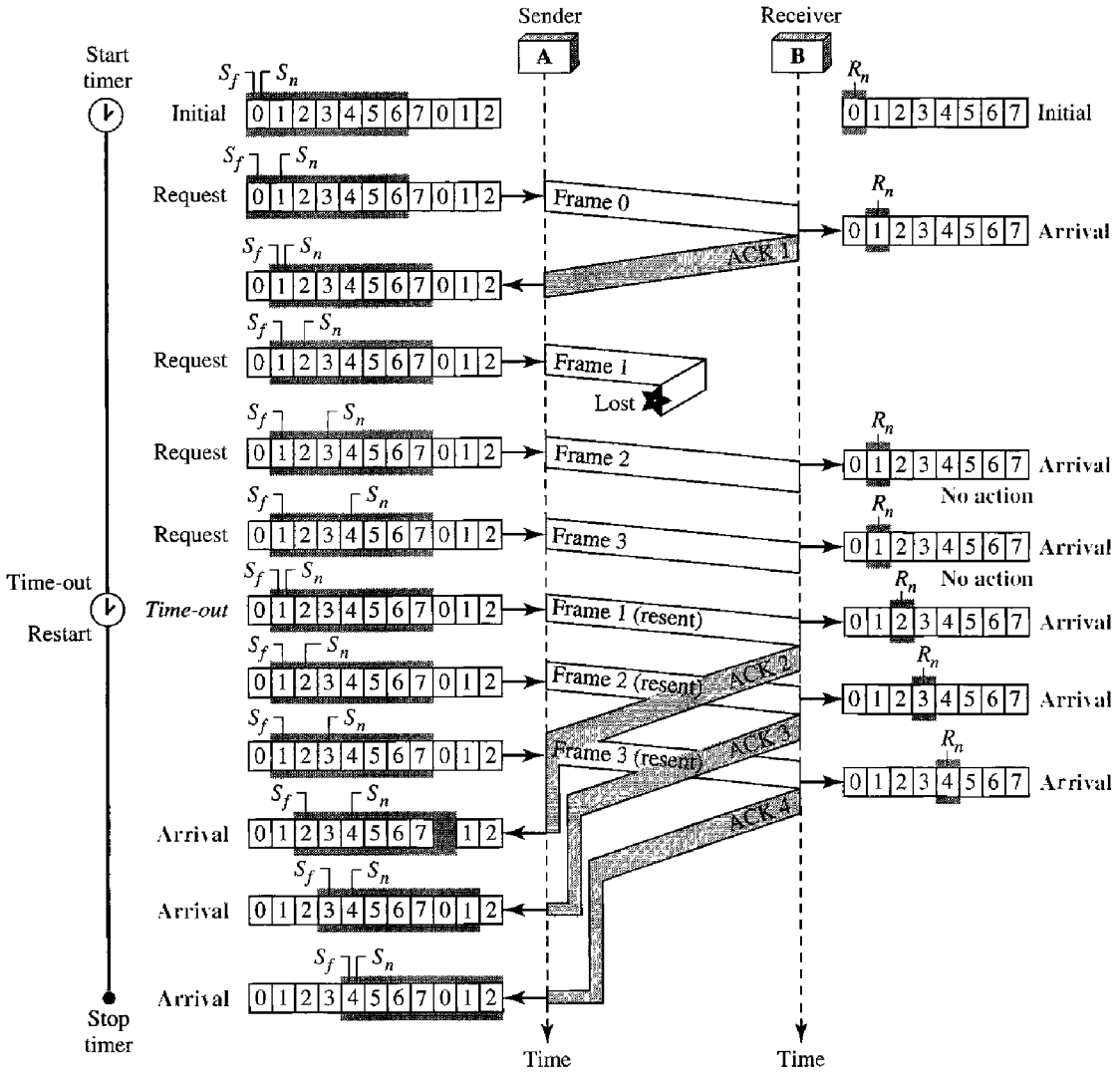
Example 11.7

Figure 11.17 shows what happens when a frame is lost. Frames 0, 1, 2, and 3 are sent. However, frame 1 is lost. The receiver receives frames 2 and 3, but they are discarded because they are received out of order (frame 1 is expected). The sender receives no acknowledgment about frames 1, 2, or 3. Its timer finally expires. The sender sends all outstanding frames (1, 2, and 3) because it does not know what is wrong. Note that the resending of frames 1, 2, and 3 is the response to one single event. When the sender is responding to this event, it cannot accept the triggering of other events. This means that when ACK 2 arrives, the sender is still busy with sending frame 3. The physical layer must wait until this event is completed and the data link layer goes back to its sleeping state. We have shown a vertical line to indicate the delay. It is the same story with ACK 3; but when ACK 3 arrives, the sender is busy responding to ACK 2. It happens again when ACK 4 arrives. Note that before the second timer expires, all outstanding frames have been sent and the timer is stopped.

Go-Back-N ARQ Versus Stop-and-Wait ARQ

The reader may find that there is a similarity between Go-Back-*N* ARQ and Stop-and-Wait ARQ. We can say that the Stop-and-Wait ARQ Protocol is actually a Go-Back-*N* ARQ in which there are only two sequence numbers and the send window size is 1. In other words, $m = 1$, $2^m - 1 = 1$. In Go-Back-*N* ARQ, we said that the addition is modulo-2^m; in Stop-and-Wait ARQ it is 2, which is the same as 2^m when $m = 1$.

Figure 11.17 *Flow diagram for Example 11.7*



Stop-and-Wait ARQ is a special case of Go-Back-*N* ARQ in which the size of the send window is 1.

Selective Repeat Automatic Repeat Request

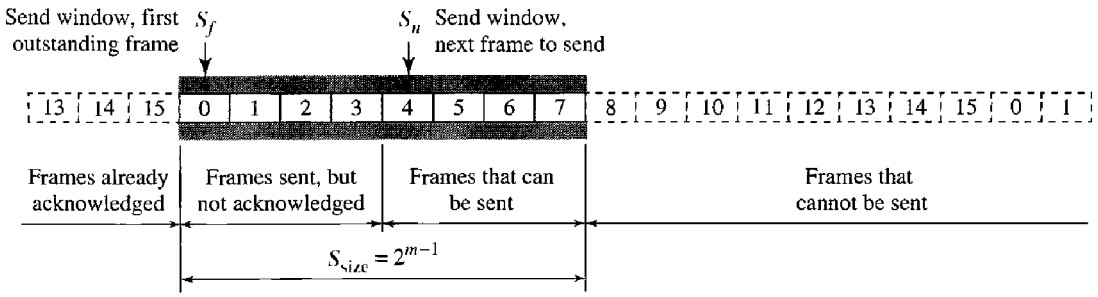
Go-Back-*N* ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission. For noisy links, there is another mechanism that does not resend *N* frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called **Selective Repeat ARQ.** It is more efficient for noisy links, but the processing at the receiver is more complex.

Windows

The Selective Repeat Protocol also uses two windows: a send window and a receive window. However, there are differences between the windows in this protocol and the ones in Go-Back-*N*. First, the size of the send window is much smaller; it is 2^{m-1}. The reason for this will be discussed later. Second, the receive window is the same size as the send window.

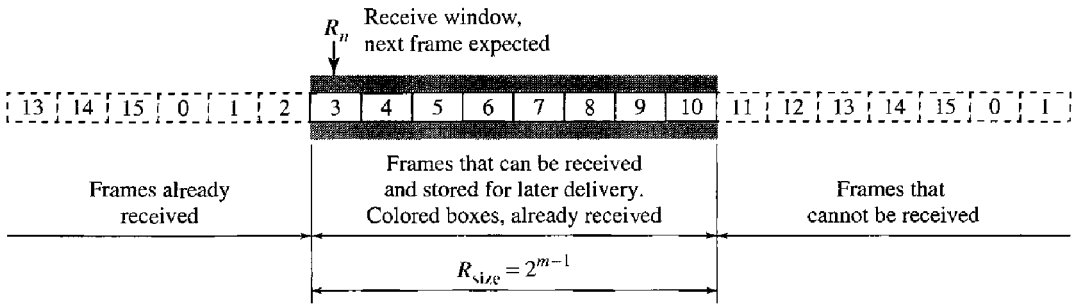
The send window maximum size can be 2^{m-1}. For example, if $m = 4$, the sequence numbers go from 0 to 15, but the size of the window is just 8 (it is 15 in the Go-Back-*N* Protocol). The smaller window size means less efficiency in filling the pipe, but the fact that there are fewer duplicate frames can compensate for this. The protocol uses the same variables as we discussed for Go-Back-*N*. We show the Selective Repeat send window in Figure 11.18 to emphasize the size. Compare it with Figure 11.12.

Figure 11.18 *Send window for Selective Repeat ARQ*



The receive window in Selective Repeat is totally different from the one in Go-Back-*N*. First, the size of the receive window is the same as the size of the send window (2^{m-1}). The Selective Repeat Protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer. Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered. We need, however, to mention that the receiver never delivers packets out of order to the network layer. Figure 11.19 shows the receive window in this

Figure 11.19 *Receive window for Selective Repeat ARQ*

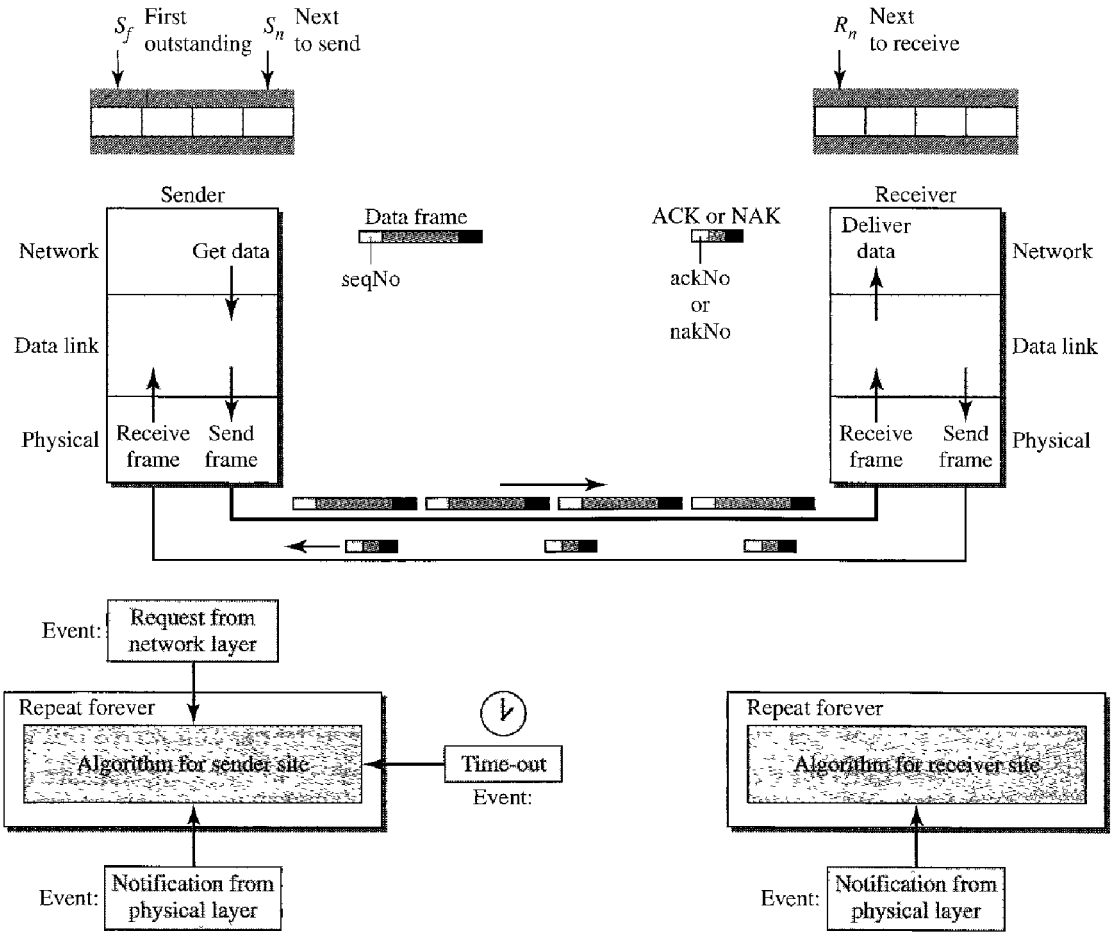


protocol. Those slots inside the window that are colored define frames that have arrived out of order and are waiting for their neighbors to arrive before delivery to the network layer.

Design

The design in this case is to some extent similar to the one we described for the Go-Back-*N*, but more complicated, as shown in Figure 11.20.

Figure 11.20 *Design of Selective Repeat ARQ*

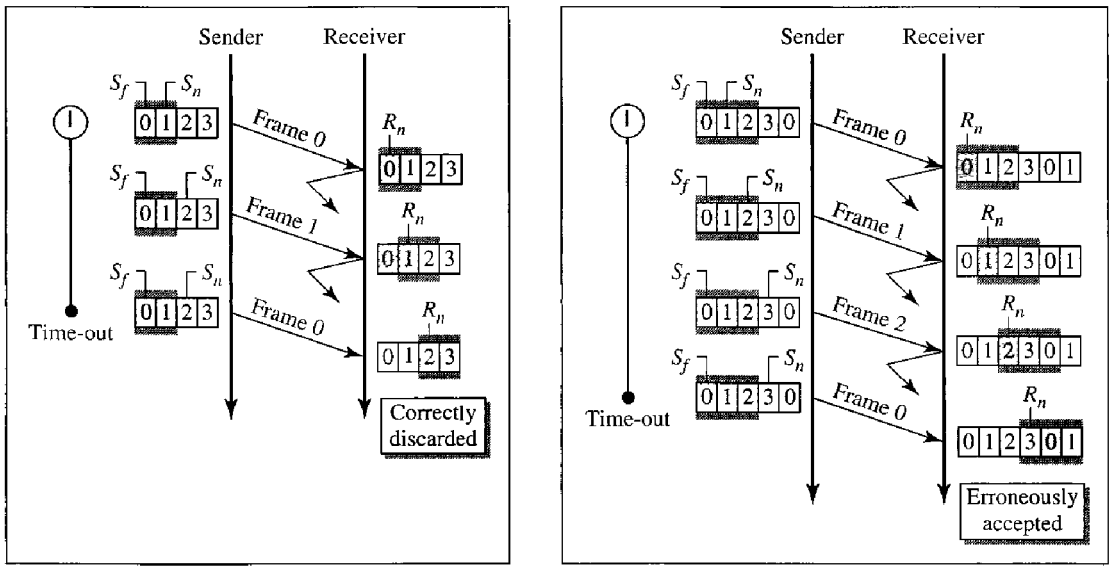


Window Sizes

We can now show why the size of the sender and receiver windows must be at most one-half of 2^m. For an example, we choose $m = 2$, which means the size of the window is $2^m/2$, or 2. Figure 11.21 compares a window size of 2 with a window size of 3.

If the size of the window is 2 and all acknowledgments are lost, the timer for frame 0 expires and frame 0 is resent. However, the window of the receiver is now expecting

Figure 11.21 *Selective Repeat ARQ, window size*



a. Window size = 2^{m-1} b. Window size > 2^{m-1}

frame 2, not frame 0, so this duplicate frame is correctly discarded. When the size of the window is 3 and all acknowledgments are lost, the sender sends a duplicate of frame 0. However, this time, the window of the receiver expects to receive frame 0 (0 is part of the window), so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is clearly an error.

In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^m.

Algorithms

Algorithm 11.9 shows the procedure for the sender.

Algorithm 11.9 ***Sender-site Selective Repeat algorithm***

```
1  Sw = 2^(m-1) ;
2  Sf = 0;
3  Sn = 0;
4
5  while (true)                          //Repeat forever
6  {
7    WaitForEvent();
8    if(Event(RequestToSend))            //There is a packet to send
9    {
```

Algorithm 11.9 *Sender-site Selective Repeat algorithm (continued)*

```
10      if(Sn-Sf >= Sw)                 //If window is full
11            Sleep();
12      GetData();
13      MakeFrame(Sn);
14      StoreFrame(Sn);
15      SendFrame(Sn);
16      Sn = Sn + 1;
17      StartTimer(Sn);
18    }
19
20    if(Event(ArrivalNotification)) //ACK arrives
21    {
22      Receive(frame);               //Receive ACK or NAK
23      if(corrupted(frame))
24           Sleep();
25      if (FrameType == NAK)
26         if (nakNo between Sf and Sn)
27         {
28          resend(nakNo);
29          StartTimer(nakNo);
30         }
31      if (FrameType == ACK)
32         if (ackNo between Sf and Sn)
33         {
34           while(sf < ackNo)
35           {
36            Purge(sf);
37            StopTimer(sf);
38            Sf = Sf + 1;
39           }
40         }
41    }
42
43    if(Event(TimeOut(t)))           //The timer expires
44    {
45     StartTimer(t);
46     SendFrame(t);
47    }
48  }
```

Analysis The handling of the request event is similar to that of the previous protocol except that one timer is started for each frame sent. The arrival event is more complicated here. An ACK or a NAK frame may arrive. If a valid NAK frame arrives, we just resend the corresponding frame. If a valid ACK arrives, we use a loop to purge the buffers, stop the corresponding timer, and move the left wall of the window. The time-out event is simpler here; only the frame which times out is resent.

Algorithm 11.10 shows the procedure for the receiver.

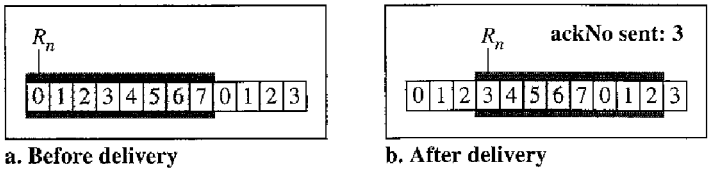
Algorithm 11.10 ***Receiver-site Selective Repeat algorithm***

```
1  Rn = 0;
2  NakSent = false;
3  AckNeeded = false;
4  Repeat(for all slots)
5      Marked(slot) = false;
6
7  while (true)                                  //Repeat forever
8  {
9    WaitForEvent();
10
11   if(Event(ArrivalNotification))              /Data frame arrives
12   {
13      Receive(Frame);
14      if(corrupted(Frame))&& (NOT NakSent)
15      {
16       SendNAK(Rn);
17       NakSent = true;
18       Sleep();
19      }
20      if(seqNo <> Rn)&& (NOT NakSent)
21      {
22       SendNAK(Rn);
23       NakSent = true;
24       if ((seqNo in window)&&(!Marked(seqNo))
25       {
26        StoreFrame(seqNo)
27        Marked(seqNo)= true;
28        while(Marked(Rn))
29        {
30         DeliverData(Rn);
31         Purge(Rn);
32         Rn = Rn + 1;
33         AckNeeded = true;
34        }
35         if(AckNeeded);
36         {
37         SendAck(Rn);
38         AckNeeded = false;
39         NakSent = false;
40         }
41       }
42      }
43   }
44 }
```

Analysis Here we need more initialization. In order not to overwhelm the other side with NAKs, we use a variable called NakSent. To know when we need to send an ACK, we use a variable called AckNeeded. Both of these are initialized to false. We also use a set of variables to

mark the slots in the receive window once the corresponding frame has arrived and is stored. If we receive a corrupted frame and a NAK has not yet been sent, we send a NAK to tell the other site that we have not received the frame we expected. If the frame is not corrupted and the sequence number is in the window, we store the frame and mark the slot. If contiguous frames, starting from R_n have been marked, we deliver their data to the network layer and slide the window. Figure 11.22 shows this situation.

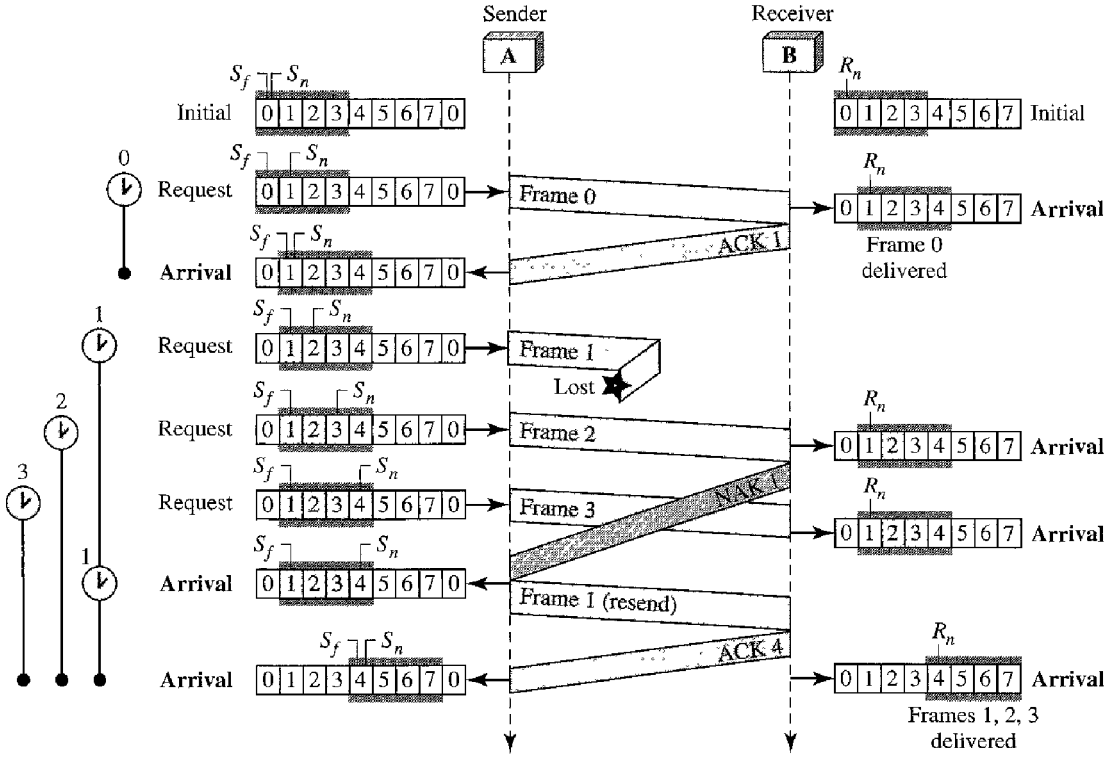
Figure 11.22 *Delivery of data in Selective Repeat ARQ*



Example 11.8

This example is similar to Example 11.3 in which frame 1 is lost. We show how Selective Repeat behaves in this case. Figure 11.23 shows the situation.

Figure 11.23 *Flow diagram for Example 11.8*



One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1, 2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives. The timer for frame 1 starts at the second request, restarts when a NAK arrives, and finally stops when the last ACK arrives. The other two timers start when the corresponding frames are sent and stop at the last arrival event.

At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer. At the second arrival, frame 2 arrives and is stored and marked (colored slot), but it cannot be delivered because frame 1 is missing. At the next arrival, frame 3 arrives and is marked and stored, but still none of the frames can be delivered. Only at the last arrival, when finally a copy of frame 1 arrives, can frames 1, 2, and 3 be delivered to the network layer. There are two conditions for the delivery of frames to the network layer: First, a set of consecutive frames must have arrived. Second, the set starts from the beginning of the window. After the first arrival, there was only one frame and it started from the beginning of the window. After the last arrival, there are three frames and the first one starts from the beginning of the window.

Another important point is that a NAK is sent after the second arrival, but not after the third, although both situations look the same. The reason is that the protocol does not want to crowd the network with unnecessary NAKs and unnecessary resent frames. The second NAK would still be NAK1 to inform the sender to resend frame 1 again; this has already been done. The first NAK sent is remembered (using the nakSent variable) and is not sent again until the frame slides. A NAK is sent once for each window position and defines the first slot in the window.

The next point is about the ACKs. Notice that only two ACKs are sent here. The first one acknowledges only the first frame; the second one acknowledges three frames. In Selective Repeat, ACKs are sent when data are delivered to the network layer. If the data belonging to n frames are delivered in one shot, only one ACK is sent for all of them.

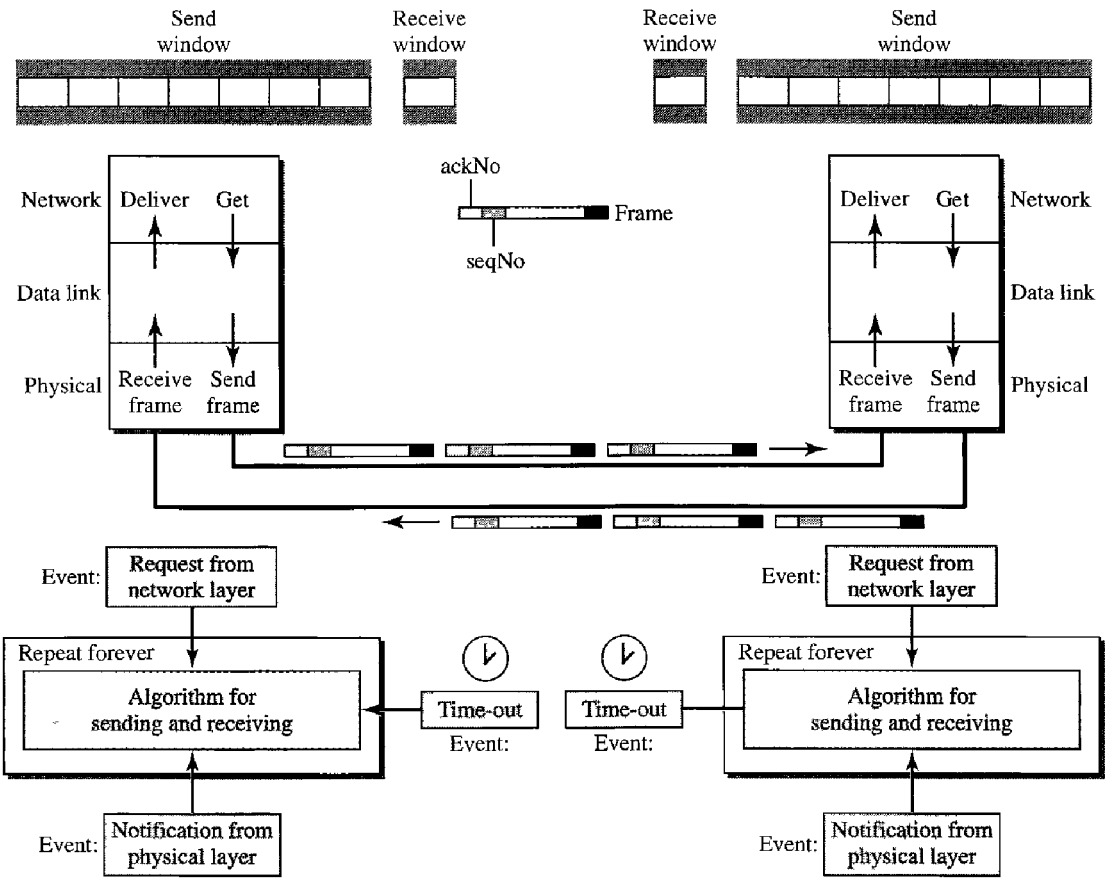
Piggybacking

The three protocols we discussed in this section are all unidirectional: data frames flow in only one direction although control information such as ACK and NAK frames can travel in the other direction. In real life, data frames are normally flowing in both directions: from node A to node B and from node B to node A. This means that the control information also needs to flow in both directions. A technique called **piggybacking** is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

We show the design for a Go-Back-N ARQ using piggybacking in Figure 11.24. Note that each node now has two windows: one send window and one receive window. Both also need to use a timer. Both are involved in three types of events: request, arrival, and time-out. However, the arrival event here is complicated; when a frame arrives, the site needs to handle control information as well as the frame itself. Both of these concerns must be taken care of in one event, the arrival event. The request event uses only the send window at each site; the arrival event needs to use both windows.

An important point about piggybacking is that both sites must use the same algorithm. This algorithm is complicated because it needs to combine two arrival events into one. We leave this task as an exercise.

Figure 11.24 *Design of piggybacking in Go-Back-N ARQ*



11.6 HDLC

High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. It implements the ARQ mechanisms we discussed in this chapter.

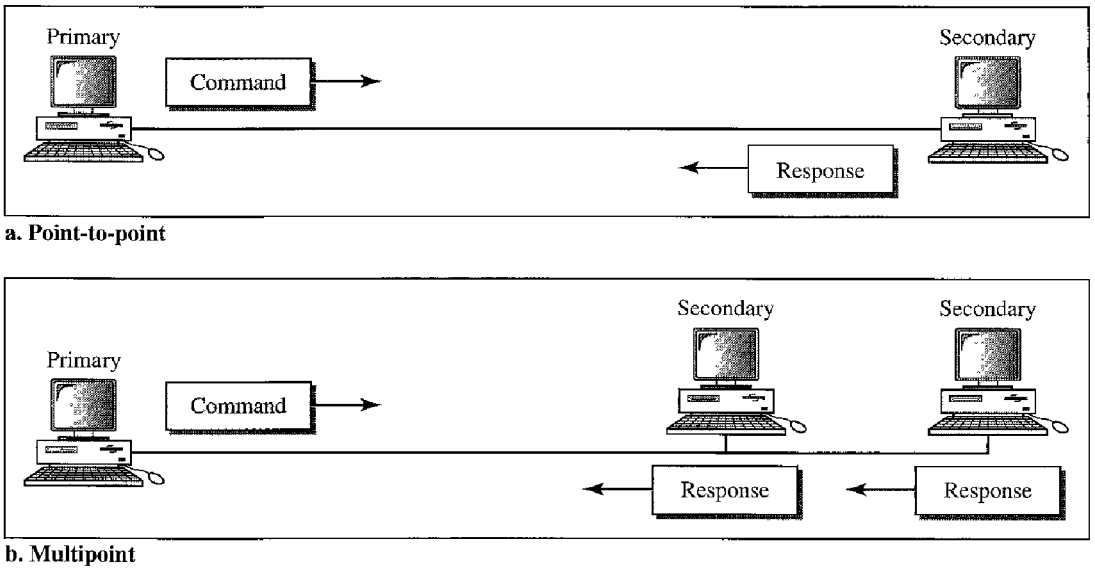
Configurations and Transfer Modes

HDLC provides two common transfer modes that can be used in different configurations: **normal response mode (NRM)** and **asynchronous balanced mode (ABM).**

Normal Response Mode

In normal response mode (NRM), the station configuration is unbalanced. We have one primary station and multiple secondary stations. A **primary station** can send commands; a **secondary station** can only respond. The NRM is used for both point-to-point and multiple-point links, as shown in Figure 11.25.

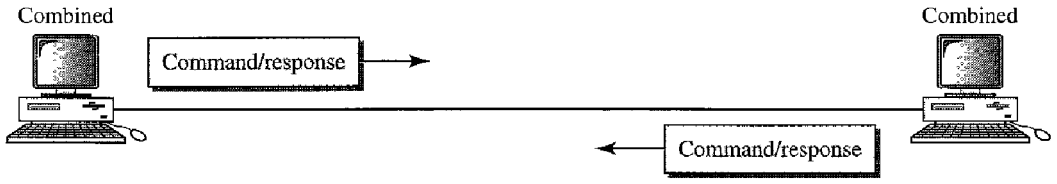
Figure 11.25 *Normal response mode*



Asynchronous Balanced Mode

In asynchronous balanced mode (ABM), the configuration is balanced. The link is point-to-point, and each station can function as a primary and a secondary (acting as peers), as shown in Figure 11.26. This is the common mode today.

Figure 11.26 *Asynchronous balanced mode*



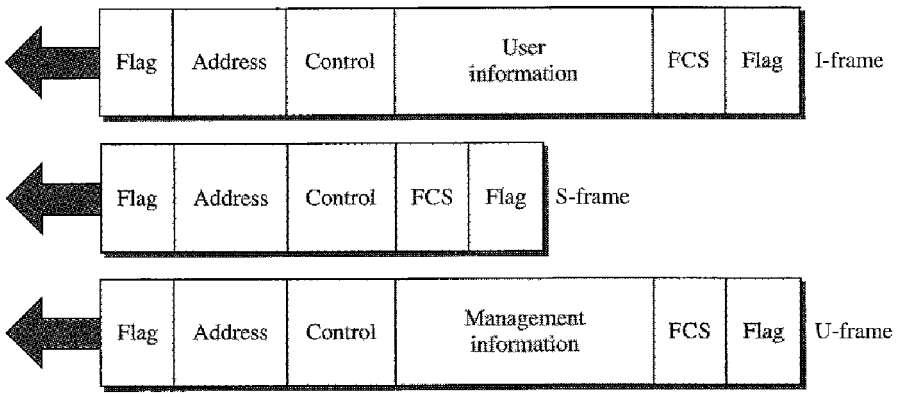
Frames

To provide the flexibility necessary to support all the options possible in the modes and configurations just described, HDLC defines three types of frames: **information frames (I-frames), supervisory frames (S-frames),** and **unnumbered frames (U-frames).** Each type of frame serves as an envelope for the transmission of a different type of message. I-frames are used to transport user data and control information relating to user data (piggybacking). S-frames are used only to transport control information. U-frames are reserved for system management. Information carried by U-frames is intended for managing the link itself.

Frame Format

Each frame in HDLC may contain up to six fields, as shown in Figure 11.27: a beginning flag field, an address field, a control field, an information field, a frame check sequence (FCS) field, and an ending flag field. In multiple-frame transmissions, the ending flag of one frame can serve as the beginning flag of the next frame.

Figure 11.27 *HDLC frames*



Fields

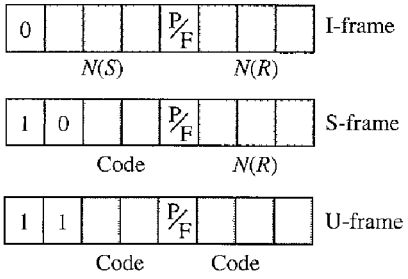
Let us now discuss the fields and their use in different frame types.

- **Flag field.** The flag field of an HDLC frame is an 8-bit sequence with the bit pattern 01111110 that identifies both the beginning and the end of a frame and serves as a synchronization pattern for the receiver.
- **Address field.** The second field of an HDLC frame contains the address of the secondary station. If a primary station created the frame, it contains a *to* address. If a secondary creates the frame, it contains a *from* address. An **address field** can be 1 byte or several bytes long, depending on the needs of the network. One byte can identify up to 128 stations (1 bit is used for another purpose). Larger networks require multiple-byte address fields. If the address field is only 1 byte, the last bit is always a 1. If the address is more than 1 byte, all bytes but the last one will end with 0; only the last will end with 1. Ending each intermediate byte with 0 indicates to the receiver that there are more address bytes to come.
- **Control field.** The control field is a 1- or 2-byte segment of the frame used for flow and error control. The interpretation of bits in this field depends on the frame type. We discuss this field later and describe its format for each frame type.
- **Information field.** The information field contains the user's data from the network layer or management information. Its length can vary from one network to another.
- **FCS field.** The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte ITU-T CRC.

Control Field

The control field determines the type of frame and defines its functionality. So let us discuss the format of this field in greater detail. The format is specific for the type of frame, as shown in Figure 11.28.

Figure 11.28 *Control field format for the different frame types*



Control Field for I-Frames

I-frames are designed to carry user data from the network layer. In addition, they can include flow and error control information (piggybacking). The subfields in the control field are used to define these functions. The first bit defines the type. If the first bit of the control field is 0, this means the frame is an I-frame. The next 3 bits, called $N(S)$, define the sequence number of the frame. Note that with 3 bits, we can define a sequence number between 0 and 7; but in the extension format, in which the control field is 2 bytes, this field is larger. The last 3 bits, called $N(R)$, correspond to the acknowledgment number when piggybacking is used. The single bit between $N(S)$ and $N(R)$ is called the P/F bit. The P/F field is a single bit with a dual purpose. It has meaning only when it is set (bit = 1) and can mean poll or final. It means *poll* when the frame is sent by a primary station to a secondary (when the address field contains the address of the receiver). It means *final* when the frame is sent by a secondary to a primary (when the address field contains the address of the sender).

Control Field for S-Frames

Supervisory frames are used for flow and error control whenever piggybacking is either impossible or inappropriate (e.g., when the station either has no data of its own to send or needs to send a command or response other than an acknowledgment). S-frames do not have information fields. If the first 2 bits of the control field is 10, this means the frame is an S-frame. The last 3 bits, called $N(R)$, corresponds to the acknowledgment number (ACK) or negative acknowledgment number (NAK) depending on the type of S-frame. The 2 bits called code is used to define the type of S-frame itself. With 2 bits, we can have four types of S-frames, as described below:

- **Receive ready (RR).** If the value of the code subfield is 00, it is an RR S-frame. This kind of frame acknowledges the receipt of a safe and sound frame or group of frames. In this case, the value $N(R)$ field defines the acknowledgment number.

- **Receive not ready (RNR).** If the value of the code subfield is 10, it is an RNR S-frame. This kind of frame is an RR frame with additional functions. It acknowledges the receipt of a frame or group of frames, and it announces that the receiver is busy and cannot receive more frames. It acts as a kind of congestion control mechanism by asking the sender to slow down. The value of $N(R)$ is the acknowledgment number.
- **Reject (REJ).** If the value of the code subfield is 01, it is a REJ S-frame. This is a NAK frame, but not like the one used for Selective Repeat ARQ. It is a NAK that can be used in Go-Back-N ARQ to improve the efficiency of the process by informing the sender, before the sender time expires, that the last frame is lost or damaged. The value of $N(R)$ is the negative acknowledgment number.
- **Selective reject (SREJ).** If the value of the code subfield is 11, it is an SREJ S-frame. This is a NAK frame used in Selective Repeat ARQ. Note that the HDLC Protocol uses the term *selective reject* instead of *selective repeat*. The value of $N(R)$ is the negative acknowledgment number.

Control Field for U-Frames

Unnumbered frames are used to exchange session management and control information between connected devices. Unlike S-frames, U-frames contain an information field, but one used for system management information, not user data. As with S-frames, however, much of the information carried by U-frames is contained in codes included in the control field. U-frame codes are divided into two sections: a 2-bit prefix before the P/F bit and a 3-bit suffix after the P/F bit. Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames. Some of the more common types are shown in Table 11.1.

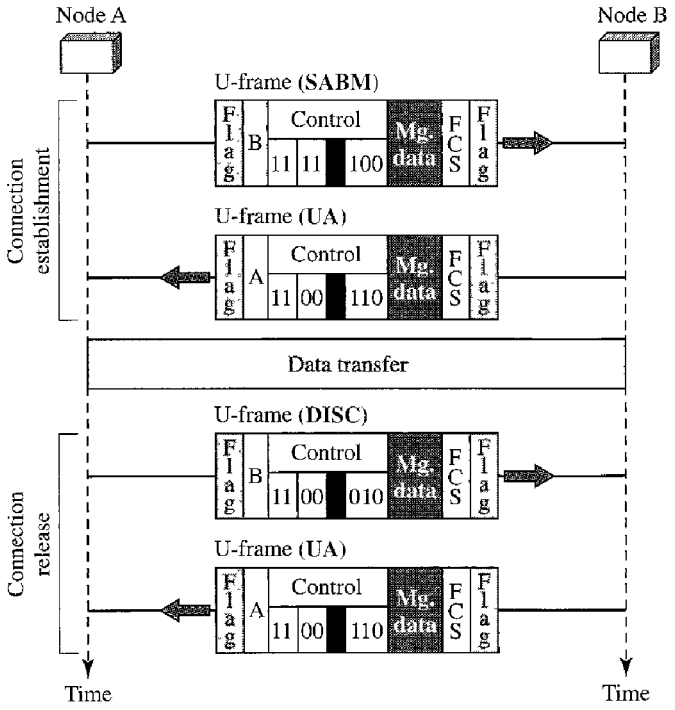
Table 11.1 *U-frame control command and response*

Code	*Command*	*Response*	*Meaning*
00 001	SNRM		Set normal response mode
11 011	SNRME		Set normal response mode, extended
11 100	SABM	DM	Set asynchronous balanced mode or **disconnect mode**
11 110	SABME		Set asynchronous balanced mode, extended
00 000	UI	UI	Unnumbered information
00 110		UA	**Unnumbered acknowledgment**
00 010	DISC	RD	Disconnect or **request disconnect**
10 000	SIM	RIM	Set initialization mode or **request information mode**
00 100	UP		Unnumbered poll
11 001	RSET		Reset
11 101	XID	XID	Exchange ID
10 001	FRMR	FRMR	Frame reject

Example 11.9: Connection/Disconnection

Figure 11.29 shows how U-frames can be used for connection establishment and connection release. Node A asks for a connection with a set asynchronous balanced mode (SABM) frame; node B gives a positive response with an unnumbered acknowledgment (UA) frame. After these two exchanges, data can be transferred between the two nodes (not shown in the figure). After data transfer, node A sends a DISC (disconnect) frame to release the connection; it is confirmed by node B responding with a UA (unnumbered acknowledgment).

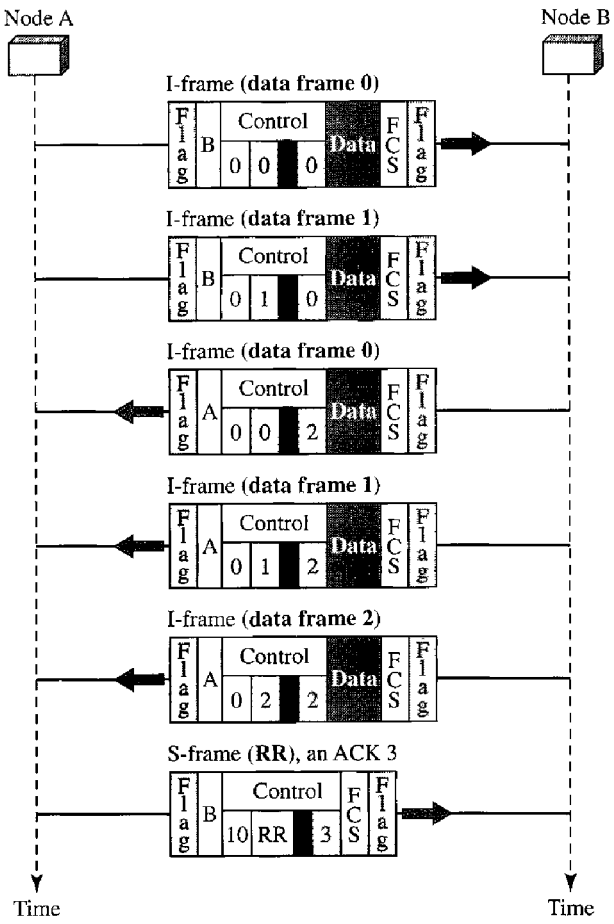
Figure 11.29 *Example of connection and disconnection*



Example 11.10: Piggybacking without Error

Figure 11.30 shows an exchange using piggybacking. Node A begins the exchange of information with an I-frame numbered 0 followed by another I-frame numbered 1. Node B piggybacks its acknowledgment of both frames onto an I-frame of its own. Node B's first I-frame is also numbered 0 [$N(S)$ field] and contains a 2 in its $N(R)$ field, acknowledging the receipt of A's frames 1 and 0 and indicating that it expects frame 2 to arrive next. Node B transmits its second and third I-frames (numbered 1 and 2) before accepting further frames from node A. Its $N(R)$ information, therefore, has not changed: B frames 1 and 2 indicate that node B is still expecting A's frame 2 to arrive next. Node A has sent all its data. Therefore, it cannot piggyback an acknowledgment onto an I-frame and sends an S-frame instead. The RR code indicates that A is still ready to receive. The number 3 in the $N(R)$ field tells B that frames 0, 1, and 2 have all been accepted and that A is now expecting frame number 3.

Figure 11.30 *Example of piggybacking without error*



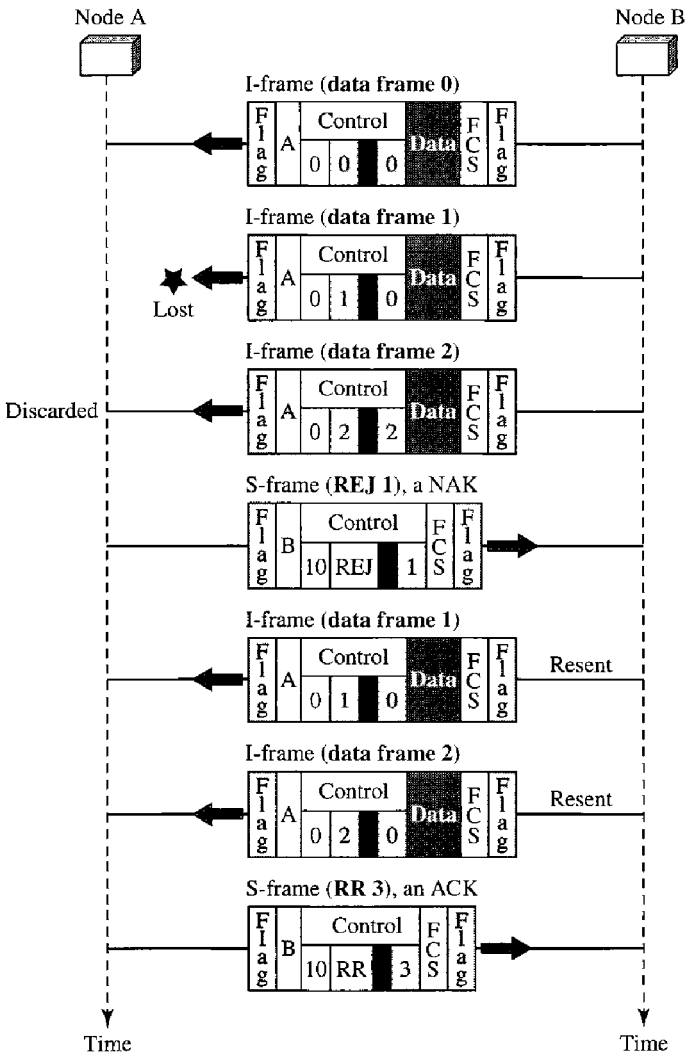
Example 11.11: Piggybacking with Error

Figure 11.31 shows an exchange in which a frame is lost. Node B sends three data frames (0, 1, and 2), but frame 1 is lost. When node A receives frame 2, it discards it and sends a REJ frame for frame 1. Note that the protocol being used is Go-Back-*N* with the special use of an REJ frame as a NAK frame. The NAK frame does two things here: It confirms the receipt of frame 0 and declares that frame 1 and any following frames must be resent. Node B, after receiving the REJ frame, resends frames 1 and 2. Node A acknowledges the receipt by sending an RR frame (ACK) with acknowledgment number 3.

11.7 POINT-TO-POINT PROTOCOL

Although HDLC is a general protocol that can be used for both point-to-point and multipoint configurations, one of the most common protocols for point-to-point access is the **Point-to-Point Protocol (PPP).** Today, millions of Internet users who need to connect their home computers to the server of an Internet service provider use PPP. The majority of these users have a traditional modem; they are connected to the Internet through a telephone line, which provides the services of the physical layer. But to control and

Figure 11.31 *Example of piggybacking with error*



manage the transfer of data, there is a need for a point-to-point protocol at the data link layer. PPP is by far the most common.

PPP provides several services:

1. PPP defines the format of the frame to be exchanged between devices.
2. PPP defines how two devices can negotiate the establishment of the link and the exchange of data.
3. PPP defines how network layer data are encapsulated in the data link frame.
4. PPP defines how two devices can authenticate each other.
5. PPP provides multiple network layer services supporting a variety of network layer protocols.
6. PPP provides connections over multiple links.
7. PPP provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

On the other hand, to keep PPP simple, several services are missing:

1. PPP does not provide flow control. A sender can send several frames one after another with no concern about overwhelming the receiver.
2. PPP has a very simple mechanism for error control. A CRC field is used to detect errors. If the frame is corrupted, it is silently discarded; the upper-layer protocol needs to take care of the problem. Lack of error control and sequence numbering may cause a packet to be received out of order.
3. PPP does not provide a sophisticated addressing mechanism to handle frames in a multipoint configuration.

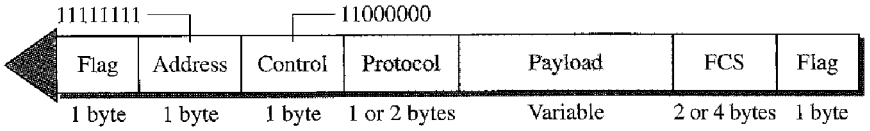
Framing

PPP is a byte-oriented protocol. Framing is done according to the discussion of byte-oriented protocols at the beginning of this chapter.

Frame Format

Figure 11.32 shows the format of a PPP frame. The description of each field follows:

Figure 11.32 *PPP frame format*



- **Flag.** A PPP frame starts and ends with a 1-byte flag with the bit pattern 01111110. Although this pattern is the same as that used in HDLC, there is a big difference. PPP is a byte-oriented protocol; HDLC is a bit-oriented protocol. The flag is treated as a byte, as we will explain later.
- **Address.** The address field in this protocol is a constant value and set to 11111111 (broadcast address). During negotiation (discussed later), the two parties may agree to omit this byte.
- **Control.** This field is set to the constant value 11000000 (imitating unnumbered frames in HDLC). As we will discuss later, PPP does not provide any flow control. Error control is also limited to error detection. This means that this field is not needed at all, and again, the two parties can agree, during negotiation, to omit this byte.
- **Protocol.** The protocol field defines what is being carried in the data field: either user data or other information. We discuss this field in detail shortly. This field is by default 2 bytes long, but the two parties can agree to use only 1 byte.
- **Payload field.** This field carries either the user data or other information that we will discuss shortly. The data field is a sequence of bytes with the default of a maximum of 1500 bytes; but this can be changed during negotiation. The data field is byte-stuffed if the flag byte pattern appears in this field. Because there is no field defining the size of the data field, padding is needed if the size is less than the maximum default value or the maximum negotiated value.
- **FCS.** The frame check sequence (FCS) is simply a 2-byte or 4-byte standard CRC.

Byte Stuffing

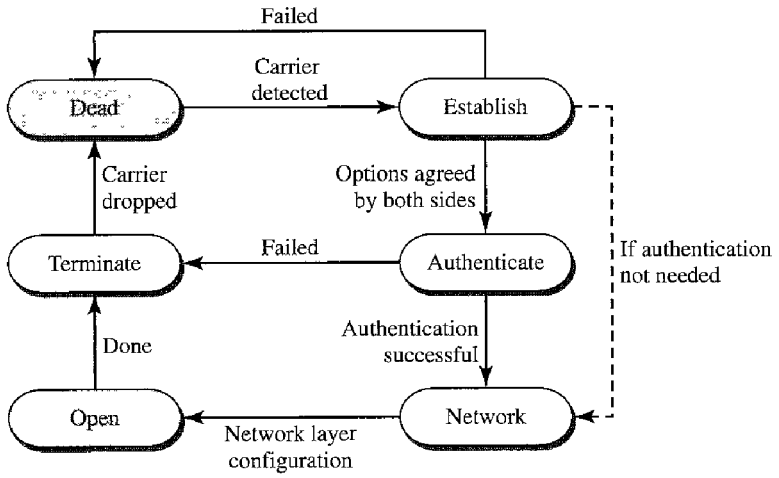
The similarity between PPP and HDLC ends at the frame format. PPP, as we discussed before, is a byte-oriented protocol totally different from HDLC. As a byte-oriented protocol, the flag in PPP is a byte and needs to be escaped whenever it appears in the data section of the frame. The escape byte is 01111101, which means that every time the flaglike pattern appears in the data, this extra byte is stuffed to tell the receiver that the next byte is not a flag.

PPP is a byte-oriented protocol using byte stuffing with the escape byte 01111101.

Transition Phases

A PPP connection goes through phases which can be shown in a **transition phase** diagram (see Figure 11.33).

Figure 11.33 *Transition phases*



- **Dead.** In the dead phase the link is not being used. There is no active carrier (at the physical layer) and the line is quiet.
- **Establish.** When one of the nodes starts the communication, the connection goes into this phase. In this phase, options are negotiated between the two parties. If the negotiation is successful, the system goes to the authentication phase (if authentication is required) or directly to the networking phase. The link control protocol packets, discussed shortly, are used for this purpose. Several packets may be exchanged here.
- **Authenticate.** The authentication phase is optional; the two nodes may decide, during the establishment phase, not to skip this phase. However, if they decide to proceed with authentication, they send several authentication packets, discussed later. If the result is successful, the connection goes to the networking phase; otherwise, it goes to the termination phase.
- **Network.** In the network phase, negotiation for the network layer protocols takes place. PPP specifies that two nodes establish a network layer agreement before data at

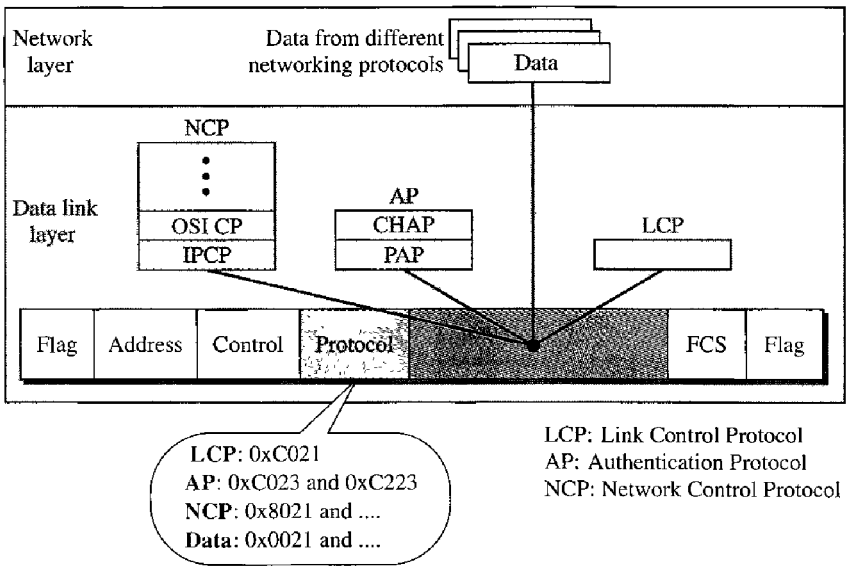
the network layer can be exchanged. The reason is that PPP supports multiple protocols at the network layer. If a node is running multiple protocols simultaneously at the network layer, the receiving node needs to know which protocol will receive the data.

- **Open.** In the open phase, data transfer takes place. When a connection reaches this phase, the exchange of data packets can be started. The connection remains in this phase until one of the endpoints wants to terminate the connection.
- **Terminate.** In the termination phase the connection is terminated. Several packets are exchanged between the two ends for house cleaning and closing the link.

Multiplexing

Although PPP is a data link layer protocol, PPP uses another set of other protocols to establish the link, authenticate the parties involved, and carry the network layer data. Three sets of protocols are defined to make PPP powerful: the Link Control Protocol (LCP), two Authentication Protocols (APs), and several Network Control Protocols (NCPs). At any moment, a PPP packet can carry data from one of these protocols in its data field, as shown in Figure 11.34. Note that there is one LCP, two APs, and several NCPs. Data may also come from several different network layers.

Figure 11.34 *Multiplexing in PPP*



Link Control Protocol

The **Link Control Protocol (LCP)** is responsible for establishing, maintaining, configuring, and terminating links. It also provides negotiation mechanisms to set options between the two endpoints. Both endpoints of the link must reach an agreement about the options before the link can be established. See Figure 11.35.

All LCP packets are carried in the payload field of the PPP frame with the protocol field set to C021 in hexadecimal.

The code field defines the type of LCP packet. There are 11 types of packets as shown in Table 11.2.

Figure 11.35 *LCP packet encapsulated in a frame*

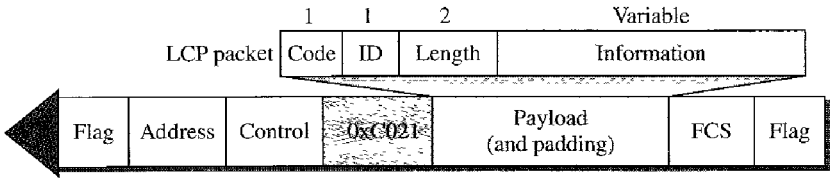


Table 11.2 *LCP packets*

Code	*Packet Type*	*Description*
0x01	Configure-request	Contains the list of proposed options and their values
0x02	Configure-ack	Accepts all options proposed
0x03	Configure-nak	Announces that some options are not acceptable
0x04	Configure-reject	Announces that some options are not recognized
0x05	Terminate-request	Request to shut down the line
0x06	Terminate-ack	Accept the shutdown request
0x07	Code-reject	Announces an unknown code
0x08	Protocol-reject	Announces an unknown protocol
0x09	Echo-request	A type of hello message to check if the other end is alive
0x0A	Echo-reply	The response to the echo-request message
0x0B	Discard-request	A request to discard the packet

There are three categories of packets. The first category, comprising the first four packet types, is used for link configuration during the establish phase. The second category, comprising packet types 5 and 6, is used for link termination during the termination phase. The last five packets are used for link monitoring and debugging.

The ID field holds a value that matches a request with a reply. One endpoint inserts a value in this field, which will be copied into the reply packet. The length field defines the length of the entire LCP packet. The information field contains information, such as options, needed for some LCP packets.

There are many options that can be negotiated between the two endpoints. Options are inserted in the information field of the configuration packets. In this case, the information field is divided into three fields: option type, option length, and option data. We list some of the most common options in Table 11.3.

Table 11.3 *Common options*

Option	*Default*
Maximum receive unit (payload field size)	1500
Authentication protocol	None
Protocol field compression	Off
Address and control field compression	Off

Authentication Protocols

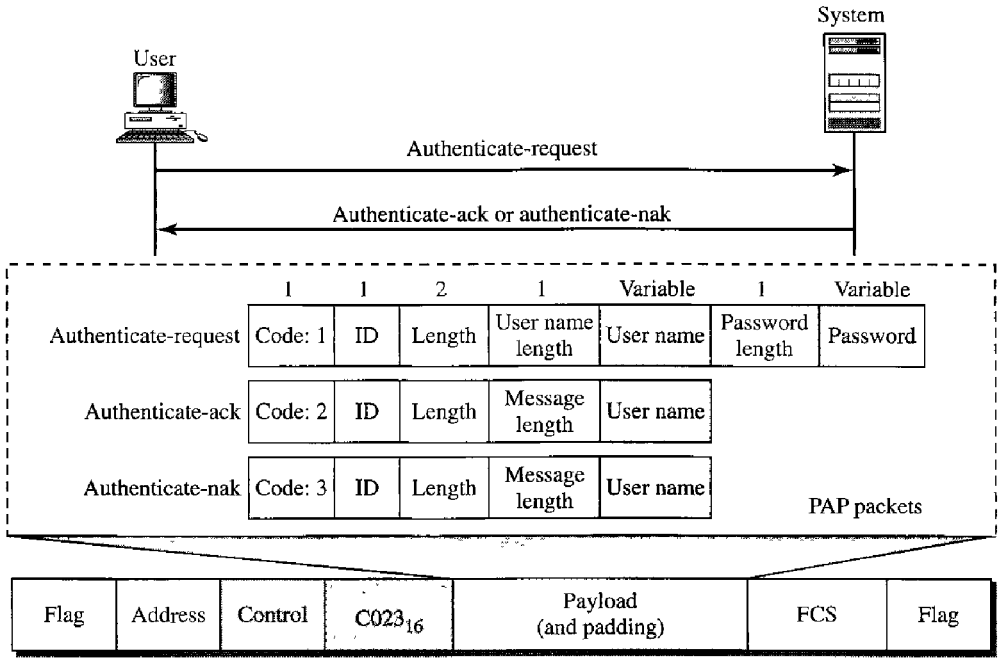
Authentication plays a very important role in PPP because PPP is designed for use over dial-up links where verification of user identity is necessary. **Authentication** means validating the identity of a user who needs to access a set of resources. PPP has created two protocols for authentication: Password Authentication Protocol and Challenge Handshake Authentication Protocol. Note that these protocols are used during the authentication phase.

PAP The **Password Authentication Protocol (PAP)** is a simple authentication procedure with a two-step process:

1. The user who wants to access a system sends an authentication identification (usually the user name) and a password.
2. The system checks the validity of the identification and password and either accepts or denies connection.

Figure 11.36 shows the three types of packets used by PAP and how they are actually exchanged. When a PPP frame is carrying any PAP packets, the value of the protocol field is 0xC023. The three PAP packets are authenticate-request, authenticate-ack, and authenticate-nak. The first packet is used by the user to send the user name and password. The second is used by the system to allow access. The third is used by the system to deny access.

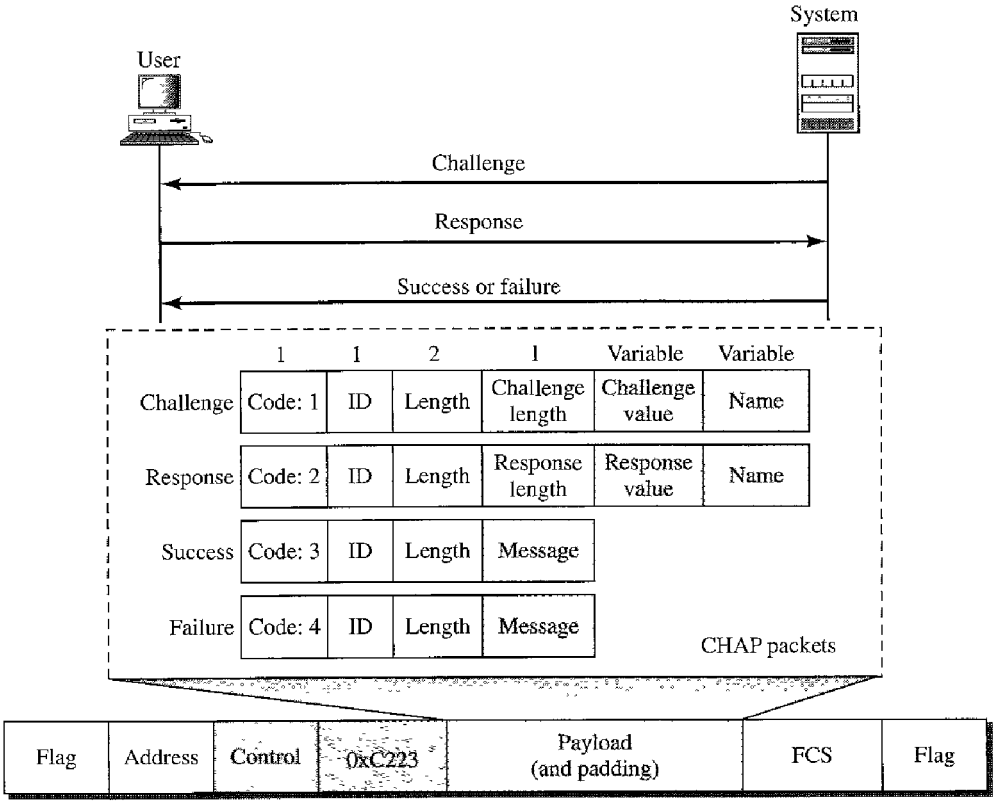
Figure 11.36 *PAP packets encapsulated in a PPP frame*



CHAP The **Challenge Handshake Authentication Protocol (CHAP)** is a three-way hand-shaking authentication protocol that provides greater security than PAP. In this method, the password is kept secret; it is never sent online.

1. The system sends the user a challenge packet containing a challenge value, usually a few bytes.
2. The user applies a predefined function that takes the challenge value and the user's own password and creates a result. The user sends the result in the response packet to the system.
3. The system does the same. It applies the same function to the password of the user (known to the system) and the challenge value to create a result. If the result created is the same as the result sent in the response packet, access is granted; otherwise, it is denied. CHAP is more secure than PAP, especially if the system continuously changes the challenge value. Even if the intruder learns the challenge value and the result, the password is still secret. Figure 11.37 shows the packets and how they are used.

Figure 11.37 *CHAP packets encapsulated in a PPP frame*



CHAP packets are encapsulated in the PPP frame with the protocol value C223 in hexadecimal. There are four CHAP packets: challenge, response, success, and failure. The first packet is used by the system to send the challenge value. The second is used by the user to return the result of the calculation. The third is used by the system to allow access to the system. The fourth is used by the system to deny access to the system.

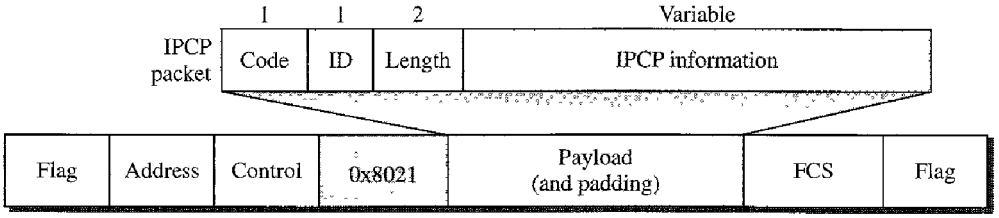
Network Control Protocols

PPP is a multiple-network layer protocol. It can carry a network layer data packet from protocols defined by the Internet, OSI, Xerox, DECnet, AppleTalk, Novel, and so on.

To do this, PPP has defined a specific Network Control Protocol for each network protocol. For example, IPCP (Internet Protocol Control Protocol) configures the link for carrying IP data packets. Xerox CP does the same for the Xerox protocol data packets, and so on. Note that none of the NCP packets carry network layer data; they just configure the link at the network layer for the incoming data.

IPCP One NCP protocol is the **Internet Protocol Control Protocol (IPCP).** This protocol configures the link used to carry IP packets in the Internet. IPCP is especially of interest to us. The format of an IPCP packet is shown in Figure 11.38. Note that the value of the protocol field in hexadecimal is 8021.

Figure 11.38 *IPCP packet encapsulated in PPP frame*



IPCP defines seven packets, distinguished by their code values, as shown in Table 11.4.

Table 11.4 *Code value for IPCP packets*

Code	*IPCP Packet*
0x01	Configure-request
0x02	Configure-ack
0x03	Configure-nak
0x04	Configure-reject
0x05	Terminate-request
0x06	Terminate-ack
0x07	Code-reject

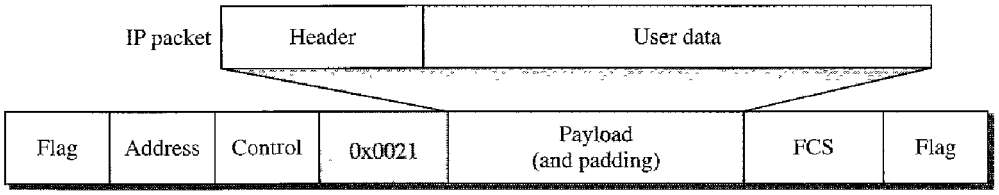
Other Protocols There are other NCP protocols for other network layer protocols. The OSI Network Layer Control Protocol has a protocol field value of 8023; the Xerox NS IDP Control Protocol has a protocol field value of 8025; and so on. The value of the code and the format of the packets for these other protocols are the same as shown in Table 11.4.

Data from the Network Layer

After the network layer configuration is completed by one of the NCP protocols, the users can exchange data packets from the network layer. Here again, there are different

protocol fields for different network layers. For example, if PPP is carrying data from the IP network layer, the field value is 0021 (note that the three rightmost digits are the same as for IPCP). If PPP is carrying data from the OSI network layer, the value of the protocol field is 0023, and so on. Figure 11.39 shows the frame for IP.

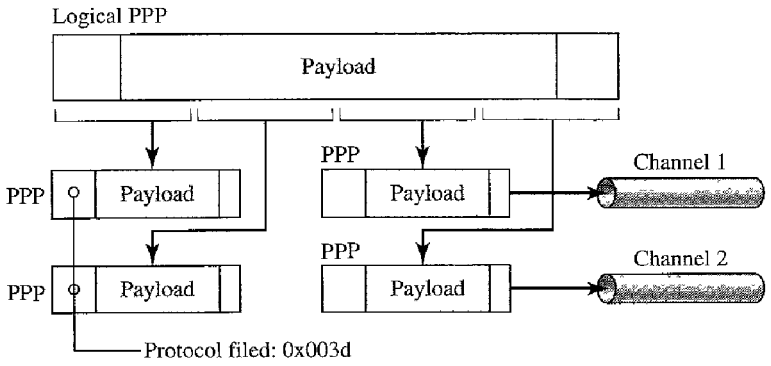
Figure 11.39 *IP datagram encapsulated in a PPP frame*



Multilink PPP

PPP was originally designed for a single-channel point-to-point physical link. The availability of multiple channels in a single point-to-point link motivated the development of Multilink PPP. In this case, a logical PPP frame is divided into several actual PPP frames. A segment of the logical frame is carried in the payload of an actual PPP frame, as shown in Figure 11.40. To show that the actual PPP frame is carrying a fragment of a

Figure 11.40 *Multilink PPP*

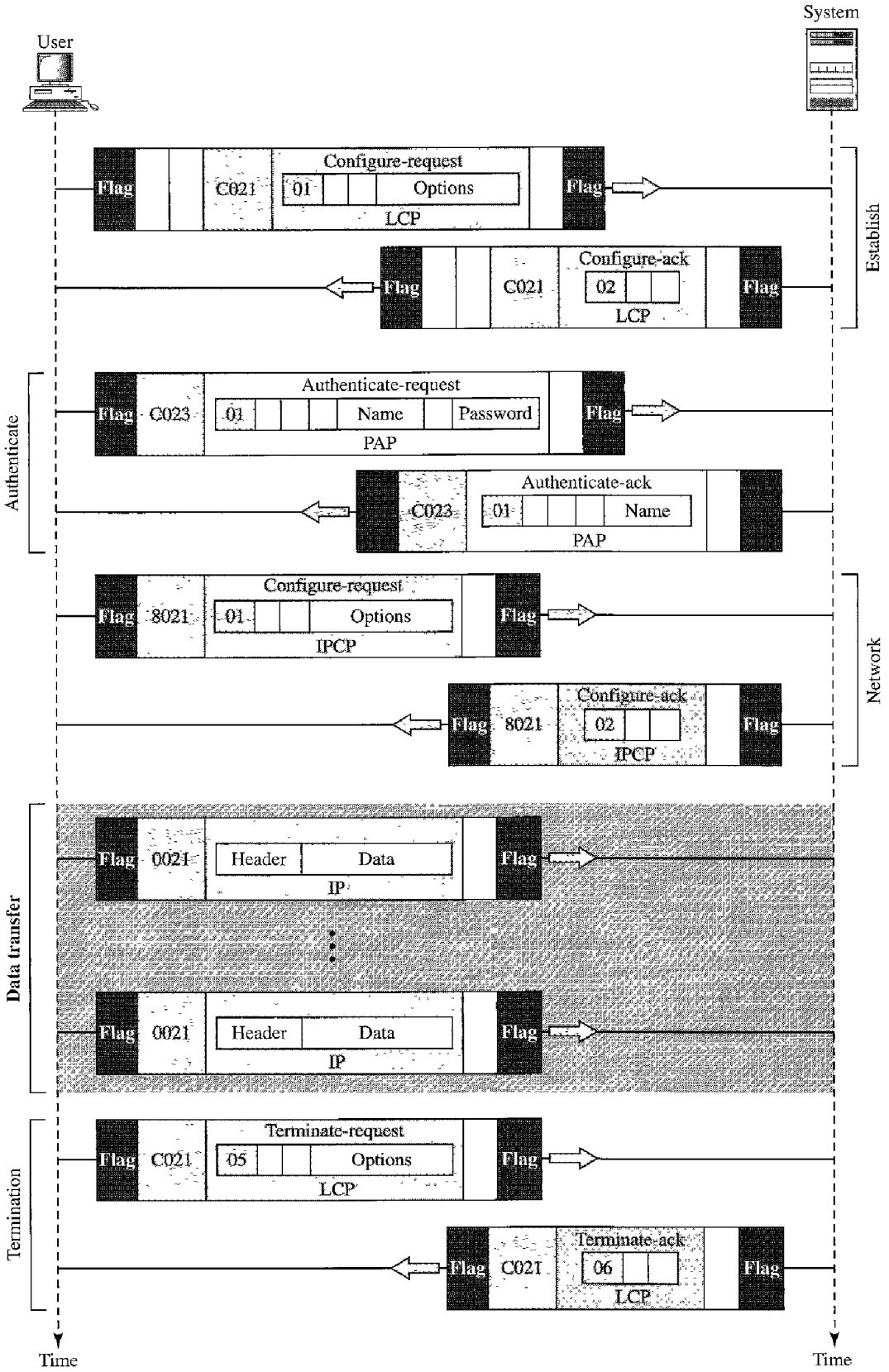


logical PPP frame, the protocol field is set to 0x003d. This new development adds complexity. For example, a sequence number needs to be added to the actual PPP frame to show a fragment's position in the logical frame.

Example 11.12

Let us go through the phases followed by a network layer packet as it is transmitted through a PPP connection. Figure 11.41 shows the steps. For simplicity, we assume unidirectional movement of data from the user site to the system site (such as sending an e-mail through an ISP).

Figure 11.41 *An example*



The first two frames show link establishment. We have chosen two options (not shown in the figure): using PAP for authentication and suppressing the address control fields. Frames 3 and 4 are for authentication. Frames 5 and 6 establish the network layer connection using IPCP.

The next several frames show that some IP packets are encapsulated in the PPP frame. The system (receiver) may have been running several network layer protocols, but it knows that the incoming data must be delivered to the IP protocol because the NCP protocol used before the data transfer was IPCP.

After data transfer, the user then terminates the data link connection, which is acknowledged by the system. Of course the user or the system could have chosen to terminate the network layer IPCP and keep the data link layer running if it wanted to run another NCP protocol.

The example is trivial, but it points out the similarities of the packets in LCP, AP, and NCP. It also shows the protocol field values and code numbers for particular protocols.

11.8 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

A discussion of data link control can be found in [GW04], Chapter 3 of [Tan03], Chapter 7 of [Sta04], Chapter 12 of [Kes97], and Chapter 2 of [PD03]. More advanced materials can be found in [KMK04].

11.9 KEY TERMS

acknowledgment (ACK)
asynchronous balanced mode (ABM)
automatic repeat request (ARQ)
bandwidth-delay product
bit-oriented protocol
bit stuffing
byte stuffing
Challenge Handshake Authentication Protocol (CHAP)
character-oriented protocol
data link control
error control
escape character (ESC)
event
fixed-size framing
flag
flow control
framing
Go-Back-*N* ARQ Protocol
High-level Data Link Control (HDLC)
information frame (I-frame)
Internet Protocol Control Protocol (IPCP)
Link Control Protocol (LCP)
negative acknowledgment (NAK)
noiseless channel
noisy channel
normal response mode (NRM)
Password Authentication Protocol (PAP)
piggybacking
pipelining
Point-to-Point Protocol (PPP)
primary station

receive sliding window
secondary station
Selective Repeat ARQ Protocol
send sliding window
sequence number
Simplest Protocol
sliding window
Stop-and-Wait ARQ Protocol
Stop-and-Wait Protocol
supervisory frame (S-frame)
transition phase
unnumbered frame (U-frame)
variable-size framing

11.10 SUMMARY

- Data link control deals with the design and procedures for communication between two adjacent nodes: node-to-node communication.
- Framing in the data link layer separates a message from one source to a destination, or from other messages going from other sources to other destinations,
- Frames can be of fixed or variable size. In fixed-size framing, there is no need for defining the boundaries of frames; in variable-size framing, we need a delimiter (flag) to define the boundary of two frames.
- Variable-size framing uses two categories of protocols: byte-oriented (or character-oriented) and bit-oriented. In a byte-oriented protocol, the data section of a frame is a sequence of bytes; in a bit-oriented protocol, the data section of a frame is a sequence of bits.
- In byte-oriented (or character-oriented) protocols, we use byte stuffing; a special byte added to the data section of the frame when there is a character with the same pattern as the flag.
- In bit-oriented protocols, we use bit stuffing; an extra 0 is added to the data section of the frame when there is a sequence of bits with the same pattern as the flag.
- Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment. Error control refers to methods of error detection and correction.
- For the noiseless channel, we discussed two protocols: the Simplest Protocol and the Stop-and-Wait Protocol. The first protocol has neither flow nor error control; the second has no error control. In the Simplest Protocol, the sender sends its frames one after another with no regards to the receiver. In the Stop-and-Wait Protocol, the sender sends one frame, stops until it receives confirmation from the receiver, and then sends the next frame.
- For the noisy channel, we discussed three protocols: Stop-and-Wait ARQ, Go-Back-*N*, and Selective Repeat ARQ. The Stop-and-Wait ARQ Protocol, adds a simple error control mechanism to the Stop-and-Wait Protocol. In the Go-Back-*N* ARQ Protocol, we can send several frames before receiving acknowledgments, improving the efficiency of transmission. In the Selective Repeat ARQ protocol we avoid unnecessary transmission by sending only frames that are corrupted.
- Both Go-Back-*N* and Selective-Repeat Protocols use a sliding window. In Go-Back-*N* ARQ, if m is the number of bits for the sequence number, then the size of

the send window must be less than 2^m; the size of the receiver window is always 1. In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^m.

- A technique called piggybacking is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about frames from B; when a frame is carrying data from B to A, it can also carry control information about frames from A.
- High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. However, the most common protocols for point-to-point access is the Point-to-Point Protocol (PPP), which is a byte-oriented protocol.

11.11 PRACTICE SET

Review Questions

1. Briefly describe the services provided by the data link layer.
2. Define framing and the reason for its need.
3. Compare and contrast byte-oriented and bit-oriented protocols. Which category has been popular in the past (explain the reason)? Which category is popular now (explain the reason)?
4. Compare and contrast byte-stuffing and bit-stuffing. Which technique is used in byte-oriented protocols? Which technique is used in bit-oriented protocols?
5. Compare and contrast flow control and error control.
6. What are the two protocols we discussed for noiseless channels in this chapter?
7. What are the three protocols we discussed for noisy channels in this chapter?
8. Explain the reason for moving from the Stop-and-Wait ARQ Protocol to the Go-Back-*N* ARQ Protocol.
9. Compare and contrast the Go-Back-*N* ARQ Protocol with Selective-Repeat ARQ.
10. Compare and contrast HDLC with PPP. Which one is byte-oriented; which one is bit-oriented?
11. Define piggybacking and its usefulness.
12. Which of the protocols described in this chapter utilize pipelining?

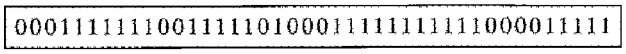
Exercises

13. Byte-stuff the data in Figure 11.42.

Figure 11.42 *Exercise 13*

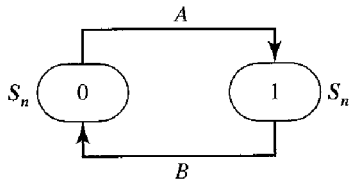
14. Bit-stuff the data in Figure 11.43.

Figure 11.43 *Exercise 14*



15. Design two simple algorithms for byte-stuffing. The first adds bytes at the sender; the second removes bytes at the receiver.
16. Design two simple algorithms for bit-stuffing. The first adds bits at the sender; the second removes bits at the receiver.
17. A sender sends a series of packets to the same destination using 5-bit sequence numbers. If the sequence number starts with 0, what is the sequence number after sending 100 packets?
18. Using 5-bit sequence numbers, what is the maximum size of the send and receive windows for each of the following protocols?
    a. Stop-and-Wait ARQ
    b. Go-Back-*N* ARQ
    c. Selective-Repeat ARQ
19. Design a bidirectional algorithm for the Simplest Protocol using piggybacking. Note that the both parties need to use the same algorithm.
20. Design a bidirectional algorithm for the Stop-and-Wait Protocol using piggybacking. Note that both parties need to use the same algorithm.
21. Design a bidirectional algorithm for the Stop-and-Wait ARQ Protocol using piggybacking. Note that both parties need to use the same algorithm.
22. Design a bidirectional algorithm for the Go-Back-*N* ARQ Protocol using piggybacking. Note that both parties need to use the same algorithm.
23. Design a bidirectional algorithm for the Selective-Repeat ARQ Protocol using piggybacking. Note that both parties need to use the same algorithm.
24. Figure 11.44 shows a state diagram to simulate the behavior of Stop-and-Wait ARQ at the sender site.

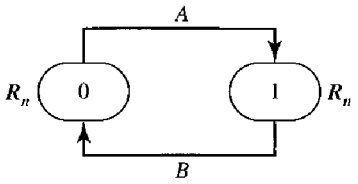
Figure 11.44 *Exercise 24*



The states have a value of S_n (0 or 1). The arrows shows the transitions. Explain the events that cause the two transitions labeled A and B.

25. Figure 11.45 shows a state diagram to simulate the behavior of Stop-and-Wait ARQ at the receiver site.

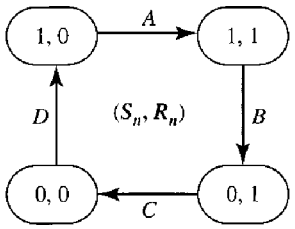
Figure 11.45 *Exercise 25*



The states have a value of R_n (0 or 1). The arrows shows the transitions. Explain the events that cause the two transitions labeled A and B.

26. In Stop-and-Wait ARQ, we can combine the state diagrams of the sender and receiver in Exercises 24 and 25. One state defines the combined values of R_n and S_n. This means that we can have four states, each defined by (x, y), where x defines the value of S_n and y defines the value of R_n. In other words, we can have the four states shown in Figure 11.46. Explain the events that cause the four transitions labeled A, B, C, and D.

Figure 11.46 *Exercise 26*



27. The timer of a system using the Stop-and-Wait ARQ Protocol has a time-out of 6 ms. Draw the flow diagram similar to Figure 11.11 for four frames if the round trip delay is 4 ms. Assume no data frame or control frame is lost or damaged.
28. Repeat Exercise 27 if the time-out is 4 ms and the round trip delay is 6.
29. Repeat Exercise 27 if the first frame (frame 0) is lost.
30. A system uses the Stop-and-Wait ARQ Protocol. If each packet carries 1000 bits of data, how long does it take to send 1 million bits of data if the distance between the sender and receiver is 5000 Km and the propagation speed is 2×10^8 m? Ignore transmission, waiting, and processing delays. We assume no data or control frame is lost or damaged.
31. Repeat Exercise 30 using the Go-back-N ARQ Protocol with a window size of 7. Ignore the overhead due to the header and trailer.
32. Repeat Exercise 30 using the Selective-Repeat ARQ Protocol with a window size of 4. Ignore the overhead due to the header and the trailer.

CHAPTER 12

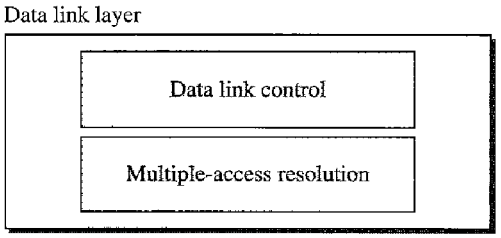
Multiple Access

In Chapter 11 we discussed data link control, a mechanism which provides a link with reliable communication. In the protocols we described, we assumed that there is an available dedicated link (or channel) between the sender and the receiver. This assumption may or may not be true. If, indeed, we have a dedicated link, as when we connect to the Internet using PPP as the data link control protocol, then the assumption is true and we do not need anything else.

On the other hand, if we use our cellular phone to connect to another cellular phone, the channel (the band allocated to the vendor company) is not dedicated. A person a few feet away from us may be using the same channel to talk to her friend.

We can consider the data link layer as two sublayers. The upper sublayer is responsible for data link control, and the lower sublayer is responsible for resolving access to the shared media. If the channel is dedicated, we do not need the lower sublayer. Figure 12.1 shows these two sublayers in the data link layer.

Figure 12.1 *Data link layer divided into two functionality-oriented sublayers*



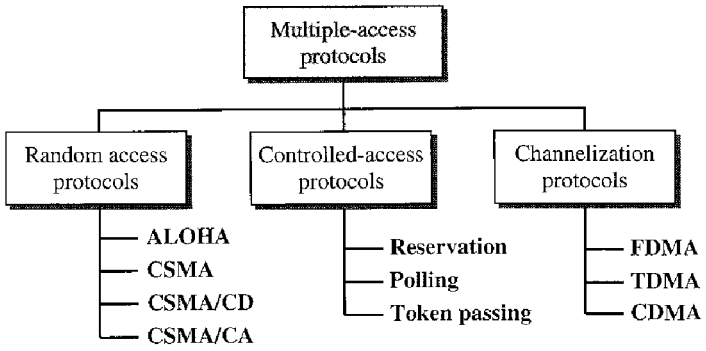
We will see in Chapter 13 that the IEEE has actually made this division for LANs. The upper sublayer that is responsible for flow and error control is called the logical link control (LLC) layer; the lower sublayer that is mostly responsible for multiple-access resolution is called the media access control (MAC) layer.

When nodes or stations are connected and use a common link, called a multipoint or broadcast link, we need a multiple-access protocol to coordinate access to the link. The problem of controlling the access to the medium is similar to the rules of speaking

in an assembly. The procedures guarantee that the right to speak is upheld and ensure that two people do not speak at the same time, do not interrupt each other, do not monopolize the discussion, and so on.

The situation is similar for multipoint networks. Many formal protocols have been devised to handle access to a shared link. We categorize them into three groups. Protocols belonging to each group are shown in Figure 12.2.

Figure 12.2 *Taxonomy of multiple-access protocols discussed in this chapter*



12.1 RANDOM ACCESS

In **random access** or **contention** methods, no station is superior to another station and none is assigned the control over another. No station permits, or does not permit, another station to send. At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send. This decision depends on the state of the medium (idle or busy). In other words, each station can transmit when it desires on the condition that it follows the predefined procedure, including the testing of the state of the medium.

Two features give this method its name. First, there is no scheduled time for a station to transmit. Transmission is random among the stations. That is why these methods are called *random access*. Second, no rules specify which station should send next. Stations compete with one another to access the medium. That is why these methods are also called *contention* methods.

In a random access method, each station has the right to the medium without being controlled by any other station. However, if more than one station tries to send, there is an access conflict—**collision**—and the frames will be either destroyed or modified. To avoid access conflict or to resolve it when it happens, each station follows a procedure that answers the following questions:

- When can the station access the medium?
- What can the station do if the medium is busy?
- How can the station determine the success or failure of the transmission?
- What can the station do if there is an access conflict?

The random access methods we study in this chapter have evolved from a very interesting protocol known as ALOHA, which used a very simple procedure called **multiple access (MA).** The method was improved with the addition of a procedure that forces the station to sense the medium before transmitting. This was called carrier sense multiple access. This method later evolved into two parallel methods: **carrier sense multiple access with collision detection (CSMA/CD)** and **carrier sense multiple access with collision avoidance (CSMA/CA).** CSMA/CD tells the station what to do when a collision is detected. CSMA/CA tries to avoid the collision.

ALOHA

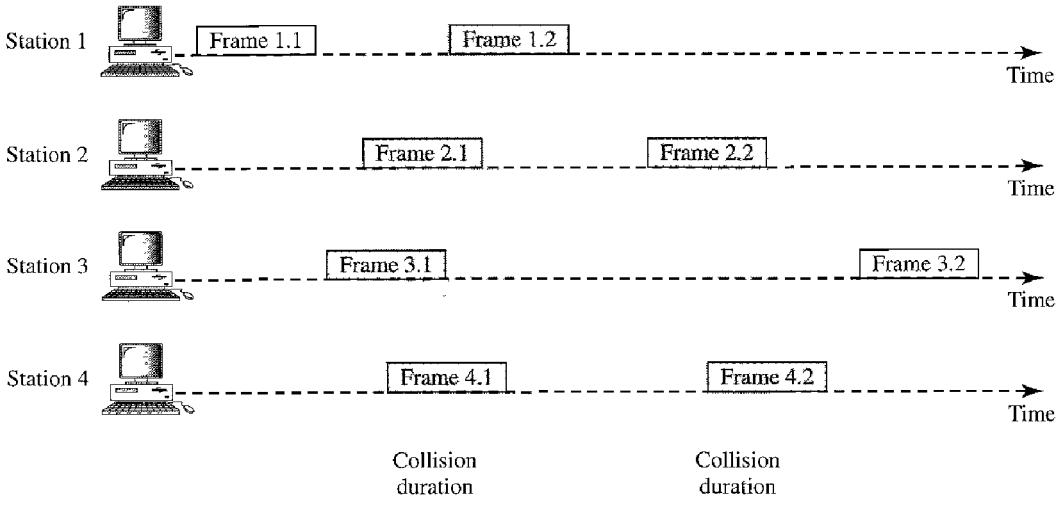
ALOHA, the earliest random access method, was developed at the University of Hawaii in early 1970. It was designed for a radio (wireless) LAN, but it can be used on any shared medium.

It is obvious that there are potential collisions in this arrangement. The medium is shared between the stations. When a station sends data, another station may attempt to do so at the same time. The data from the two stations collide and become garbled.

Pure ALOHA

The original ALOHA protocol is called **pure ALOHA.** This is a simple, but elegant protocol. The idea is that each station sends a frame whenever it has a frame to send. However, since there is only one channel to share, there is the possibility of collision between frames from different stations. Figure 12.3 shows an example of frame collisions in pure ALOHA.

Figure 12.3 *Frames in a pure ALOHA network*



There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. The figure shows that each station sends two frames; there are a total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the shared channel. Figure 12.3 shows that only

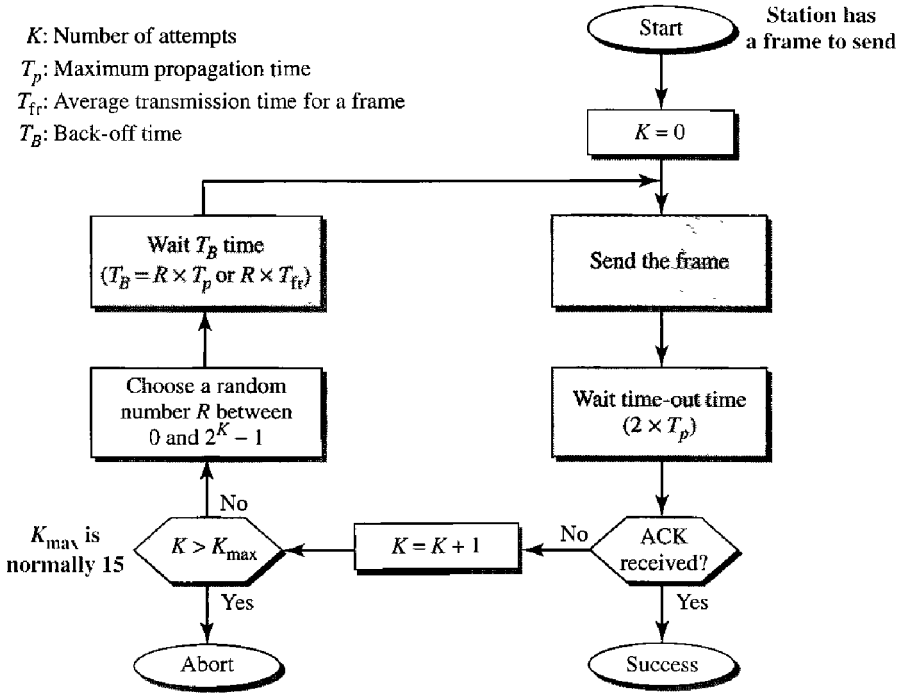
two frames survive: frame 1.1 from station 1 and frame 3.2 from station 3. We need to mention that even if one bit of a frame coexists on the channel with one bit from another frame, there is a collision and both will be destroyed.

It is obvious that we need to resend the frames that have been destroyed during transmission. The pure ALOHA protocol relies on acknowledgments from the receiver. When a station sends a frame, it expects the receiver to send an acknowledgment. If the acknowledgment does not arrive after a time-out period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame.

A collision involves two or more stations. If all these stations try to resend their frames after the time-out, the frames will collide again. Pure ALOHA dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the back-off time T_B.

Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts K_{max}, a station must give up and try later. Figure 12.4 shows the procedure for pure ALOHA based on the above strategy.

Figure 12.4 *Procedure for pure ALOHA protocol*



The time-out period is equal to the maximum possible round-trip propagation delay, which is twice the amount of time required to send a frame between the two most widely separated stations ($2 \times T_p$). The back-off time T_B is a random value that normally depends on K (the number of attempted unsuccessful transmissions). The formula for T_B depends on the implementation. One common formula is the **binary exponential back-off.** In this

method, for each retransmission, a multiplier in the range 0 to $2^K - 1$ is randomly chosen and multiplied by T_p (maximum propagation time) or T_{fr} (the average time required to send out a frame) to find T_B. Note that in this procedure, the range of the random numbers increases after each collision. The value of K_{max} is usually chosen as 15.

Example 12.1

The stations on a wireless ALOHA network are a maximum of 600 km apart. If we assume that signals propagate at 3×10^8 m/s, we find $T_p = (600 \times 10^5) / (3 \times 10^8) = 2$ ms. Now we can find the value of T_B for different values of K.

a. For $K = 1$, the range is {0, 1}. The station needs to generate a random number with a value of 0 or 1. This means that T_B is either 0 ms (0×2) or 2 ms (1×2), based on the outcome of the random variable.

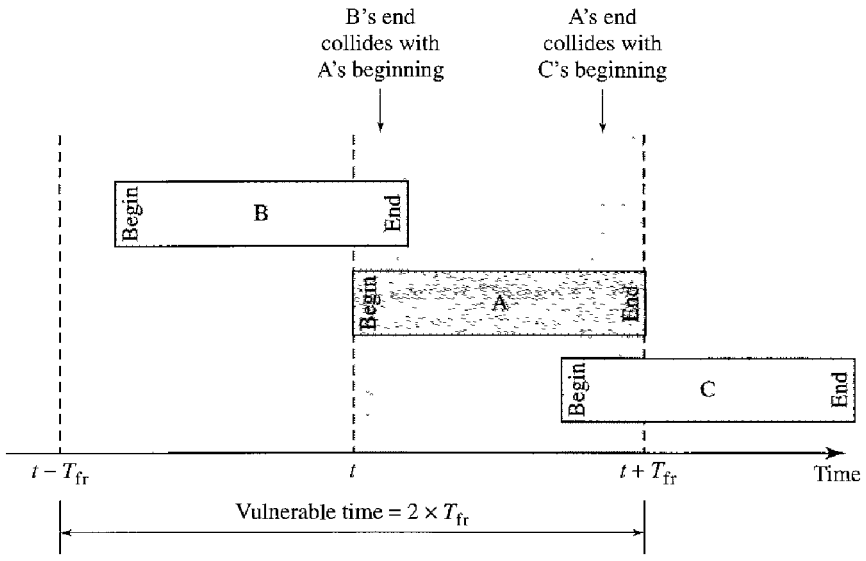
b. For $K = 2$, the range is {0, 1, 2, 3}. This means that T_B can be 0, 2, 4, or 6 ms, based on the outcome of the random variable.

c. For $K = 3$, the range is {0, 1, 2, 3, 4, 5, 6, 7}. This means that T_B can be 0, 2, 4, . . . , 14 ms, based on the outcome of the random variable.

d. We need to mention that if $K > 10$, it is normally set to 10.

Vulnerable time Let us find the length of time, the **vulnerable time,** in which there is a possibility of collision. We assume that the stations send fixed-length frames with each frame taking T_{fr} s to send. Figure 12.5 shows the vulnerable time for station A.

Figure 12.5 *Vulnerable time for pure ALOHA protocol*



Station A sends a frame at time t. Now imagine station B has already sent a frame between $t - T_{fr}$ and t. This leads to a collision between the frames from station A and station B. The end of B's frame collides with the beginning of A's frame. On the other hand, suppose that station C sends a frame between t and $t + T_{fr}$. Here, there is a collision between frames from station A and station C. The beginning of C's frame collides with the end of A's frame.

Looking at Figure 12.5, we see that the vulnerable time, during which a collision may occur in pure ALOHA, is 2 times the frame transmission time.

$$\textbf{Pure ALOHA vulnerable time} = 2 \times T_{fr}$$

Example 12.2

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the requirement to make this frame collision-free?

Solution

Average frame transmission time T_{fr} is 200 bits/200 kbps or 1 ms. The vulnerable time is 2×1 ms = 2 ms. This means no station should send later than 1 ms before this station starts transmission and no station should start sending during the one 1-ms period that this station is sending.

Throughput Let us call G the average number of frames generated by the system during one frame transmission time. Then it can be proved that the average number of successful transmissions for pure ALOHA is $S = G \times e^{-2G}$. The maximum throughput S_{max} is 0.184, for $G = \frac{1}{2}$. In other words, if one-half a frame is generated during one frame transmission time (in other words, one frame during two frame transmission times), then 18.4 percent of these frames reach their destination successfully. This is an expected result because the vulnerable time is 2 times the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other stations generate a frame during this time), the frame will reach its destination successfully.

The throughput for pure ALOHA is $S = G \times e^{-2G}$.
The maximum throughput $S_{max} = 0.184$ when $G = (1/2)$.

Example 12.3

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the throughput if the system (all stations together) produces

a. 1000 frames per second
b. 500 frames per second
c. 250 frames per second

Solution

The frame transmission time is 200/200 kbps or 1 ms.

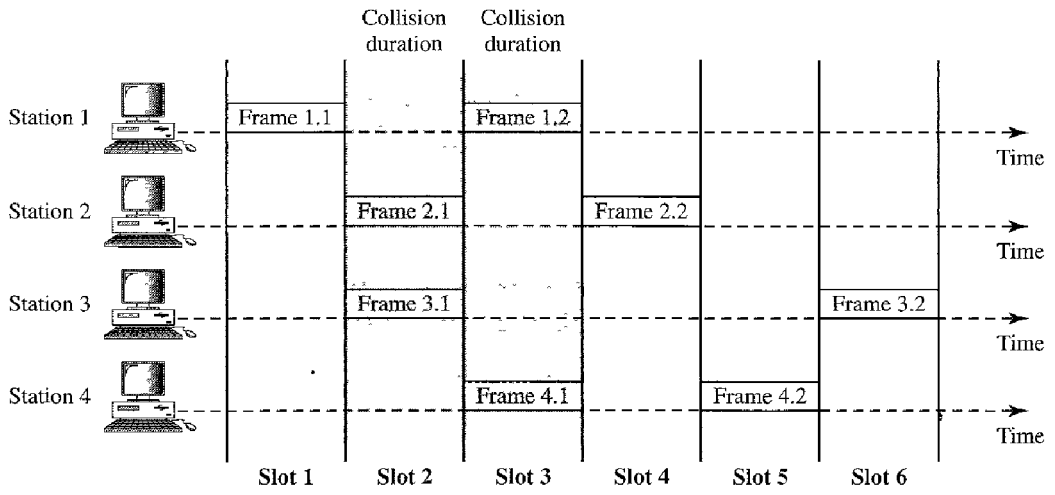
a. If the system creates 1000 frames per second, this is 1 frame per millisecond. The load is 1. In this case $S = G \times e^{-2G}$ or $S = 0.135$ (13.5 percent). This means that the throughput is $1000 \times 0.135 = 135$ frames. Only 135 frames out of 1000 will probably survive.
b. If the system creates 500 frames per second, this is (1/2) frame per millisecond. The load is (1/2). In this case $S = G \times e^{-2G}$ or $S = 0.184$ (18.4 percent). This means that the throughput is $500 \times 0.184 = 92$ and that only 92 frames out of 500 will probably survive. Note that this is the maximum throughput case, percentagewise.
c. If the system creates 250 frames per second, this is (1/4) frame per millisecond. The load is (1/4). In this case $S = G \times e^{-2G}$ or $S = 0.152$ (15.2 percent). This means that the throughput is $250 \times 0.152 = 38$. Only 38 frames out of 250 will probably survive.

Slotted ALOHA

Pure ALOHA has a vulnerable time of $2 \times T_{fr}$. This is so because there is no rule that defines when the station can send. A station may send soon after another station has started or soon before another station has finished. Slotted ALOHA was invented to improve the efficiency of pure ALOHA.

In **slotted ALOHA** we divide the time into slots of T_{fr} s and force the station to send only at the beginning of the time slot. Figure 12.6 shows an example of frame collisions in slotted ALOHA.

Figure 12.6 *Frames in a slotted ALOHA network*



Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot. This means that the station which started at the beginning of this slot has already finished sending its frame. Of course, there is still the possibility of collision if two stations try to send at the beginning of the same time slot. However, the vulnerable time is now reduced to one-half, equal to T_{fr}. Figure 12.7 shows the situation.

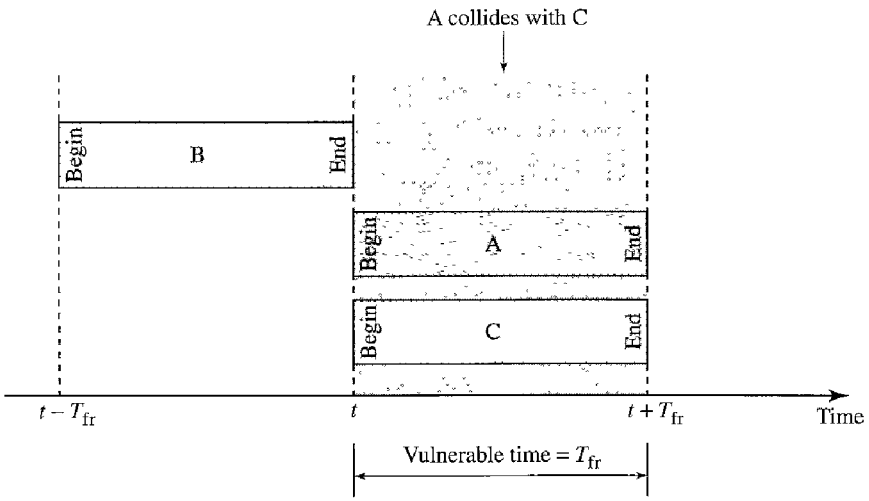
Figure 12.7 shows that the vulnerable time for slotted ALOHA is one-half that of pure ALOHA.

$$\textbf{Slotted ALOHA vulnerable time} = T_{fr}$$

Throughput It can be proved that the average number of successful transmissions for slotted ALOHA is $S = G \times e^{-G}$. The maximum throughput S_{max} is 0.368, when $G = 1$. In other words, if a frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination successfully. This result can be expected because the vulnerable time is equal to the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other station generates a frame during this time), the frame will reach its destination successfully.

The throughput for slotted ALOHA is $S = G \times e^{-G}$.
The maximum throughput $S_{max} = 0.368$ when $G = 1$.

Figure 12.7 *Vulnerable time for slotted ALOHA protocol*



Example 12.4

A slotted ALOHA network transmits 200-bit frames using a shared channel with a 200-kbps bandwidth. Find the throughput if the system (all stations together) produces

a. 1000 frames per second
b. 500 frames per second
c. 250 frames per second

Solution

This situation is similar to the previous exercise except that the network is using slotted ALOHA instead of pure ALOHA. The frame transmission time is 200/200 kbps or 1 ms.

a. In this case G is 1. So $S = G \times e^{-G}$ or $S = 0.368$ (36.8 percent). This means that the throughput is $1000 \times 0.0368 = 368$ frames. Only 368 out of 1000 frames will probably survive. Note that this is the maximum throughput case, percentagewise.
b. Here G is $\frac{1}{2}$. In this case $S = G \times e^{-G}$ or $S = 0.303$ (30.3 percent). This means that the throughput is $500 \times 0.0303 = 151$. Only 151 frames out of 500 will probably survive.
c. Now G is $\frac{1}{4}$. In this case $S = G \times e^{-G}$ or $S = 0.195$ (19.5 percent). This means that the throughput is $250 \times 0.195 = 49$. Only 49 frames out of 250 will probably survive.

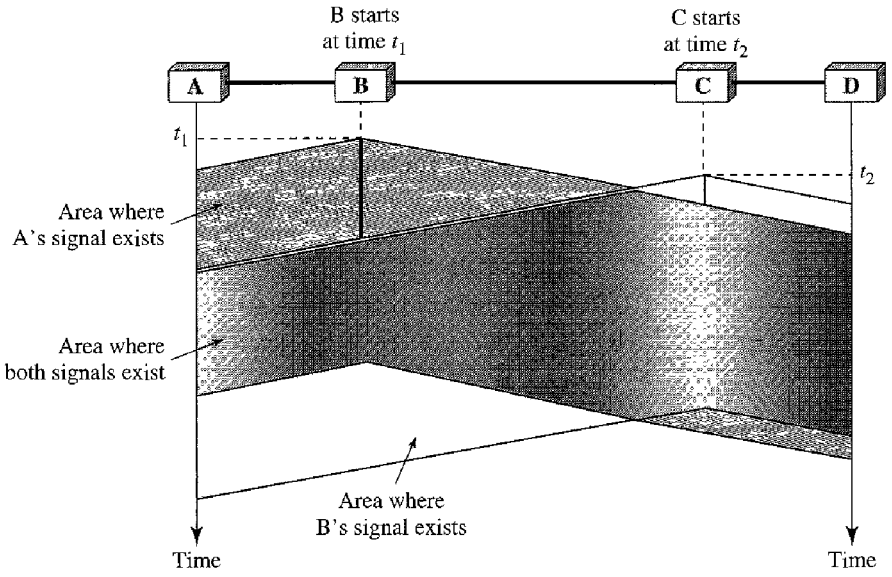
Carrier Sense Multiple Access (CSMA)

To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. **Carrier sense multiple access (CSMA)** requires that each station first listen to the medium (or check the state of the medium) before sending. In other words, CSMA is based on the principle "sense before transmit" or "listen before talk."

CSMA can reduce the possibility of collision, but it cannot eliminate it. The reason for this is shown in Figure 12.8, a space and time model of a CSMA network. Stations are connected to a shared channel (usually a dedicated medium).

The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time (although very short) for the first bit to reach every station

Figure 12.8 *Space/time model of the collision in CSMA*



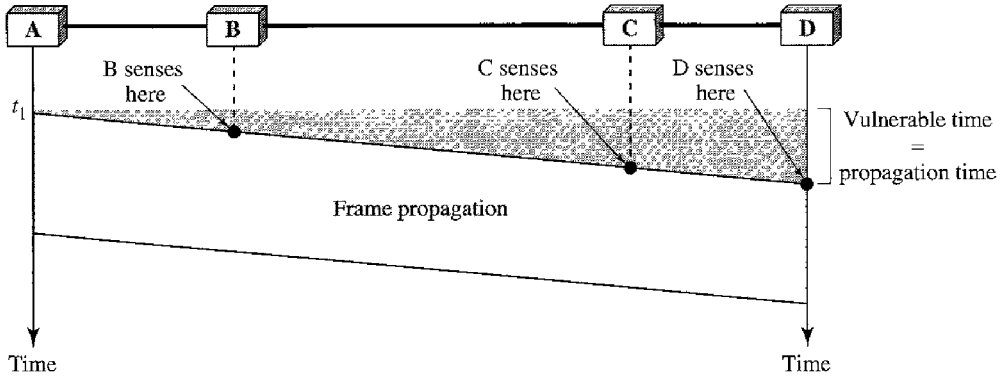
and for every station to sense it. In other words, a station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received.

At time t_1, station B senses the medium and finds it idle, so it sends a frame. At time t_2 ($t_2 > t_1$), station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed.

Vulnerable Time

The vulnerable time for CSMA is the **propagation time** T_p. This is the time needed for a signal to propagate from one end of the medium to the other. When a station sends a frame, and any other station tries to send a frame during this time, a collision will result. But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending. Figure 12.9 shows the worst

Figure 12.9 *Vulnerable time in CSMA*



case. The leftmost station A sends a frame at time t_1, which reaches the rightmost station D at time $t_1 + T_p$. The gray area shows the vulnerable area in time and space.

Persistence Methods

What should a station do if the channel is busy? What should a station do if the channel is idle? Three methods have been devised to answer these questions: the 1-persistent method, the nonpersistent method, and the *p*-persistent method. Figure 12.10 shows the behavior of three persistence methods when a station finds a channel busy.

Figure 12.10 *Behavior of three persistence methods*

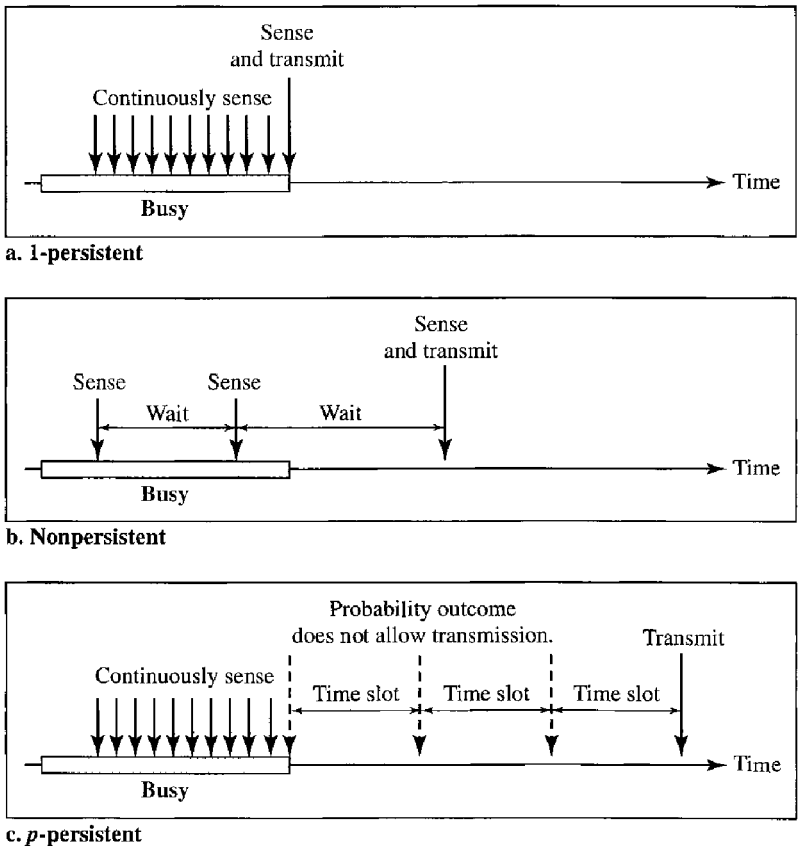
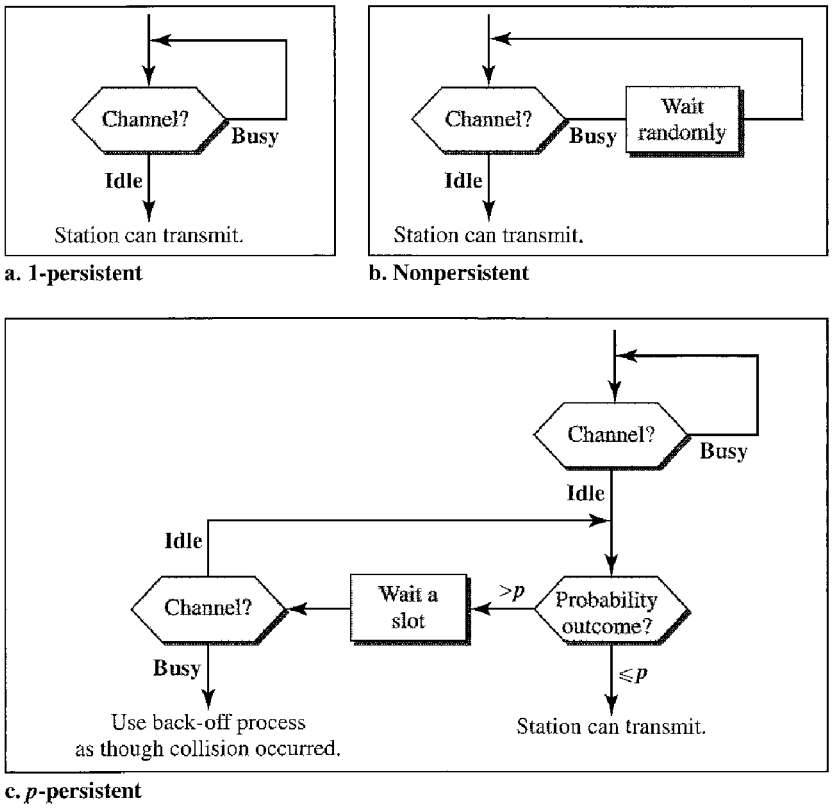


Figure 12.11 shows the flow diagrams for these methods.

1-Persistent The **1-persistent method** is simple and straightforward. In this method, after the station finds the line idle, it sends its frame immediately (with probability 1). This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately. We will see in Chapter 13 that Ethernet uses this method.

Nonpersistent In the **nonpersistent method,** a station that has a frame to send senses the line. If the line is idle, it sends immediately. If the line is not idle, it waits a

Figure 12.11 *Flow diagram for three persistence methods*



random amount of time and then senses the line again. The nonpersistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously. However, this method reduces the efficiency of the network because the medium remains idle when there may be stations with frames to send.

***p*-Persistent** The ***p*-persistent method** is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. The *p*-persistent approach combines the advantages of the other two strategies. It reduces the chance of collision and improves efficiency. In this method, after the station finds the line idle it follows these steps:

1. With probability p, the station sends its frame.
2. With probability $q = 1 - p$, the station waits for the beginning of the next time slot and checks the line again.
   a. If the line is idle, it goes to step 1.
   b. If the line is busy, it acts as though a collision has occurred and uses the back-off procedure.

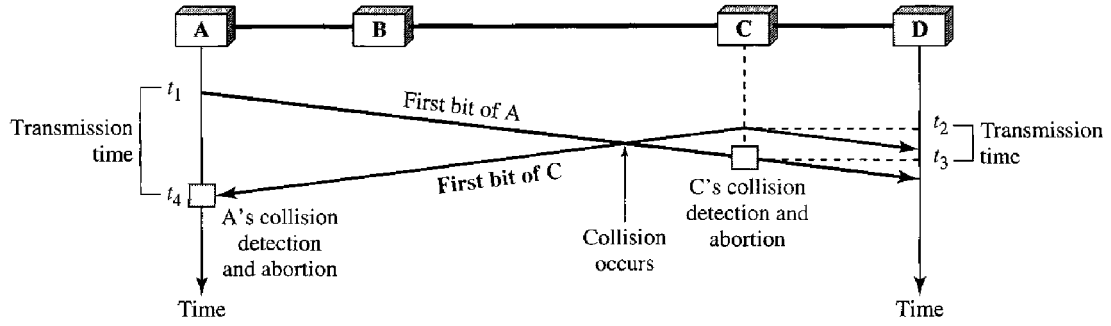
Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

The CSMA method does not specify the procedure following a collision. Carrier sense multiple access with collision detection (CSMA/CD) augments the algorithm to handle the collision.

In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.

To better understand CSMA/CD, let us look at the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In Figure 12.12, stations A and C are involved in the collision.

Figure 12.12 *Collision of the first bit in CSMA/CD*



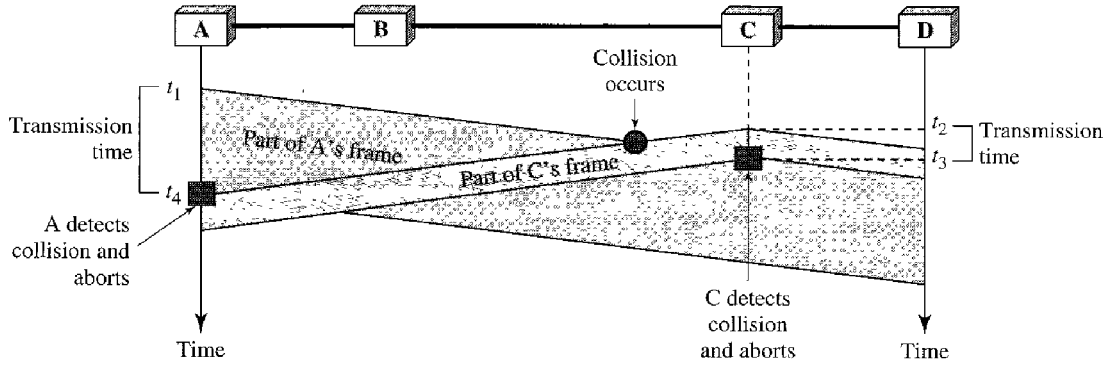
At time t_1, station A has executed its persistence procedure and starts sending the bits of its frame. At time t_2, station C has not yet sensed the first bit sent by A. Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right. The collision occurs sometime after time t_2. Station C detects a collision at time t_3 when it receives the first bit of A's frame. Station C immediately (or after a short time, but we assume immediately) aborts transmission. Station A detects collision at time t_4 when it receives the first bit of C's frame; it also immediately aborts transmission. Looking at the figure, we see that A transmits for the duration $t_4 - t_1$; C transmits for the duration $t_3 - t_2$. Later we show that, for the protocol to work, the length of any frame divided by the bit rate in this protocol must be more than either of these durations. At time t_4, the transmission of A's frame, though incomplete, is aborted; at time t_3, the transmission of B's frame, though incomplete, is aborted.

Now that we know the time durations for the two transmissions, we can show a more complete graph in Figure 12.13.

Minimum Frame Size

For CSMA/CD to work, we need a restriction on the frame size. Before sending the last bit of the frame, the sending station must detect a collision, if any, and abort the transmission. This is so because the station, once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection. Therefore, the frame transmission time T_{fr} must be at least two times the maximum propagation time T_p. To understand the reason, let us think about the worst-case scenario. If the two stations involved in a collision are the maximum distance apart, the signal from the first takes time T_p to reach the second, and the effect of the collision takes another time T_P to reach the first. So the requirement is that the first station must still be transmitting after $2T_p$.

Figure 12.13 *Collision and abortion in CSMA/CD*



Example 12.5

A network using CSMA/CD has a bandwidth of 10 Mbps. If the maximum propagation time (including the delays in the devices and ignoring the time needed to send a jamming signal, as we see later) is 25.6 μs, what is the minimum size of the frame?

Solution

The frame transmission time is $T_{fr} = 2 \times T_p = 51.2$ μs. This means, in the worst case, a station needs to transmit for a period of 51.2 μs to detect the collision. The minimum size of the frame is 10 Mbps × 51.2 μs = 512 bits or 64 bytes. This is actually the minimum size of the frame for Standard Ethernet, as we will see in Chapter 13.

Procedure

Now let us look at the flow diagram for CSMA/CD in Figure 12.14. It is similar to the one for the ALOHA protocol, but there are differences.

The first difference is the addition of the persistence process. We need to sense the channel before we start sending the frame by using one of the persistence processes we discussed previously (nonpersistent, 1-persistent, or *p*-persistent). The corresponding box can be replaced by one of the persistence processes shown in Figure 12.11.

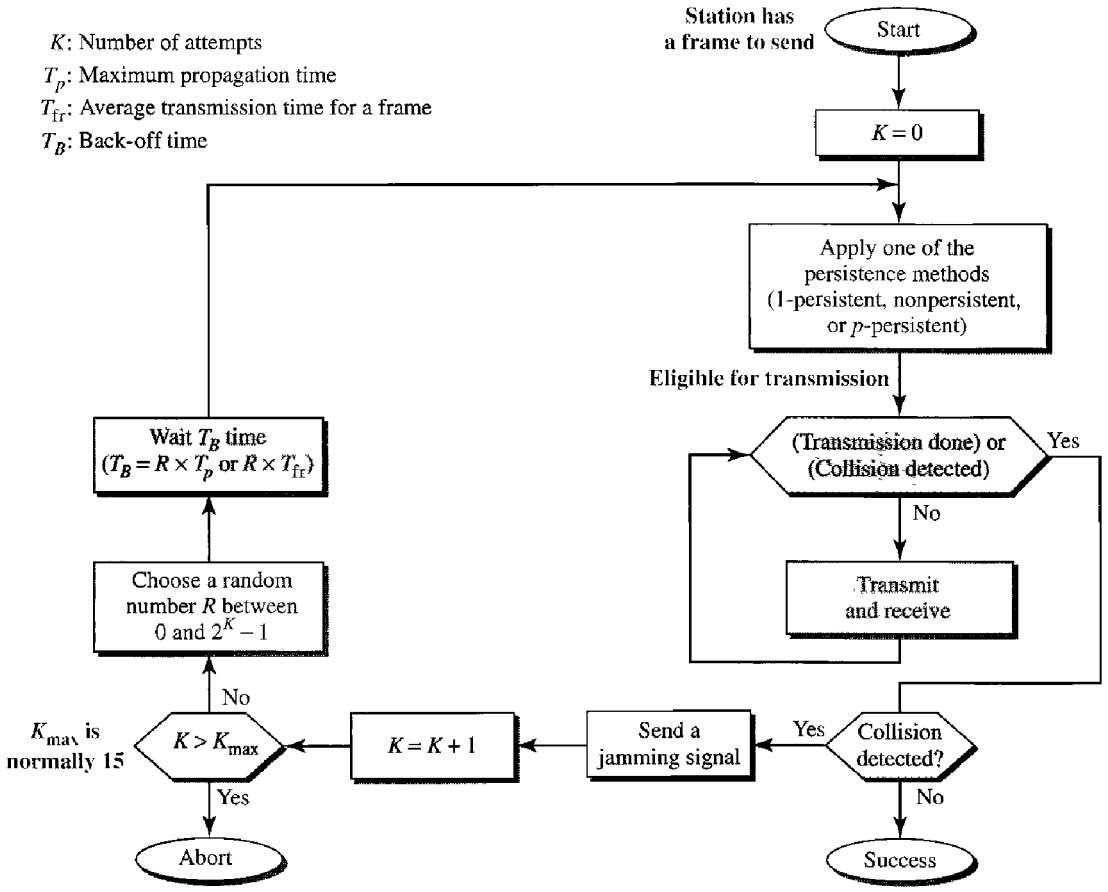
The second difference is the frame transmission. In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In CSMA/CD, transmission and collision detection is a continuous process. We do not send the entire frame and then look for a collision. The station transmits and receives continuously and simultaneously (using two different ports). We use a loop to show that transmission is a continuous process. We constantly monitor in order to detect one of two conditions: either transmission is finished or a collision is detected. Either event stops transmission. When we come out of the loop, if a collision has not been detected, it means that transmission is complete; the entire frame is transmitted. Otherwise, a collision has occurred.

The third difference is the sending of a short **jamming signal** that enforces the collision in case other stations have not yet sensed the collision.

Energy Level

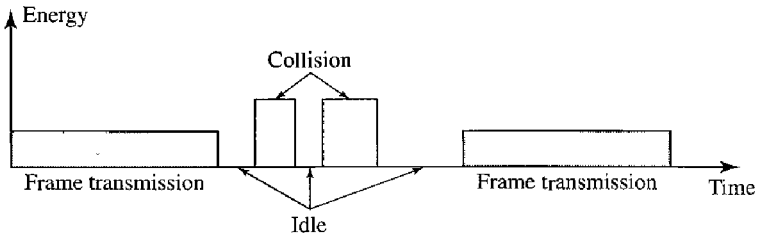
We can say that the level of energy in a channel can have three values: zero, normal, and abnormal. At the zero level, the channel is idle. At the normal level, a station has

Figure 12.14 *Flow diagram for the CSMA/CD*



successfully captured the channel and is sending its frame. At the abnormal level, there is a collision and the level of the energy is twice the normal level. A station that has a frame to send or is sending a frame needs to monitor the energy level to determine if the channel is idle, busy, or in collision mode. Figure 12.15 shows the situation.

Figure 12.15 *Energy level during transmission, idleness, or collision*



Throughput

The throughput of CSMA/CD is greater than that of pure or slotted ALOHA. The maximum throughput occurs at a different value of G and is based on the persistence method

and the value of p in the p-persistent approach. For 1-persistent method the maximum throughput is around 50 percent when $G = 1$. For nonpersistent method, the maximum throughput can go up to 90 percent when G is between 3 and 8.

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

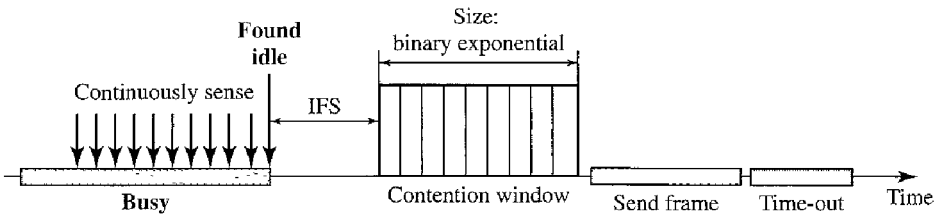
The basic idea behind CSMA/CD is that a station needs to be able to receive while transmitting to detect a collision. When there is no collision, the station receives one signal: its own signal. When there is a collision, the station receives two signals: its own signal and the signal transmitted by a second station. To distinguish between these two cases, the received signals in these two cases must be significantly different. In other words, the signal from the second station needs to add a significant amount of energy to the one created by the first station.

In a wired network, the received signal has almost the same energy as the sent signal because either the length of the cable is short or there are repeaters that amplify the energy between the sender and the receiver. This means that in a collision, the detected energy almost doubles.

However, in a wireless network, much of the sent energy is lost in transmission. The received signal has very little energy. Therefore, a collision may add only 5 to 10 percent additional energy. This is not useful for effective collision detection.

We need to avoid collisions on wireless networks because they cannot be detected. Carrier sense multiple access with collision avoidance (CSMA/CA) was invented for this network. Collisions are avoided through the use of CSMA/CA's three strategies: the interframe space, the contention window, and acknowledgments, as shown in Figure 12.16.

Figure 12.16 *Timing in CSMA/CA*



Interframe Space (IFS)

First, collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the **interframe space** or **IFS.** Even though the channel may appear idle when it is sensed, a distant station may have already started transmitting. The distant station's signal has not yet reached this station. The IFS time allows the front of the transmitted signal by the distant station to reach this station. If after the IFS time the channel is still idle, the station can send, but it still needs to wait a time equal to the contention time (described next). The IFS variable can also be used to prioritize

stations or frame types. For example, a station that is assigned a shorter IFS has a higher priority.

In CSMA/CA, the IFS can also be used to define the priority of a station or a frame.

Contention Window

The contention window is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential back-off strategy. This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time. This is very similar to the *p*-persistent method except that a random outcome defines the number of slots taken by the waiting station. One interesting point about the contention window is that the station needs to sense the channel after each time slot. However, if the station finds the channel busy, it does not restart the process; it just stops the timer and restarts it when the channel is sensed as idle. This gives priority to the station with the longest waiting time.

In CSMA/CA, if the station finds the channel busy, it does not restart the timer of the contention window; it stops the timer and restarts it when the channel becomes idle.

Acknowledgment

With all these precautions, there still may be a collision resulting in destroyed data. In addition, the data may be corrupted during the transmission. The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

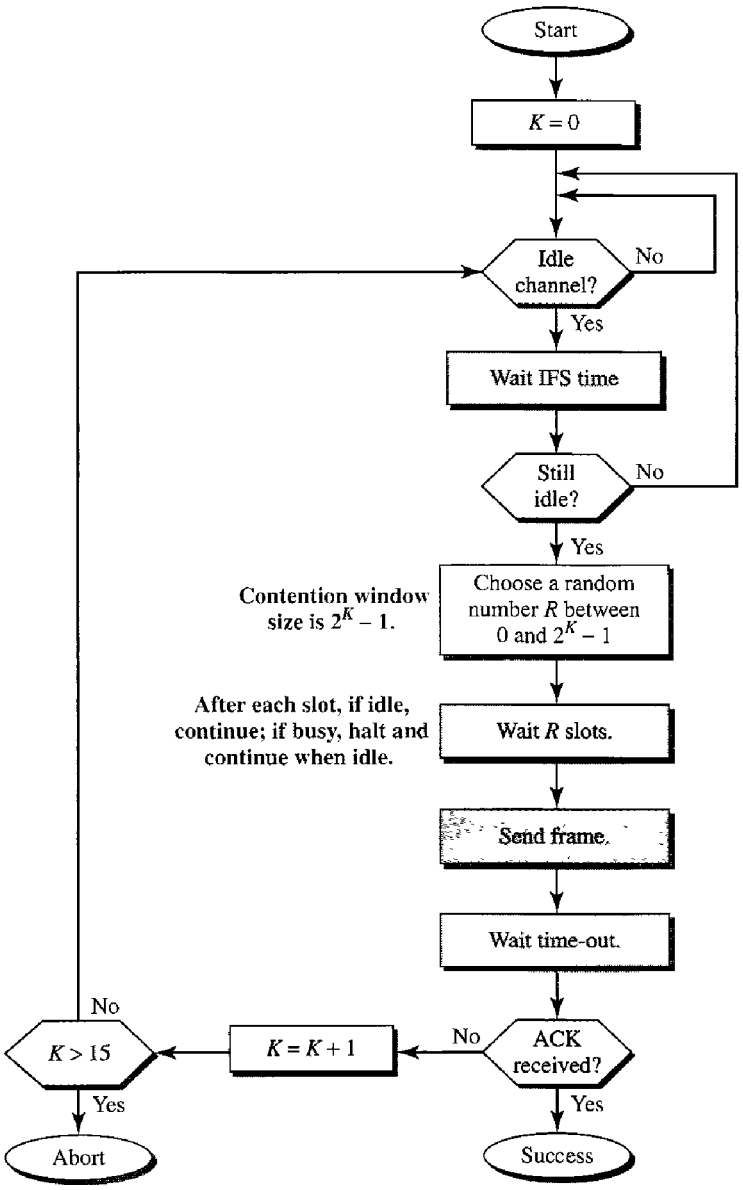
Procedure

Figure 12.17 shows the procedure. Note that the channel needs to be sensed before and after the IFS. The channel also needs to be sensed during the contention time. For each time slot of the contention window, the channel is sensed. If it is found idle, the timer continues; if the channel is found busy, the timer is stopped and continues after the timer becomes idle again.

CSMA/CA and Wireless Networks

CSMA/CA was mostly intended for use in wireless networks. The procedure described above, however, is not sophisticated enough to handle some particular issues related to wireless networks, such as hidden terminals or exposed terminals. We will see how these issues are solved by augmenting the above protocol with hand-shaking features. The use of CSMA/CA in wireless networks will be discussed in Chapter 14.

Figure 12.17 *Flow diagram for CSMA/CA*



12.2 CONTROLLED ACCESS

In **controlled access,** the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discuss three popular controlled-access methods.

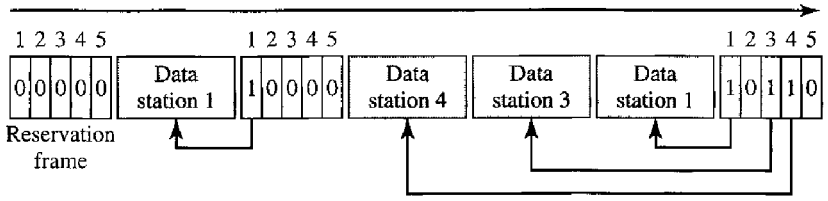
Reservation

In the **reservation** method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval.

If there are N stations in the system, there are exactly N reservation minislots in the reservation frame. Each minislot belongs to a station. When a station needs to send a data frame, it makes a reservation in its own minislot. The stations that have made reservations can send their data frames after the reservation frame.

Figure 12.18 shows a situation with five stations and a five-minislot reservation frame. In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation.

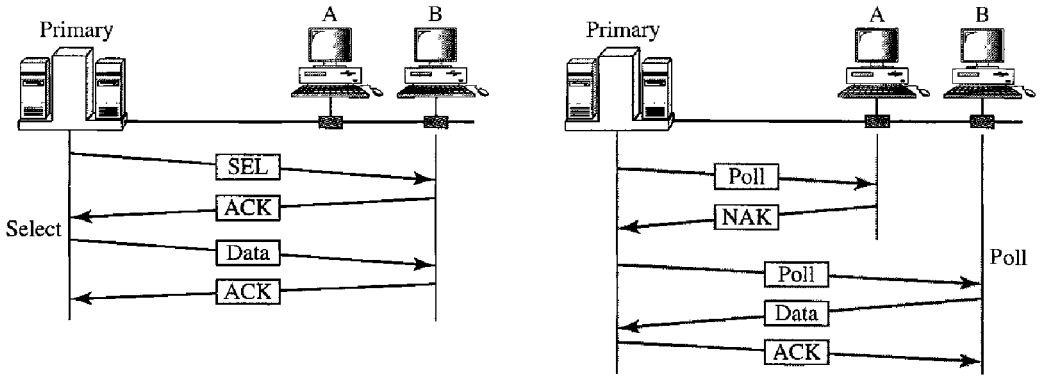
Figure 12.18 *Reservation access method*



Polling

Polling works with topologies in which one device is designated as a **primary station** and the other devices are **secondary stations.** All data exchanges must be made through the primary device even when the ultimate destination is a secondary device. The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time. The primary device, therefore, is always the initiator of a session (see Figure 12.19).

Figure 12.19 *Select and poll functions in polling access method*



If the primary wants to receive data, it asks the secondaries if they have anything to send; this is called poll function. If the primary wants to send data, it tells the secondary to get ready to receive; this is called select function.

Select

The *select* function is used whenever the primary device has something to send. Remember that the primary controls the link. If the primary is neither sending nor receiving data, it knows the link is available.

If it has something to send, the primary device sends it. What it does not know, however, is whether the target device is prepared to receive. So the primary must alert the secondary to the upcoming transmission and wait for an acknowledgment of the secondary's ready status. Before sending data, the primary creates and transmits a select (SEL) frame, one field of which includes the address of the intended secondary.

Poll

The *poll* function is used by the primary device to solicit transmissions from the secondary devices. When the primary is ready to receive data, it must ask (poll) each device in turn if it has anything to send. When the first secondary is approached, it responds either with a NAK frame if it has nothing to send or with data (in the form of a data frame) if it does. If the response is negative (a NAK frame), then the primary polls the next secondary in the same manner until it finds one with data to send. When the response is positive (a data frame), the primary reads the frame and returns an acknowledgment (ACK frame), verifying its receipt.

Token Passing

In the **token-passing** method, the stations in a network are organized in a logical ring. In other words, for each station, there is a *predecessor* and a *successor*. The predecessor is the station which is logically before the station in the ring; the successor is the station which is after the station in the ring. The current station is the one that is accessing the channel now. The right to this access has been passed from the predecessor to the current station. The right will be passed to the successor when the current station has no more data to send.

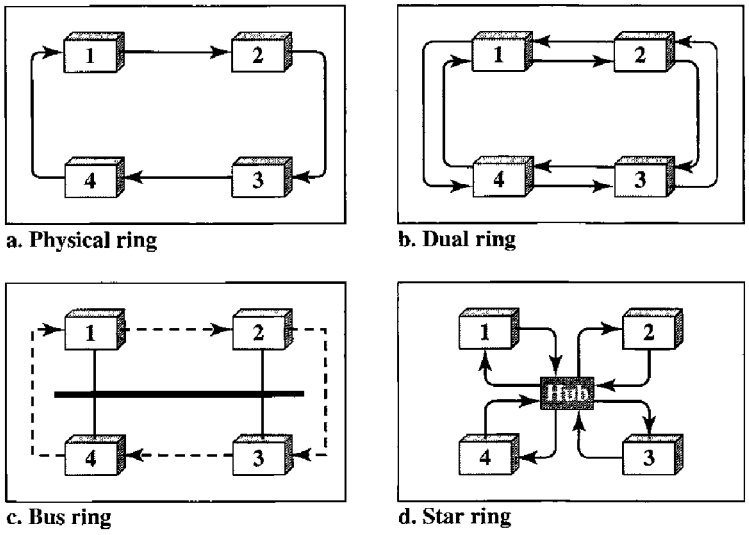
But how is the right to access the channel passed from one station to another? In this method, a special packet called a **token** circulates through the ring. The possession of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical station in the ring. The station cannot send data until it receives the token again in the next round. In this process, when a station receives the token and has no data to send, it just passes the data to the next station.

Token management is needed for this access method. Stations must be limited in the time they can have possession of the token. The token must be monitored to ensure it has not been lost or destroyed. For example, if a station that is holding the token fails, the token will disappear from the network. Another function of token management is to assign priorities to the stations and to the types of data being transmitted. And finally, token management is needed to make low-priority stations release the token to high-priority stations.

Logical Ring

In a token-passing network, stations do not have to be physically connected in a ring; the ring can be a logical one. Figure 12.20 show four different physical topologies that can create a logical ring.

Figure 12.20 *Logical ring and physical topology in token-passing access method*



In the physical ring topology, when a station sends the token to its successor, the token cannot be seen by other stations; the successor is the next one in line. This means that the token does not have to have the address of the next successor. The problem with this topology is that if one of the links—the medium between two adjacent stations—fails, the whole system fails.

The dual ring topology uses a second (auxiliary) ring which operates in the reverse direction compared with the main ring. The second ring is for emergencies only (such as a spare tire for a car). If one of the links in the main ring fails, the system automatically combines the two rings to form a temporary ring. After the failed link is restored, the auxiliary ring becomes idle again. Note that for this topology to work, each station needs to have two transmitter ports and two receiver ports. The high-speed Token Ring networks called FDDI (Fiber Distributed Data Interface) and CDDI (Copper Distributed Data Interface) use this topology.

In the bus ring topology, also called a token bus, the stations are connected to a single cable called a bus. They, however, make a logical ring, because each station knows the address of its successor (and also predecessor for token management purposes). When a station has finished sending its data, it releases the token and inserts the address of its successor in the token. Only the station with the address matching the destination address of the token gets the token to access the shared media. The Token Bus LAN, standardized by IEEE, uses this topology.

In a star ring topology, the physical topology is a star. There is a hub, however, that acts as the connector. The wiring inside the hub makes the ring; the stations are connected to this ring through the two wire connections. This topology makes the network

less prone to failure because if a link goes down, it will be bypassed by the hub and the rest of the stations can operate. Also adding and removing stations from the ring is easier. This topology is still used in the Token Ring LAN designed by IBM.

12.3 CHANNELIZATION

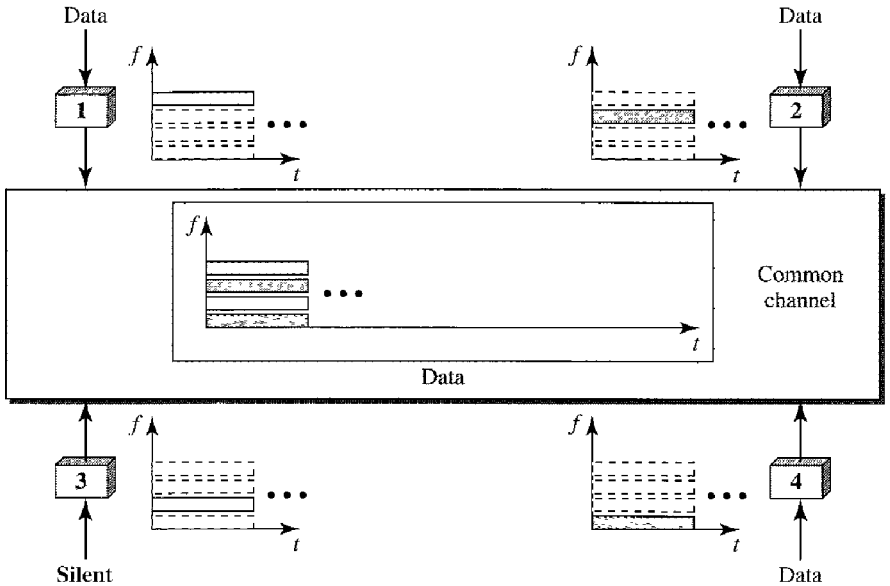
Channelization is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, between different stations. In this section, we discuss three channelization protocols: FDMA, TDMA, and CDMA.

We see the application of all these methods in Chapter 16 when we discuss cellular phone systems.

Frequency-Division Multiple Access (FDMA)

In **frequency-division multiple access (FDMA),** the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time. Each station also uses a bandpass filter to confine the transmitter frequencies. To prevent station interferences, the allocated bands are separated from one another by small *guard bands.* Figure 12.21 shows the idea of FDMA.

Figure 12.21 *Frequency-division multiple access (FDMA)*



In FDMA, the available bandwidth of the common channel is divided into bands that are separated by guard bands.

FDMA specifies a predetermined frequency band for the entire period of communication. This means that stream data (a continuous flow of data that may not be packetized) can easily be used with FDMA. We will see in Chapter 16 how this feature can be used in cellular telephone systems.

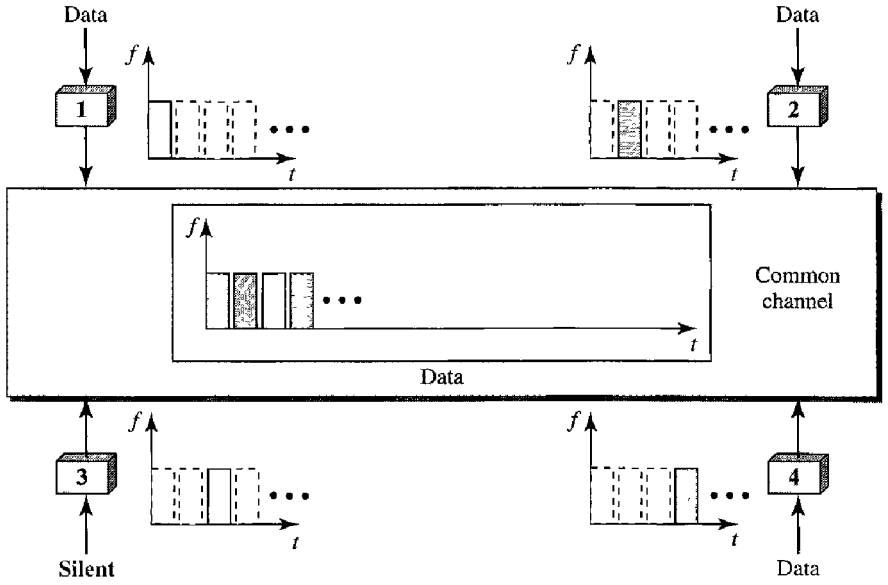
We need to emphasize that although FDMA and FDM conceptually seem similar, there are differences between them. FDM, as we saw in Chapter 6, is a physical layer technique that combines the loads from low-bandwidth channels and transmits them by using a high-bandwidth channel. The channels that are combined are low-pass. The multiplexer modulates the signals, combines them, and creates a bandpass signal. The bandwidth of each channel is shifted by the multiplexer.

FDMA, on the other hand, is an access method in the data link layer. The data link layer in each station tells its physical layer to make a bandpass signal from the data passed to it. The signal must be created in the allocated band. There is no physical multiplexer at the physical layer. The signals created at each station are automatically bandpass-filtered. They are mixed when they are sent to the common channel.

Time-Division Multiple Access (TDMA)

In **time-division multiple access (TDMA),** the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in is assigned time slot. Figure 12.22 shows the idea behind TDMA.

Figure 12.22 *Time-division multiple access (TDMA)*



The main problem with TDMA lies in achieving synchronization between the different stations. Each station needs to know the beginning of its slot and the location of its slot. This may be difficult because of propagation delays introduced in the system if the stations are spread over a large area. To compensate for the delays, we can insert *guard*

times. Synchronization is normally accomplished by having some synchronization bits (normally referred to as preamble bits) at the beginning of each slot.

In TDMA, the bandwidth is just one channel that is timeshared between different stations.

We also need to emphasize that although TDMA and TDM conceptually seem the same, there are differences between them. TDM, as we saw in Chapter 6, is a physical layer technique that combines the data from slower channels and transmits them by using a faster channel. The process uses a physical multiplexer that interleaves data units from each channel.

TDMA, on the other hand, is an access method in the data link layer. The data link layer in each station tells its physical layer to use the allocated time slot. There is no physical multiplexer at the physical layer.

Code-Division Multiple Access (CDMA)

Code-division multiple access (CDMA) was conceived several decades ago. Recent advances in electronic technology have finally made its implementation possible. CDMA differs from FDMA because only one channel occupies the entire bandwidth of the link. It differs from TDMA because all stations can send data simultaneously; there is no timesharing.

In CDMA, one channel carries all transmissions simultaneously.

Analogy

Let us first give an analogy. CDMA simply means communication with different codes. For example, in a large room with many people, two people can talk in English if nobody else understands English. Another two people can talk in Chinese if they are the only ones who understand Chinese, and so on. In other words, the common channel, the space of the room in this case, can easily allow communication between several couples, but in different languages (codes).

Idea

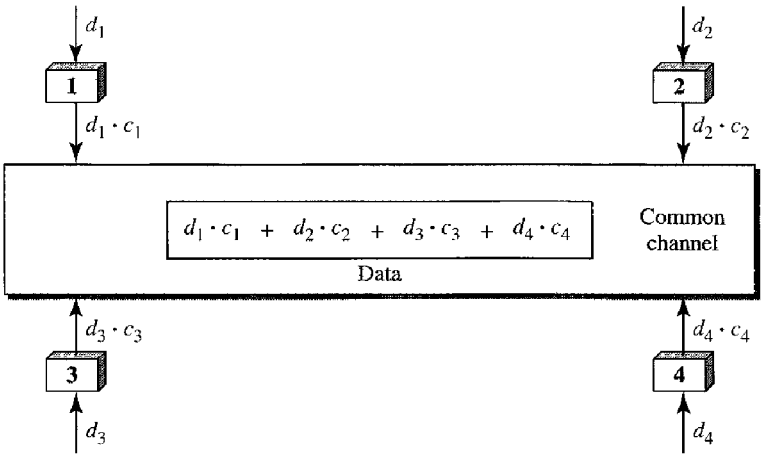
Let us assume we have four stations 1, 2, 3, and 4 connected to the same channel. The data from station 1 are d_1, from station 2 are d_2, and so on. The code assigned to the first station is c_1, to the second is c_2, and so on. We assume that the assigned codes have two properties.

1. If we multiply each code by another, we get 0.
2. If we multiply each code by itself, we get 4 (the number of stations).

With these two properties in mind, let us see how the above four stations can send data using the same common channel, as shown in Figure 12.23.

Station 1 multiplies (a special kind of multiplication, as we will see) its data by its code to get $d_1 \cdot c_1$. Station 2 multiplies its data by its code to get $d_2 \cdot c_2$. And so on. The

Figure 12.23 *Simple idea of communication with code*



data that go on the channel are the sum of all these terms, as shown in the box. Any station that wants to receive data from one of the other three multiplies the data on the channel by the code of the sender. For example, suppose stations 1 and 2 are talking to each other. Station 2 wants to hear what station 1 is saying. It multiplies the data on the channel by c_1, the code of station 1.

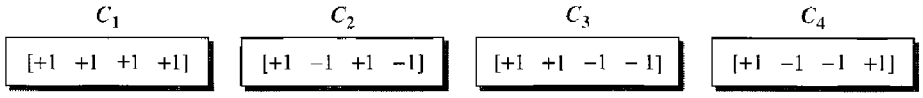
Because $(c_1 \cdot c_1)$ is 4, but $(c_2 \cdot c_1)$, $(c_3 \cdot c_1)$, and $(c_4 \cdot c_1)$ are all 0s, station 2 divides the result by 4 to get the data from station 1.

$$\begin{aligned}\text{data} &= (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1 \\ &= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 = 4 \times d_1\end{aligned}$$

Chips

CDMA is based on coding theory. Each station is assigned a code, which is a sequence of numbers called chips, as shown in Figure 12.24. The codes are for the previous example.

Figure 12.24 *Chip sequences*



Later in this chapter we show how we chose these sequences. For now, we need to know that we did not choose the sequences randomly; they were carefully selected. They are called **orthogonal sequences** and have the following properties:

1. Each sequence is made of N elements, where N is the number of stations.

2. If we multiply a sequence by a number, every element in the sequence is multiplied by that element. This is called multiplication of a sequence by a scalar. For example,

$$2 \cdot [+1 \ +1 \ -1 \ -1] = [+2 \ +2 \ -2 \ -2]$$

3. If we multiply two equal sequences, element by element, and add the results, we get N, where N is the number of elements in the each sequence. This is called the **inner product** of two equal sequences. For example,

$$[+1 \ +1 \ -1 \ -1] \cdot [+1 \ +1 \ -1 \ -1] = 1 + 1 + 1 + 1 = 4$$

4. If we multiply two different sequences, element by element, and add the results, we get 0. This is called inner product of two different sequences. For example,

$$[+1 \ +1 \ -1 \ -1] \cdot [+1 \ +1 \ +1 \ +1] = 1 + 1 - 1 - 1 = 0$$

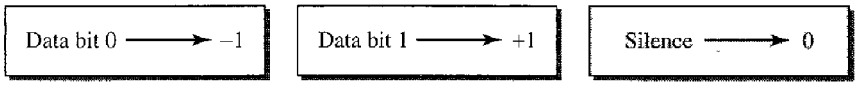
5. Adding two sequences means adding the corresponding elements. The result is another sequence. For example,

$$[+1 \ +1 \ -1 \ -1] + [+1 \ +1 \ +1 \ +1] = [+2 \ +2 \ \ 0 \ \ 0]$$

Data Representation

We follow these rules for encoding: If a station needs to send a 0 bit, it encodes it as −1; if it needs to send a 1 bit, it encodes it as +1. When a station is idle, it sends no signal, which is interpreted as a 0. These are shown in Figure 12.25.

Figure 12.25 *Data representation in CDMA*



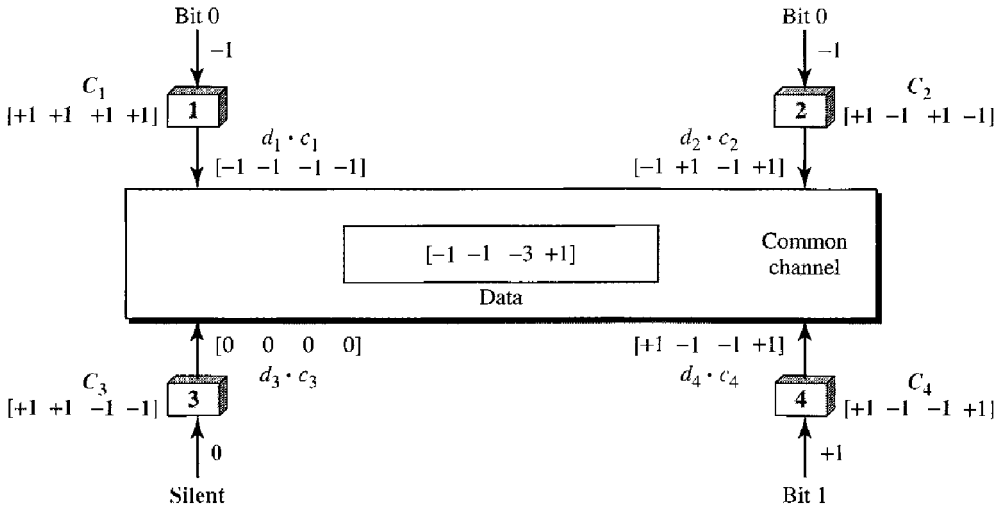
Encoding and Decoding

As a simple example, we show how four stations share the link during a 1-bit interval. The procedure can easily be repeated for additional intervals. We assume that stations 1 and 2 are sending a 0 bit and channel 4 is sending a 1 bit. Station 3 is silent. The data at the sender site are translated to −1, −1, 0, and +1. Each station multiplies the corresponding number by its chip (its orthogonal sequence), which is unique for each station. The result is a new sequence which is sent to the channel. For simplicity, we assume that all stations send the resulting sequences at the same time. The sequence on the channel is the sum of all four sequences as defined before. Figure 12.26 shows the situation.

Now imagine station 3, which we said is silent, is listening to station 2. Station 3 multiplies the total data on the channel by the code for station 2, which is [+1 −1 +1 −1], to get

$$[-1 \ -1 \ -3 \ +1] \cdot [+1 \ -1 \ +1 \ -1] = -4/4 = -1 \Rightarrow \text{bit 1}$$

Figure 12.26 *Sharing channel in CDMA*



Signal Level

The process can be better understood if we show the digital signal produced by each station and the data recovered at the destination (see Figure 12.27). The figure shows the corresponding signals for each station (using NRZ-L for simplicity) and the signal that is on the common channel.

Figure 12.27 *Digital signal created by four stations in CDMA*

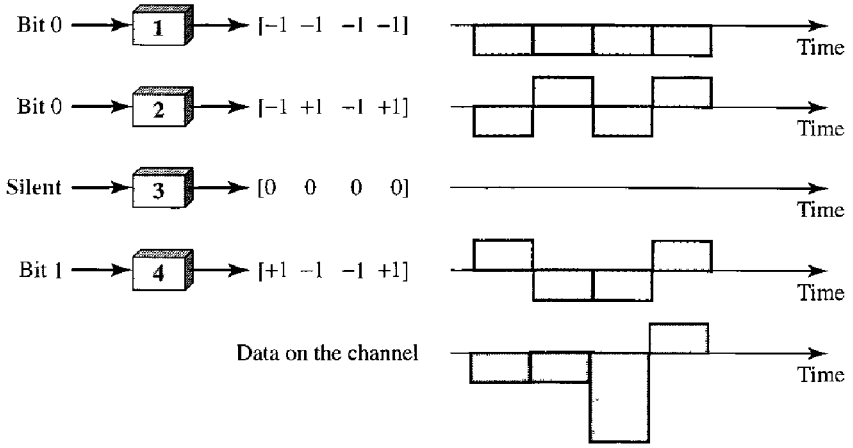
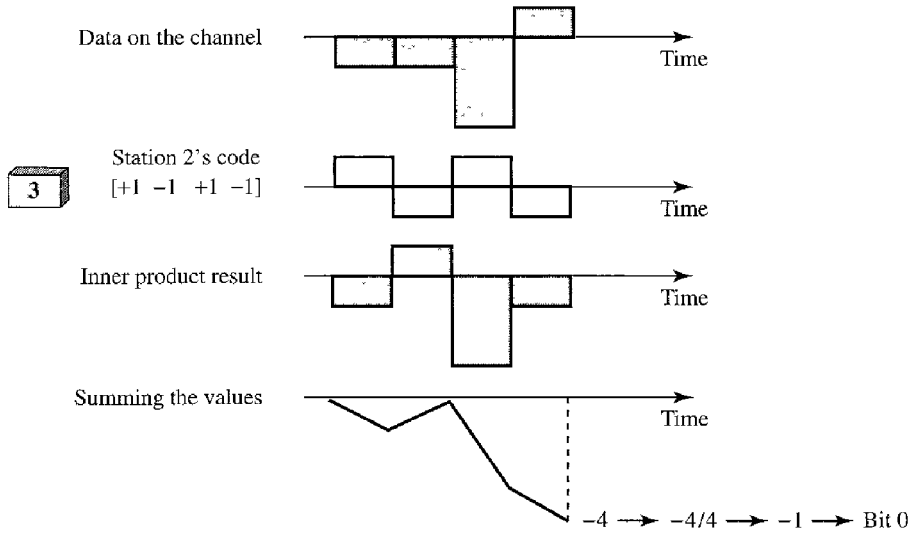


Figure 12.28 shows how station 3 can detect the data sent by station 2 by using the code for station 2. The total data on the channel are multiplied (inner product operation) by the signal representing station 2 chip code to get a new signal. The station then integrates and adds the area under the signal, to get the value −4, which is divided by 4 and interpreted as bit 0.

Figure 12.28 *Decoding of the composite signal for one in CDMA*



Sequence Generation

To generate chip sequences, we use a **Walsh table,** which is a two-dimensional table with an equal number of rows and columns, as shown in Figure 12.29.

Figure 12.29 *General rule and examples of creating Walsh tables*

$$W_1 = \begin{bmatrix} +1 \end{bmatrix} \qquad W_{2N} = \begin{bmatrix} W_N & W_N \\ W_N & \overline{W_N} \end{bmatrix}$$

a. Two basic rules

$$W_1 = \begin{bmatrix} +1 \end{bmatrix} \qquad W_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \qquad W_4 = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}$$

b. Generation of W_1, W_2, and W_4

In the Walsh table, each row is a sequence of chips. W_1 for a one-chip sequence has one row and one column. We can choose −1 or +1 for the chip for this trivial table (we chose +1). According to Walsh, if we know the table for N sequences W_N, we can create the table for $2N$ sequences W_{2N}, as shown in Figure 12.29. The W_N with the overbar $\overline{W_N}$ stands for the complement of W_N, where each +1 is changed to −1 and vice versa. Figure 12.29 also shows how we can create W_2 and W_4 from W_1. After we select W_1, W_2

can be made from four W_1's, with the last one the complement of W_1. After W_2 is generated, W_4 can be made of four W_2's, with the last one the complement of W_2. Of course, W_8 is composed of four W_4's, and so on. Note that after W_N is made, each station is assigned a chip corresponding to a row.

Something we need to emphasize is that the number of sequences N needs to be a power of 2. In other words, we need to have $N = 2^m$.

The number of sequences in a Walsh table needs to be $N = 2^m$.

Example 12.6

Find the chips for a network with

a. Two stations

b. Four stations

Solution

We can use the rows of W_2 and W_4 in Figure 12.29:

a. For a two-station network, we have [+1 +1] and [+1 −1].

b. For a four-station network we have [+1 +1 +1 +1], [+1 −1 +1 −1], [+1 +1 −1 −1], and [+1 −1 −1 +1].

Example 12.7

What is the number of sequences if we have 90 stations in our network?

Solution

The number of sequences needs to be 2^m. We need to choose $m = 7$ and $N = 2^7$ or 128. We can then use 90 of the sequences as the chips.

Example 12.8

Prove that a receiving station can get the data sent by a specific sender if it multiplies the entire data on the channel by the sender's chip code and then divides it by the number of stations.

Solution

Let us prove this for the first station, using our previous four-station example. We can say that the data on the channel $D = (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4)$. The receiver which wants to get the data sent by station 1 multiplies these data by c_1.

$$
\begin{aligned}
D \cdot c_1 &= (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1 \\
&= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 \\
&= d_1 \times N + d_2 \times 0 + d_3 \times 0 + d_4 \times 0 \\
&= d_1 \times N
\end{aligned}
$$

When we divide the result by N, we get d_1.

12.4 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

Multiple access is discussed in Chapter 4 of [Tan03], Chapter 16 of [Sta04], Chapter 6 of [GW04], and Chapter 8 of [For03]. More advanced materials can be found in [KMK04].

12.5 KEY TERMS

1-persistent method
ALOHA
binary exponential backup
carrier sense multiple access (CSMA)
carrier sense multiple access with collision avoidance (CSMA/CA)
carrier sense multiple access with collision detection (CSMA/CD)
channelization
code-division multiple access (CDMA)
collision
contention
controlled access
frequency-division multiple access (FDMA)
inner product
interframe space (IFS)
jamming signal
multiple access (MA)
nonpersistent method
orthogonal sequence
polling
p-persistent method
primary station
propagation time
pure ALOHA
random access
reservation
secondary station
slotted ALOHA
time-division multiple access (TDMA)
token
token passing
vulnerable time
Walsh table

12.6 SUMMARY

- We can consider the data link layer as two sublayers. The upper sublayer is responsible for data link control, and the lower sublayer is responsible for resolving access to the shared media.
- Many formal protocols have been devised to handle access to a shared link. We categorize them into three groups: random access protocols, controlled access protocols, and channelization protocols.
- In random access or contention methods, no station is superior to another station and none is assigned the control over another.
- ALOHA allows multiple access (MA) to the shared medium.There are potential collisions in this arrangement. When a station sends data, another station may attempt to do so at the same time. The data from the two stations collide and become garbled.
- To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station

senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium before sending. Three methods have been devised for carrier sensing: 1-persistent, nonpersistent, and *p*-persistent.

- Carrier sense multiple access with collision detection (CSMA/CD) augments the CSMA algorithm to handle collision. In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.
- To avoid collisions on wireless networks, carrier sense multiple access with collision avoidance (CSMA/CA) was invented. Collisions are avoided through the use three strategies: the interframe space, the contention window, and acknowledgments.
- In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discussed three popular controlled-access methods: reservation, polling, and token passing.
- In the reservation access method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval.
- In the polling method, all data exchanges must be made through the primary device even when the ultimate destination is a secondary device. The primary device controls the link; the secondary devices follow its instructions.
- In the token-passing method, the stations in a network are organized in a logical ring. Each station has a predecessor and a successor. A special packet called a token circulates through the ring.
- Channelization is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, between different stations. We discussed three channelization protocols: FDMA, TDMA, and CDMA.
- In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time.
- In time-division multiple access (TDMA), the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot.
- In code-division multiple access (CDMA), the stations use different codes to achieve multiple access. CDMA is based on coding theory and uses sequences of numbers called chips. The sequences are generated using orthogonal codes such the Walsh tables.

12.7 PRACTICE SET

Review Questions

1. List three categories of multiple access protocols discussed in this chapter.
2. Define random access and list three protocols in this category.

3. Define controlled access and list three protocols in this category.
4. Define channelization and list three protocols in this category.
5. Explain why collision is an issue in a random access protocol but not in controlled access or channelizing protocols.
6. Compare and contrast a random access protocol with a controlled access protocol.
7. Compare and contrast a random access protocol with a channelizing protocol.
8. Compare and contrast a controlled access protocol with a channelizing protocol.
9. Do we need a multiple access protocol when we use the local loop of the telephone company to access the Internet? Why?
10. Do we need a multiple access protocol when we use one CATV channel to access the Internet? Why?

Exercises

11. We have a pure ALOHA network with 100 stations. If $T_{fr} = 1$ μs, what is the number of frames/s each station can send to achieve the maximum efficiency.
12. Repeat Exercise 11 for slotted ALOHA.
13. One hundred stations on a pure ALOHA network share a 1-Mbps channel. If frames are 1000 bits long, find the throughput if each station is sending 10 frames per second.
14. Repeat Exercise 13 for slotted ALOHA.
15. In a CDMA/CD network with a data rate of 10 Mbps, the minimum frame size is found to be 512 bits for the correct operation of the collision detection process. What should be the minimum frame size if we increase the data rate to 100 Mbps? To 1 Gbps? To 10 Gbps?
16. In a CDMA/CD network with a data rate of 10 Mbps, the maximum distance between any station pair is found to be 2500 m for the correct operation of the collision detection process. What should be the maximum distance if we increase the data rate to 100 Mbps? To 1 Gbps? To 10 Gbps?
17. In Figure 12.12, the data rate is 10 Mbps, the distance between station A and C is 2000 m, and the propagation speed is 2×10^8 m/s. Station A starts sending a long frame at time $t_1 = 0$; station C starts sending a long frame at time $t_2 = 3$ μs. The size of the frame is long enough to guarantee the detection of collision by both stations. Find:
    a. The time when station C hears the collision (t_3).
    b. The time when station A hears the collision (t_4).
    c. The number of bits station A has sent before detecting the collision.
    d. The number of bits station C has sent before detecting the collision.
18. Repeat Exercise 17 if the data rate is 100 Mbps.
19. Calculate the Walsh table W_8 from W_4 in Figure 12.29.
20. Recreate the W_2 and W_4 tables in Figure 12.29 using $W_1 = [-1]$. Compare the recreated tables with the ones in Figure 12.29.
21. Prove the third and fourth orthogonal properties of Walsh chips for W_4 in Figure 12.29.

22. Prove the third and fourth orthogonal properties of Walsh chips for W_4 recreated in Exercise 20.
23. Repeat the scenario depicted in Figures 12.27 to 12.28 if both stations 1 and 3 are silent.
24. A network with one primary and four secondary stations uses polling. The size of a data frame is 1000 bytes. The size of the poll, ACK, and NAK frames are 32 bytes each. Each station has 5 frames to send. How many total bytes are exchanged if there is no limitation on the number of frames a station can send in response to a poll?
25. Repeat Exercise 24 if each station can send only one frame in response to a poll.

Research Activities

26. Can you explain why the vulnerable time in ALOHA depends on T_{Fr}, but in CSMA depends on T_p?
27. In analyzing ALOHA, we use only one parameter, time; in analyzing CSMA, we use two parameters, space and time. Can you explain the reason?

CHAPTER 13

Wired LANs: Ethernet

In Chapter 1, we learned that a local area network (LAN) is a computer network that is designed for a limited geographic area such as a building or a campus. Although a LAN can be used as an isolated network to connect computers in an organization for the sole purpose of sharing resources, most LANs today are also linked to a wide area network (WAN) or the Internet.

The LAN market has seen several technologies such as Ethernet, Token Ring, Token Bus, FDDI, and ATM LAN. Some of these technologies survived for a while, but Ethernet is by far the dominant technology.

In this chapter, we first briefly discuss the IEEE Standard Project 802, designed to regulate the manufacturing and interconnectivity between different LANs. We then concentrate on the Ethernet LANs.

Although Ethernet has gone through a four-generation evolution during the last few decades, the main concept has remained. Ethernet has changed to meet the market needs and to make use of the new technologies.

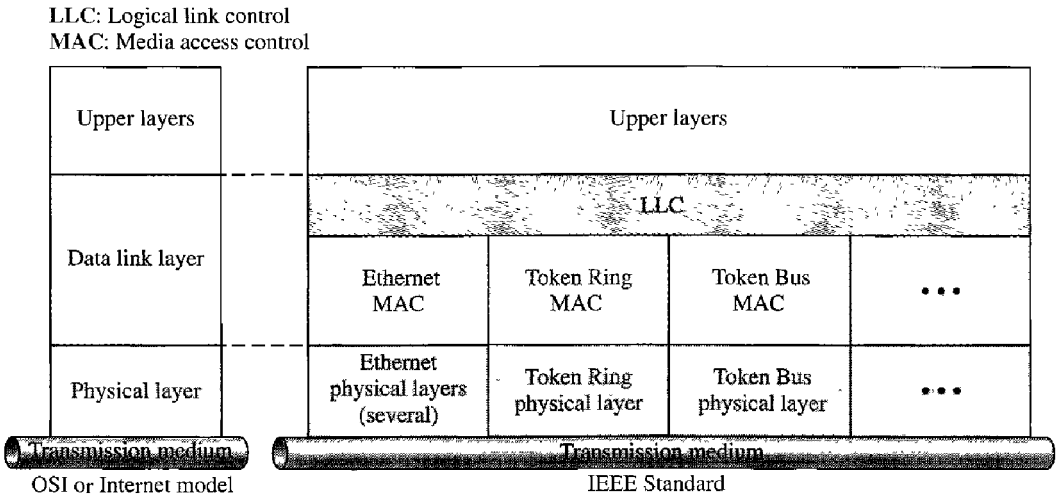
13.1 IEEE STANDARDS

In 1985, the Computer Society of the IEEE started a project, called **Project 802,** to set standards to enable intercommunication among equipment from a variety of manufacturers. Project 802 does not seek to replace any part of the OSI or the Internet model. Instead, it is a way of specifying functions of the physical layer and the data link layer of major LAN protocols.

The standard was adopted by the American National Standards Institute (ANSI). In 1987, the International Organization for Standardization (ISO) also approved it as an international standard under the designation ISO 8802.

The relationship of the 802 Standard to the traditional OSI model is shown in Figure 13.1. The IEEE has subdivided the data link layer into two sublayers: **logical link control (LLC)** and **media access control (MAC).** IEEE has also created several physical layer standards for different LAN protocols.

Figure 13.1 *IEEE standard for LANs*



Data Link Layer

As we mentioned before, the data link layer in the IEEE standard is divided into two sublayers: LLC and MAC.

Logical Link Control (LLC)

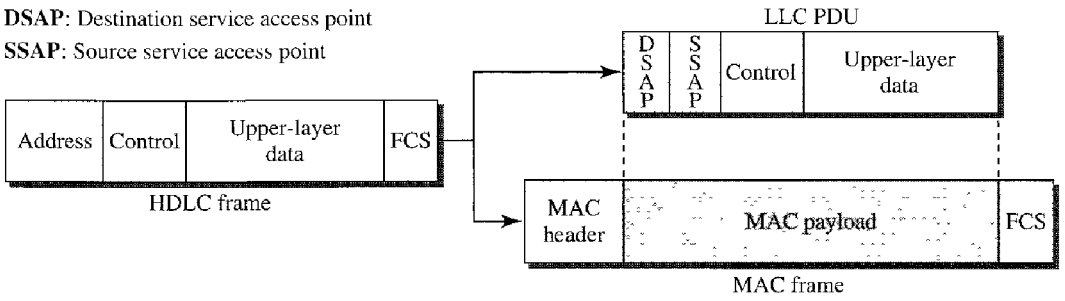
In Chapter 11, we discussed data link control. We said that data link control handles framing, flow control, and error control. In IEEE Project 802, flow control, error control, and part of the framing duties are collected into one sublayer called the logical link control. Framing is handled in both the LLC sublayer and the MAC sublayer.

The LLC provides one single data link control protocol for all IEEE LANs. In this way, the LLC is different from the media access control sublayer, which provides different protocols for different LANs. A single LLC protocol can provide interconnectivity between different LANs because it makes the MAC sublayer transparent. Figure 13.1 shows one single LLC protocol serving several MAC protocols.

Framing LLC defines a protocol data unit (PDU) that is somewhat similar to that of HDLC. The header contains a control field like the one in HDLC; this field is used for flow and error control. The two other header fields define the upper-layer protocol at the source and destination that uses LLC. These fields are called the **destination service access point (DSAP)** and the **source service access point (SSAP).** The other fields defined in a typical data link control protocol such as HDLC are moved to the MAC sublayer. In other words, a frame defined in HDLC is divided into a PDU at the LLC sublayer and a frame at the MAC sublayer, as shown in Figure 13.2.

Need for LLC The purpose of the LLC is to provide flow and error control for the upper-layer protocols that actually demand these services. For example, if a LAN or several LANs are used in an isolated system, LLC may be needed to provide flow and error control for the application layer protocols. However, most upper-layer protocols

Figure 13.2 *HDLC frame compared with LLC and MAC frames*



such as IP (discussed in Chapter 20), do not use the services of LLC. For this reason, we end our discussion of LLC.

Media Access Control (MAC)

In Chapter 12, we discussed multiple access methods including random access, controlled access, and channelization. IEEE Project 802 has created a sublayer called media access control that defines the specific access method for each LAN. For example, it defines CSMA/CD as the media access method for Ethernet LANs and the token-passing method for Token Ring and Token Bus LANs. As we discussed in the previous section, part of the framing function is also handled by the MAC layer.

In contrast to the LLC sublayer, the MAC sublayer contains a number of distinct modules; each defines the access method and the framing format specific to the corresponding LAN protocol.

Physical Layer

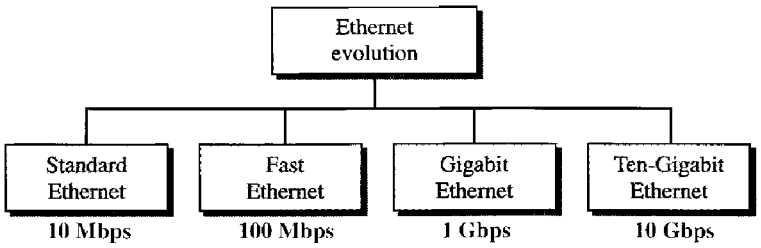
The physical layer is dependent on the implementation and type of physical media used. IEEE defines detailed specifications for each LAN implementation. For example, although there is only one MAC sublayer for Standard Ethernet, there is a different physical layer specifications for each Ethernet implementations as we will see later.

13.2 STANDARD ETHERNET

The original Ethernet was created in 1976 at Xerox's Palo Alto Research Center (PARC). Since then, it has gone through four generations: **Standard Ethernet** (10[†] Mbps), **Fast Ethernet** (100 Mbps), **Gigabit Ethernet** (1 Gbps), and **Ten-Gigabit Ethernet** (10 Gbps), as shown in Figure 13.3. We briefly discuss all these generations starting with the first, Standard (or traditional) Ethernet.

[†] Ethernet defined some 1-Mbps protocols, but they did not survive.

Figure 13.3 *Ethernet evolution through four generations*



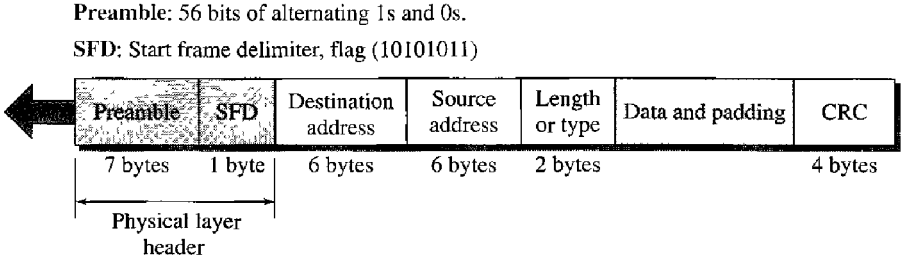
MAC Sublayer

In Standard Ethernet, the MAC sublayer governs the operation of the access method. It also frames data received from the upper layer and passes them to the physical layer.

Frame Format

The Ethernet frame contains seven fields: preamble, SFD, DA, SA, length or type of protocol data unit (PDU), upper-layer data, and the CRC. Ethernet does not provide any mechanism for acknowledging received frames, making it what is known as an unreliable medium. Acknowledgments must be implemented at the higher layers. The format of the MAC frame is shown in Figure 13.4.

Figure 13.4 *802.3 MAC frame*



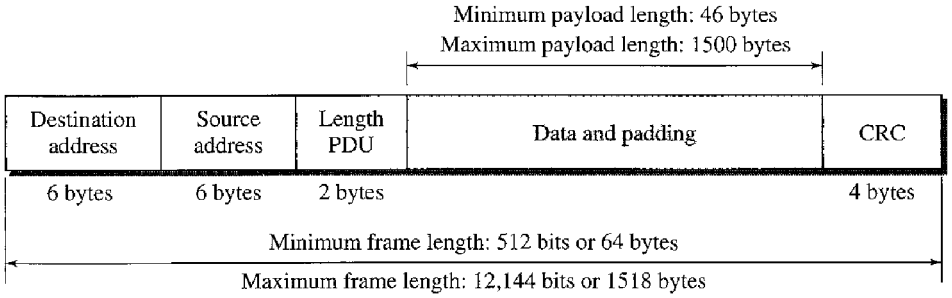
- **Preamble.** The first field of the 802.3 frame contains 7 bytes (56 bits) of alternating 0s and 1s that alerts the receiving system to the coming frame and enables it to synchronize its input timing. The pattern provides only an alert and a timing pulse. The 56-bit pattern allows the stations to miss some bits at the beginning of the frame. The **preamble** is actually added at the physical layer and is not (formally) part of the frame.
- **Start frame delimiter (SFD).** The second field (1 byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits is 11 and alerts the receiver that the next field is the destination address.

- **Destination address (DA).** The DA field is 6 bytes and contains the physical address of the destination station or stations to receive the packet. We will discuss addressing shortly.
- **Source address (SA).** The SA field is also 6 bytes and contains the physical address of the sender of the packet. We will discuss addressing shortly.
- **Length or type.** This field is defined as a type field or length field. The original Ethernet used this field as the type field to define the upper-layer protocol using the MAC frame. The IEEE standard used it as the length field to define the number of bytes in the data field. Both uses are common today.
- **Data.** This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500 bytes, as we will see later.
- **CRC.** The last field contains error detection information, in this case a CRC-32 (see Chapter 10).

Frame Length

Ethernet has imposed restrictions on both the minimum and maximum lengths of a frame, as shown in Figure 13.5.

Figure 13.5 *Minimum and maximum lengths*



The minimum length restriction is required for the correct operation of CSMA/CD as we will see shortly. An Ethernet frame needs to have a minimum length of 512 bits or 64 bytes. Part of this length is the header and the trailer. If we count 18 bytes of header and trailer (6 bytes of source address, 6 bytes of destination address, 2 bytes of length or type, and 4 bytes of CRC), then the minimum length of data from the upper layer is 64 − 18 = 46 bytes. If the upper-layer packet is less than 46 bytes, padding is added to make up the difference.

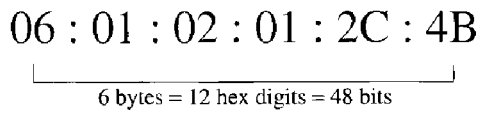
The standard defines the maximum length of a frame (without preamble and SFD field) as 1518 bytes. If we subtract the 18 bytes of header and trailer, the maximum length of the payload is 1500 bytes. The maximum length restriction has two historical reasons. First, memory was very expensive when Ethernet was designed: a maximum length restriction helped to reduce the size of the buffer. Second, the maximum length restriction prevents one station from monopolizing the shared medium, blocking other stations that have data to send.

Frame length:
Minimum: 64 bytes (512 bits) Maximum: 1518 bytes (12,144 bits)

Addressing

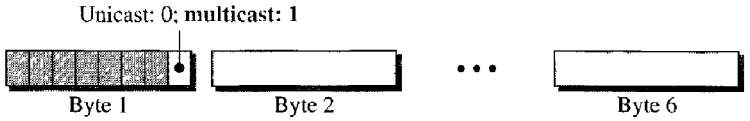
Each station on an Ethernet network (such as a PC, workstation, or printer) has its own **network interface card (NIC).** The NIC fits inside the station and provides the station with a 6-byte physical address. As shown in Figure 13.6, the Ethernet address is 6 bytes (48 bits), normally written in **hexadecimal notation,** with a colon between the bytes.

Figure 13.6 *Example of an Ethernet address in hexadecimal notation*



Unicast, Multicast, and Broadcast Addresses A source address is always a unicast address—the frame comes from only one station. The destination address, however, can be unicast, multicast, or broadcast. Figure 13.7 shows how to distinguish a unicast address from a multicast address. If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast.

Figure 13.7 *Unicast and multicast addresses*



The least significant bit of the first byte defines the type of address. If the bit is 0, the address is unicast; otherwise, it is multicast.

A unicast destination address defines only one recipient; the relationship between the sender and the receiver is one-to-one. A multicast destination address defines a group of addresses; the relationship between the sender and the receivers is one-to-many.

The broadcast address is a special case of the multicast address; the recipients are all the stations on the LAN. A broadcast destination address is forty-eight 1s.

The broadcast destination address is a special case of the multicast address in which all bits are 1s.

Example 13.1

Define the type of the following destination addresses:

a. 4A:30:10:21:10:1A
b. 47:20:1B:2E:08:EE
c. FF:FF:FF:FF:FF:FF

Solution

To find the type of the address, we need to look at the second hexadecimal digit from the left. If it is even, the address is unicast. If it is odd, the address is multicast. If all digits are F's, the address is broadcast. Therefore, we have the following:

a. This is a unicast address because A in binary is 1010 (even).
b. This is a multicast address because 7 in binary is 0111 (odd).
c. This is a broadcast address because all digits are F's.

The way the addresses are sent out on line is different from the way they are written in hexadecimal notation. The transmission is left-to-right, byte by byte; however, for each byte, the least significant bit is sent first and the most significant bit is sent last. This means that the bit that defines an address as unicast or multicast arrives first at the receiver.

Example 13.2

Show how the address 47:20:1B:2E:08:EE is sent out on line.

Solution

The address is sent left-to-right, byte by byte; for each byte, it is sent right-to-left, bit by bit, as shown below:

$$\leftarrow \quad 11100010\ 00000100\ 11011000\ 01110100\ 00010000\ 01110111$$

Access Method: CSMA/CD

Standard Ethernet uses 1-persistent CSMA/CD (see Chapter 12).

Slot Time In an Ethernet network, the round-trip time required for a frame to travel from one end of a maximum-length network to the other plus the time needed to send the jam sequence is called the slot time.

$$\text{Slot time} = \text{round-trip time} + \text{time required to send the jam sequence}$$

The slot time in Ethernet is defined in bits. It is the time required for a station to send 512 bits. This means that the actual slot time depends on the data rate; for traditional 10-Mbps Ethernet it is 51.2 μs.

Slot Time and Collision The choice of a 512-bit slot time was not accidental. It was chosen to allow the proper functioning of CSMA/CD. To understand the situation, let us consider two cases.

In the first case, we assume that the sender sends a minimum-size packet of 512 bits. Before the sender can send the entire packet out, the signal travels through the network

and reaches the end of the network. If there is another signal at the end of the network (worst case), a collision occurs. The sender has the opportunity to abort the sending of the frame and to send a jam sequence to inform other stations of the collision. The round-trip time plus the time required to send the jam sequence should be less than the time needed for the sender to send the minimum frame, 512 bits. The sender needs to be aware of the collision before it is too late, that is, before it has sent the entire frame.

In the second case, the sender sends a frame larger than the minimum size (between 512 and 1518 bits). In this case, if the station has sent out the first 512 bits and has not heard a collision, it is guaranteed that collision will never occur during the transmission of this frame. The reason is that the signal will reach the end of the network in less than one-half the slot time. If all stations follow the CSMA/CD protocol, they have already sensed the existence of the signal (carrier) on the line and have refrained from sending. If they sent a signal on the line before one-half of the slot time expired, a collision has occurred and the sender has sensed the collision. In other words, collision can only occur during the first half of the slot time, and if it does, it can be sensed by the sender during the slot time. This means that after the sender sends the first 512 bits, it is guaranteed that collision will not occur during the transmission of this frame. The medium belongs to the sender, and no other station will use it. In other words, the sender needs to listen for a collision only during the time the first 512 bits are sent.

Of course, all these assumptions are invalid if a station does not follow the CSMA/CD protocol. In this case, we do not have a collision, we have a corrupted station.

Slot Time and Maximum Network Length There is a relationship between the slot time and the maximum length of the network (collision domain). It is dependent on the propagation speed of the signal in the particular medium. In most transmission media, the signal propagates at 2×10^8 m/s (two-thirds of the rate for propagation in air). For traditional Ethernet, we calculate

$$\text{MaxLength} = \text{PropagationSpeed} \times \frac{\text{SlotTime}}{2}$$

$$\text{MaxLength} = (2 \times 10^8) \times (51.2 \times 10^{-6} / 2) = 5120 \text{ m}$$

Of course, we need to consider the delay times in repeaters and interfaces, and the time required to send the jam sequence. These reduce the maximum-length of a traditional Ethernet network to 2500 m, just 48 percent of the theoretical calculation.

$$\text{MaxLength} = 2500 \text{ m}$$

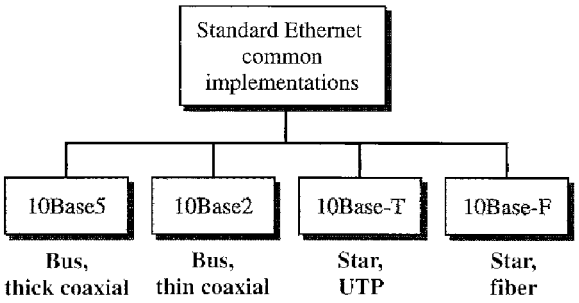
Physical Layer

The Standard Ethernet defines several physical layer implementations; four of the most common, are shown in Figure 13.8.

Encoding and Decoding

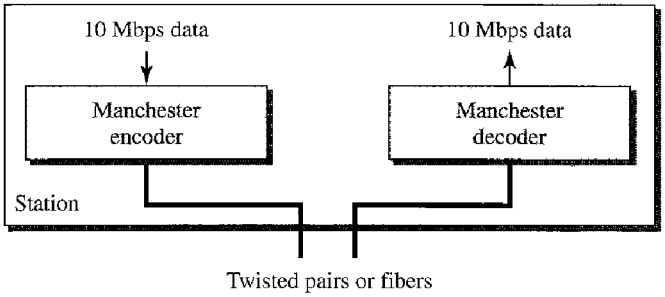
All standard implementations use digital signaling (baseband) at 10 Mbps. At the sender, data are converted to a digital signal using the Manchester scheme; at the receiver, the

Figure 13.8 *Categories of Standard Ethernet*



received signal is interpreted as Manchester and decoded into data. As we saw in Chapter 4, Manchester encoding is self-synchronous, providing a transition at each bit interval. Figure 13.9 shows the encoding scheme for Standard Ethernet.

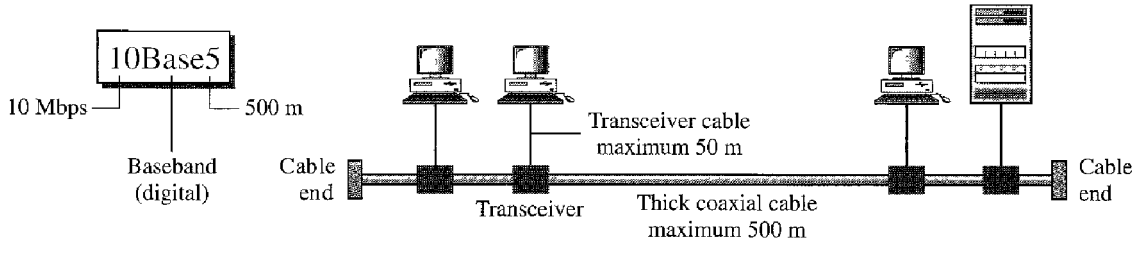
Figure 13.9 *Encoding in a Standard Ethernet implementation*



10Base5: Thick Ethernet

The first implementation is called **10Base5, thick Ethernet,** or **Thicknet.** The nickname derives from the size of the cable, which is roughly the size of a garden hose and too stiff to bend with your hands. 10Base5 was the first Ethernet specification to use a bus topology with an external **transceiver** (transmitter/receiver) connected via a tap to a thick coaxial cable. Figure 13.10 shows a schematic diagram of a 10Base5 implementation.

Figure 13.10 *10Base5 implementation*



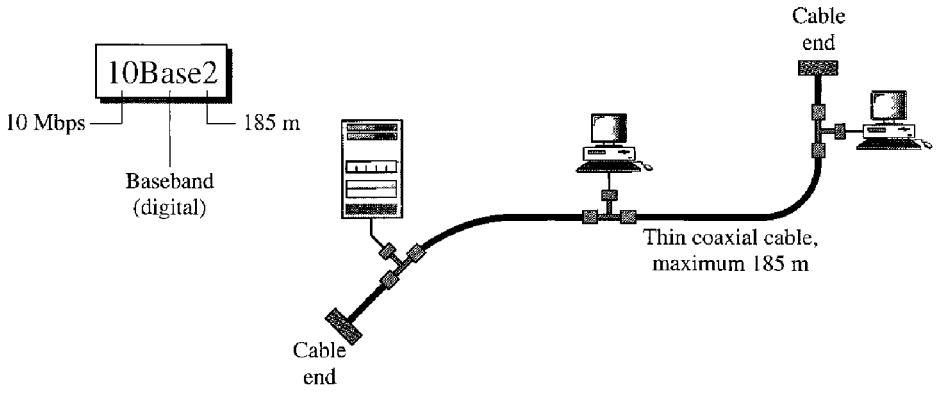
The transceiver is responsible for transmitting, receiving, and detecting collisions. The **transceiver** is connected to the station via a transceiver cable that provides separate paths for sending and receiving. This means that collision can only happen in the coaxial cable.

The maximum length of the coaxial cable must not exceed 500 m, otherwise, there is excessive degradation of the signal. If a length of more than 500 m is needed, up to five segments, each a maximum of 500-meter, can be connected using repeaters. Repeaters will be discussed in Chapter 15.

10Base2: Thin Ethernet

The second implementation is called **10Base2, thin Ethernet,** or **Cheapernet.** 10Base2 also uses a bus topology, but the cable is much thinner and more flexible. The cable can be bent to pass very close to the stations. In this case, the transceiver is normally part of the network interface card (NIC), which is installed inside the station. Figure 13.11 shows the schematic diagram of a 10Base2 implementation.

Figure 13.11 *10Base2 implementation*



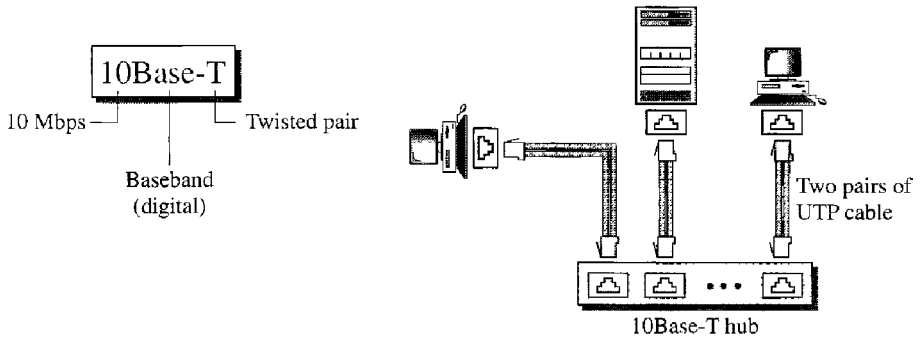
Note that the collision here occurs in the thin coaxial cable. This implementation is more cost effective than 10Base5 because thin coaxial cable is less expensive than thick coaxial and the tee connections are much cheaper than taps. Installation is simpler because the thin coaxial cable is very flexible. However, the length of each segment cannot exceed 185 m (close to 200 m) due to the high level of attenuation in thin coaxial cable.

10Base-T: Twisted-Pair Ethernet

The third implementation is called **10Base-T** or **twisted-pair Ethernet.** 10Base-T uses a physical star topology. The stations are connected to a hub via two pairs of twisted cable, as shown in Figure 13.12.

Note that two pairs of twisted cable create two paths (one for sending and one for receiving) between the station and the hub. Any collision here happens in the hub. Compared to 10Base5 or 10Base2, we can see that the hub actually replaces the coaxial

Figure 13.12 *10Base-T implementation*

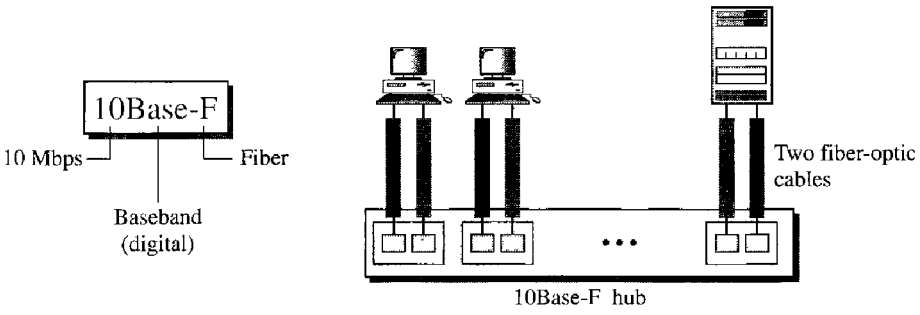


cable as far as a collision is concerned. The maximum length of the twisted cable here is defined as 100 m, to minimize the effect of attenuation in the twisted cable.

10Base-F: Fiber Ethernet

Although there are several types of optical fiber 10-Mbps Ethernet, the most common is called **10Base-F.** 10Base-F uses a star topology to connect stations to a hub. The stations are connected to the hub using two fiber-optic cables, as shown in Figure 13.13.

Figure 13.13 *10Base-F implementation*



Summary

Table 13.1 shows a summary of Standard Ethernet implementations.

Table 13.1 *Summary of Standard Ethernet implementations*

Characteristics	*10Base5*	*10Base2*	*10Base-T*	*10Base-F*
Media	Thick coaxial cable	Thin coaxial cable	2 UTP	2 Fiber
Maximum length	500 m	185 m	100 m	2000 m
Line encoding	Manchester	Manchester	Manchester	Manchester

13.3 CHANGES IN THE STANDARD

The 10-Mbps Standard Ethernet has gone through several changes before moving to the higher data rates. These changes actually opened the road to the evolution of the Ethernet to become compatible with other high-data-rate LANs. We discuss some of these changes in this section.

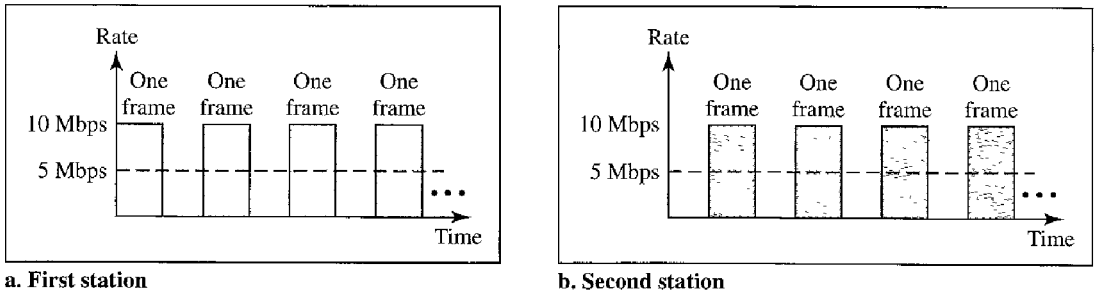
Bridged Ethernet

The first step in the Ethernet evolution was the division of a LAN by **bridges.** Bridges have two effects on an Ethernet LAN: They raise the bandwidth and they separate collision domains. We discuss bridges in Chapter 15.

Raising the Bandwidth

In an unbridged Ethernet network, the total capacity (10 Mbps) is shared among all stations with a frame to send; the stations share the bandwidth of the network. If only one station has frames to send, it benefits from the total capacity (10 Mbps). But if more than one station needs to use the network, the capacity is shared. For example, if two stations have a lot of frames to send, they probably alternate in usage. When one station is sending, the other one refrains from sending. We can say that, in this case, each station on average, sends at a rate of 5 Mbps. Figure 13.14 shows the situation.

Figure 13.14 *Sharing bandwidth*

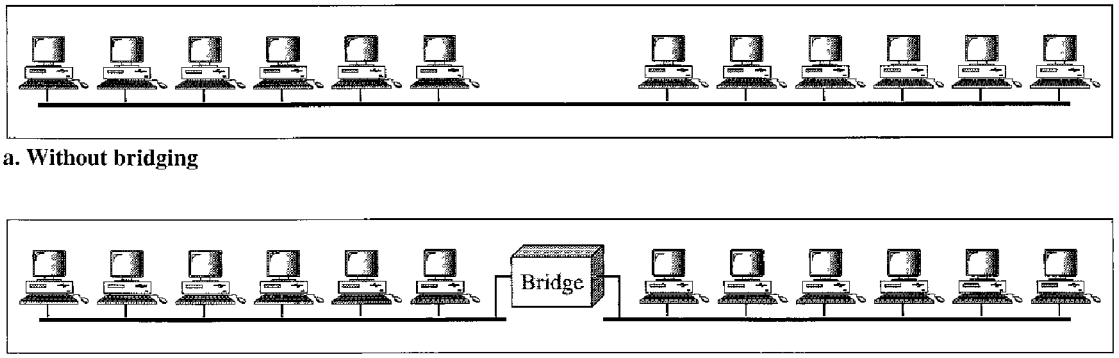


a. First station **b. Second station**

The bridge, as we will learn in Chapter 15, can help here. A bridge divides the network into two or more networks. Bandwidth-wise, each network is independent. For example, in Figure 13.15, a network with 12 stations is divided into two networks, each with 6 stations. Now each network has a capacity of 10 Mbps. The 10-Mbps capacity in each segment is now shared between 6 stations (actually 7 because the bridge acts as a station in each segment), not 12 stations. In a network with a heavy load, each station theoretically is offered 10/6 Mbps instead of 10/12 Mbps, assuming that the traffic is not going through the bridge.

It is obvious that if we further divide the network, we can gain more bandwidth for each segment. For example, if we use a four-port bridge, each station is now offered 10/3 Mbps, which is 4 times more than an unbridged network.

Figure 13.15 *A network with and without a bridge*



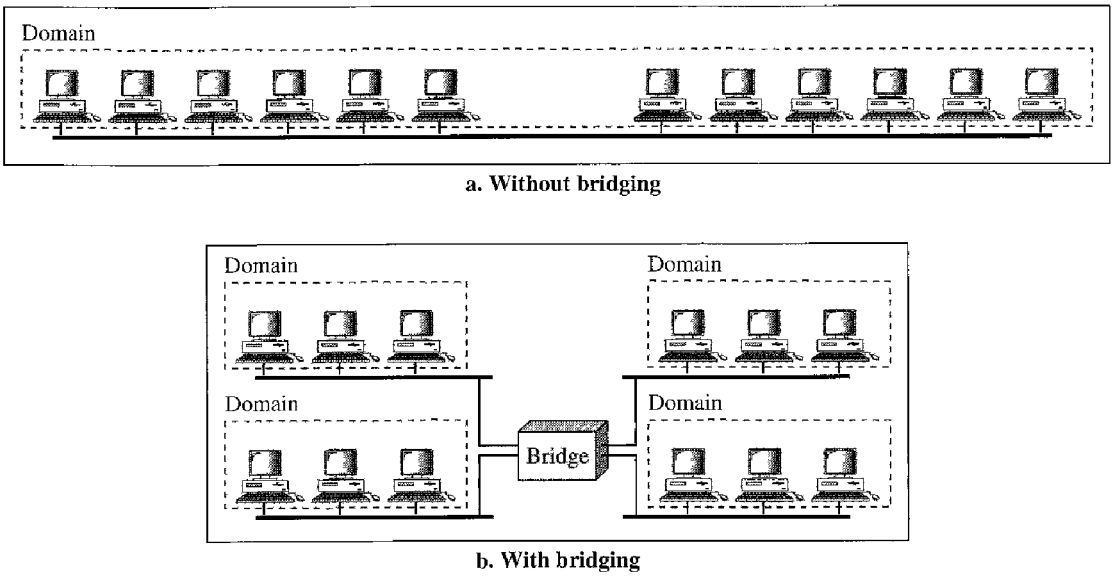
a. Without bridging

b. With bridging

Separating Collision Domains

Another advantage of a bridge is the separation of the **collision domain.** Figure 13.16 shows the collision domains for an unbridged and a bridged network. You can see that the collision domain becomes much smaller and the probability of collision is reduced tremendously. Without bridging, 12 stations contend for access to the medium; with bridging only 3 stations contend for access to the medium.

Figure 13.16 *Collision domains in an unbridged network and a bridged network*



a. Without bridging

b. With bridging

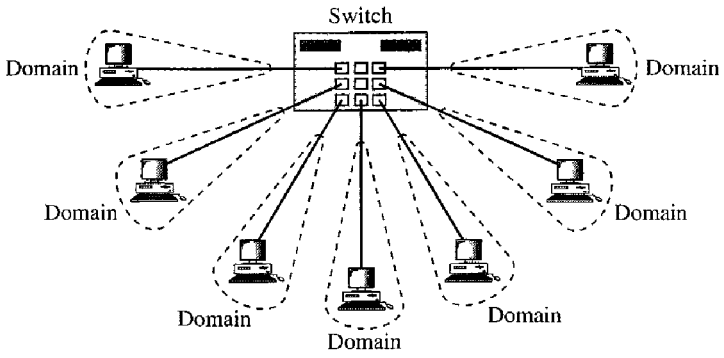
Switched Ethernet

The idea of a bridged LAN can be extended to a switched LAN. Instead of having two to four networks, why not have N networks, where N is the number of stations on the LAN? In other words, if we can have a multiple-port bridge, why not have an N-port

switch? In this way, the bandwidth is shared only between the station and the switch (5 Mbps each). In addition, the collision domain is divided into N domains.

A layer 2 **switch** is an N-port bridge with additional sophistication that allows faster handling of the packets. Evolution from a bridged Ethernet to a **switched Ethernet** was a big step that opened the way to an even faster Ethernet, as we will see. Figure 13.17 shows a switched LAN.

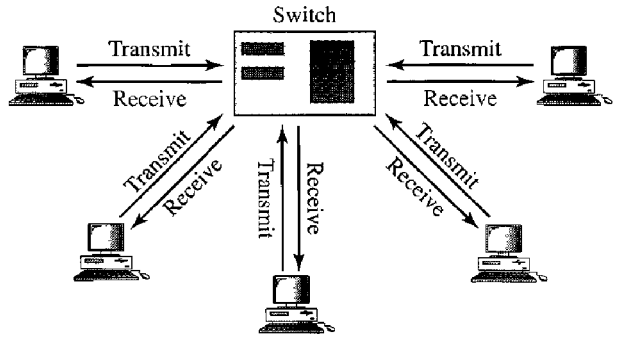
Figure 13.17 *Switched Ethernet*



Full-Duplex Ethernet

One of the limitations of 10Base5 and 10Base2 is that communication is half-duplex (10Base-T is always full-duplex); a station can either send or receive, but may not do both at the same time. The next step in the evolution was to move from switched Ethernet to **full-duplex switched Ethernet.** The full-duplex mode increases the capacity of each domain from 10 to 20 Mbps. Figure 13.18 shows a switched Ethernet in full-duplex mode. Note that instead of using one link between the station and the switch, the configuration uses two links: one to transmit and one to receive.

Figure 13.18 *Full-duplex switched Ethernet*



No Need for CSMA/CD

In full-duplex switched Ethernet, there is no need for the CSMA/CD method. In a full-duplex switched Ethernet, each station is connected to the switch via two separate links.

Each station or switch can send and receive independently without worrying about collision. Each link is a point-to-point dedicated path between the station and the switch. There is no longer a need for carrier sensing; there is no longer a need for collision detection. The job of the MAC layer becomes much easier. The carrier sensing and collision detection functionalities of the MAC sublayer can be turned off.

MAC Control Layer

Standard Ethernet was designed as a connectionless protocol at the MAC sublayer. There is no explicit flow control or error control to inform the sender that the frame has arrived at the destination without error. When the receiver receives the frame, it does not send any positive or negative acknowledgment.

To provide for flow and error control in full-duplex switched Ethernet, a new sublayer, called the MAC control, is added between the LLC sublayer and the MAC sublayer.

13.4 FAST ETHERNET

Fast Ethernet was designed to compete with LAN protocols such as FDDI or Fiber Channel (or Fibre Channel, as it is sometimes spelled). IEEE created Fast Ethernet under the name 802.3u. Fast Ethernet is backward-compatible with Standard Ethernet, but it can transmit data 10 times faster at a rate of 100 Mbps. The goals of Fast Ethernet can be summarized as follows:

1. Upgrade the data rate to 100 Mbps.
2. Make it compatible with Standard Ethernet.
3. Keep the same 48-bit address.
4. Keep the same frame format.
5. Keep the same minimum and maximum frame lengths.

MAC Sublayer

A main consideration in the evolution of Ethernet from 10 to 100 Mbps was to keep the MAC sublayer untouched. However, a decision was made to drop the bus topologies and keep only the star topology. For the star topology, there are two choices, as we saw before: half duplex and full duplex. In the half-duplex approach, the stations are connected via a hub; in the full-duplex approach, the connection is made via a switch with buffers at each port.

The access method is the same (CSMA/CD) for the half-duplex approach; for full-duplex Fast Ethernet, there is no need for CSMA/CD. However, the implementations keep CSMA/CD for backward compatibility with Standard Ethernet.

Autonegotiation

A new feature added to Fast Ethernet is called **autonegotiation.** It allows a station or a hub a range of capabilities. Autonegotiation allows two devices to negotiate the mode

or data rate of operation. It was designed particularly for the following purposes:

- To allow incompatible devices to connect to one another. For example, a device with a maximum capacity of 10 Mbps can communicate with a device with a 100 Mbps capacity (but can work at a lower rate).
- To allow one device to have multiple capabilities.
- To allow a station to check a hub's capabilities.

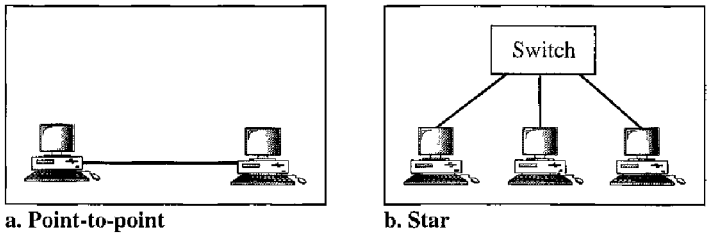
Physical Layer

The physical layer in Fast Ethernet is more complicated than the one in Standard Ethernet. We briefly discuss some features of this layer.

Topology

Fast Ethernet is designed to connect two or more stations together. If there are only two stations, they can be connected point-to-point. Three or more stations need to be connected in a star topology with a hub or a switch at the center, as shown in Figure 13.19.

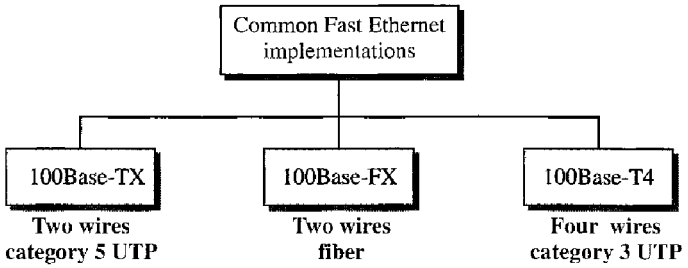
Figure 13.19 *Fast Ethernet topology*



Implementation

Fast Ethernet implementation at the physical layer can be categorized as either two-wire or four-wire. The two-wire implementation can be either category 5 UTP (100Base-TX) or fiber-optic cable (100Base-FX). The four-wire implementation is designed only for category 3 UTP (100Base-T4). See Figure 13.20.

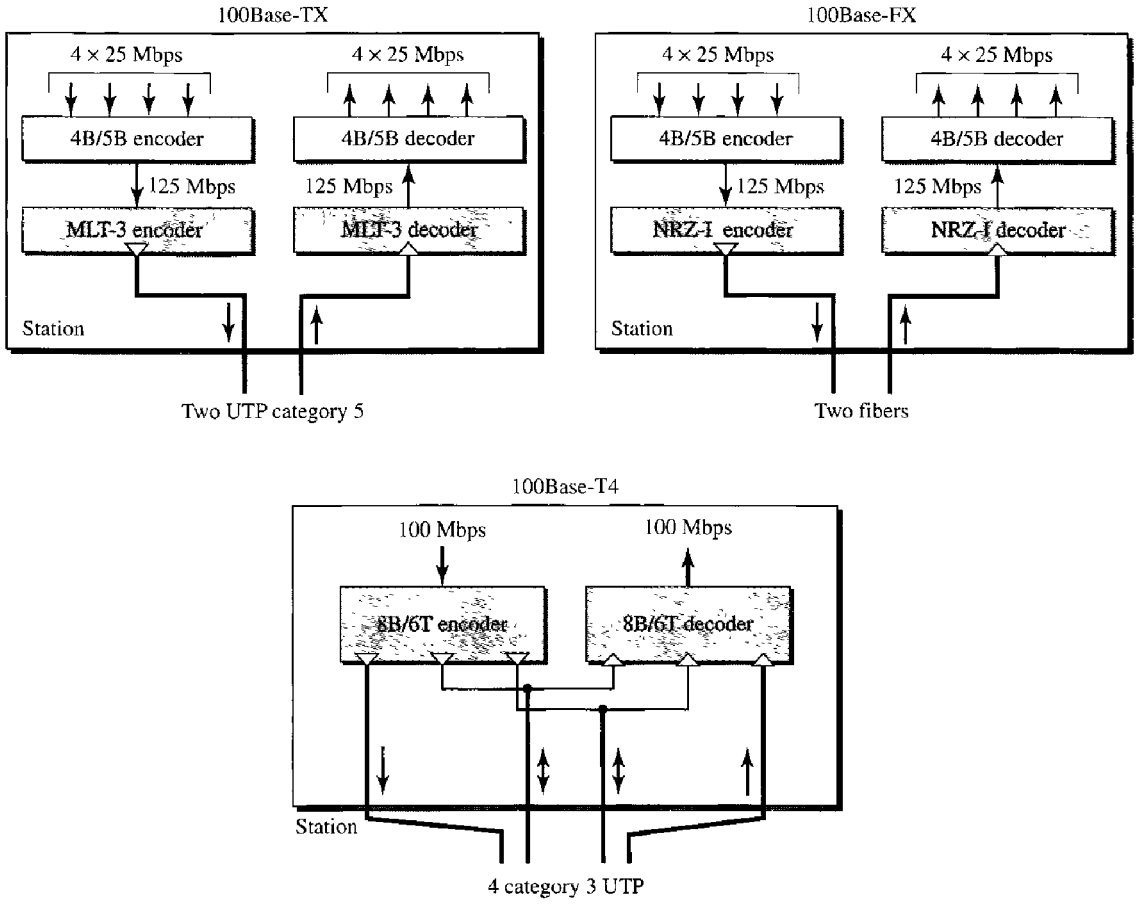
Figure 13.20 *Fast Ethernet implementations*



Encoding

Manchester encoding needs a 200-Mbaud bandwidth for a data rate of 100 Mbps, which makes it unsuitable for a medium such as twisted-pair cable. For this reason, the Fast Ethernet designers sought some alternative encoding/decoding scheme. However, it was found that one scheme would not perform equally well for all three implementations. Therefore, three different encoding schemes were chosen (see Figure 13.21).

Figure 13.21 *Encoding for Fast Ethernet implementation*



100Base-TX uses two pairs of twisted-pair cable (either category 5 UTP or STP). For this implementation, the MLT-3 scheme was selected since it has good bandwidth performance (see Chapter 4). However, since MLT-3 is not a self-synchronous line coding scheme, 4B/5B block coding is used to provide bit synchronization by preventing the occurrence of a long sequence of 0s and 1s (see Chapter 4). This creates a data rate of 125 Mbps, which is fed into MLT-3 for encoding.

100Base-FX uses two pairs of fiber-optic cables. Optical fiber can easily handle high bandwidth requirements by using simple encoding schemes. The designers of 100Base-FX selected the NRZ-I encoding scheme (see Chapter 4) for this implementation. However, NRZ-I has a bit synchronization problem for long sequences of 0s (or 1s, based on the encoding), as we saw in Chapter 4. To overcome this problem, the designers used 4B/5B

block encoding as we described for 100Base-TX. The block encoding increases the bit rate from 100 to 125 Mbps, which can easily be handled by fiber-optic cable.

A 100Base-TX network can provide a data rate of 100 Mbps, but it requires the use of category 5 UTP or STP cable. This is not cost-efficient for buildings that have already been wired for voice-grade twisted-pair (category 3). A new standard, called **100Base-T4,** was designed to use category 3 or higher UTP. The implementation uses four pairs of UTP for transmitting 100 Mbps. Encoding/decoding in 100Base-T4 is more complicated. As this implementation uses category 3 UTP, each twisted-pair cannot easily handle more than 25 Mbaud. In this design, one pair switches between sending and receiving. Three pairs of UTP category 3, however, can handle only 75 Mbaud (25 Mbaud) each. We need to use an encoding scheme that converts 100 Mbps to a 75 Mbaud signal. As we saw in Chapter 4, 8B/6T satisfies this requirement. In 8B/6T, eight data elements are encoded as six signal elements. This means that 100 Mbps uses only (6/8) × 100 Mbps, or 75 Mbaud.

Summary

Table 13.2 is a summary of the Fast Ethernet implementations.

Table 13.2 *Summary of Fast Ethernet implementations*

Characteristics	*100Base-TX*	*100Base-FX*	*100Base-T4*
Media	Cat 5 UTP or STP	Fiber	Cat 4 UTP
Number of wires	2	2	4
Maximum length	100 m	100 m	100 m
Block encoding	4B/5B	4B/5B	
Line encoding	MLT-3	NRZ-I	8B/6T

13.5 GIGABIT ETHERNET

The need for an even higher data rate resulted in the design of the Gigabit Ethernet protocol (1000 Mbps). The IEEE committee calls the Standard 802.3z. The goals of the Gigabit Ethernet design can be summarized as follows:

1. Upgrade the data rate to 1 Gbps.
2. Make it compatible with Standard or Fast Ethernet.
3. Use the same 48-bit address.
4. Use the same frame format.
5. Keep the same minimum and maximum frame lengths.
6. To support autonegotiation as defined in Fast Ethernet.

MAC Sublayer

A main consideration in the evolution of Ethernet was to keep the MAC sublayer untouched. However, to achieve a data rate 1 Gbps, this was no longer possible. Gigabit Ethernet has two distinctive approaches for medium access: half-duplex and full-duplex.

Almost all implementations of Gigabit Ethernet follow the full-duplex approach. However, we briefly discuss the half-duplex approach to show that Gigabit Ethernet can be compatible with the previous generations.

Full-Duplex Mode

In full-duplex mode, there is a central switch connected to all computers or other switches. In this mode, each switch has buffers for each input port in which data are stored until they are transmitted. There is no collision in this mode, as we discussed before. This means that CSMA/CD is not used. Lack of collision implies that the maximum length of the cable is determined by the signal attenuation in the cable, not by the collision detection process.

In the full-duplex mode of Gigabit Ethernet, there is no collision; the maximum length of the cable is determined by the signal attenuation in the cable.

Half-Duplex Mode

Gigabit Ethernet can also be used in half-duplex mode, although it is rare. In this case, a switch can be replaced by a hub, which acts as the common cable in which a collision might occur. The half-duplex approach uses CSMA/CD. However, as we saw before, the maximum length of the network in this approach is totally dependent on the minimum frame size. Three methods have been defined: traditional, carrier extension, and frame bursting.

Traditional In the traditional approach, we keep the minimum length of the frame as in traditional Ethernet (512 bits). However, because the length of a bit is 1/100 shorter in Gigabit Ethernet than in 10-Mbps Ethernet, the slot time for Gigabit Ethernet is 512 bits × 1/1000 μs, which is equal to 0.512 μs. The reduced slot time means that collision is detected 100 times earlier. This means that the maximum length of the network is 25 m. This length may be suitable if all the stations are in one room, but it may not even be long enough to connect the computers in one single office.

Carrier Extension To allow for a longer network, we increase the minimum frame length. The **carrier extension** approach defines the minimum length of a frame as 512 bytes (4096 bits). This means that the minimum length is 8 times longer. This method forces a station to add extension bits (padding) to any frame that is less than 4096 bits. In this way, the maximum length of the network can be increased 8 times to a length of 200 m. This allows a length of 100 m from the hub to the station.

Frame Bursting Carrier extension is very inefficient if we have a series of short frames to send; each frame carries redundant data. To improve efficiency, **frame bursting** was proposed. Instead of adding an extension to each frame, multiple frames are sent. However, to make these multiple frames look like one frame, padding is added between the frames (the same as that used for the carrier extension method) so that the channel is not idle. In other words, the method deceives other stations into thinking that a very large frame has been transmitted.

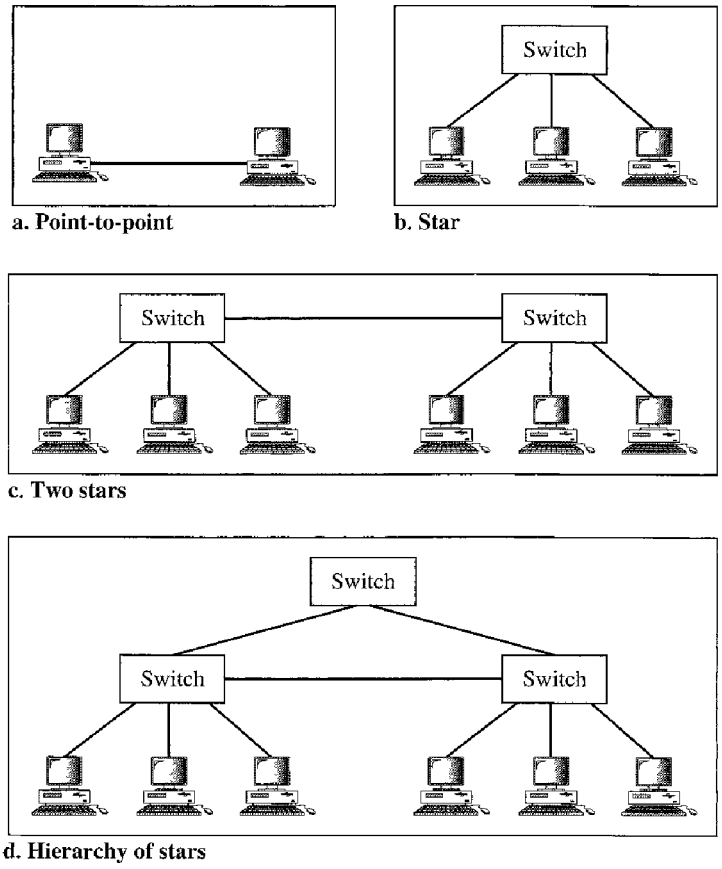
Physical Layer

The physical layer in Gigabit Ethernet is more complicated than that in Standard or Fast Ethernet. We briefly discuss some features of this layer.

Topology

Gigabit Ethernet is designed to connect two or more stations. If there are only two stations, they can be connected point-to-point. Three or more stations need to be connected in a star topology with a hub or a switch at the center. Another possible configuration is to connect several star topologies or let a star topology be part of another as shown in Figure 13.22.

Figure 13.22 *Topologies of Gigabit Ethernet*



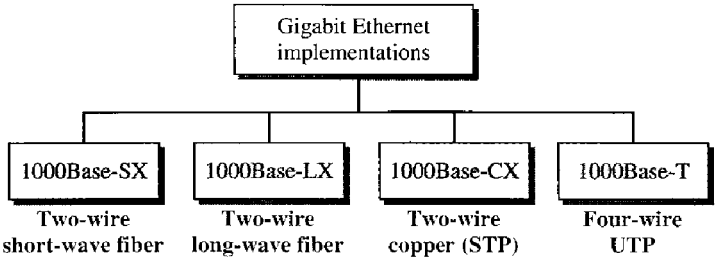
a. Point-to-point
b. Star
c. Two stars
d. Hierarchy of stars

Implementation

Gigabit Ethernet can be categorized as either a two-wire or a four-wire implementation. The two-wire implementations use fiber-optic cable (**1000Base-SX, short-wave,** or **1000Base-LX, long-wave**), or STP (**1000Base-CX**). The four-wire version uses category 5 twisted-pair cable (**1000Base-T**). In other words, we have four implementations, as shown in Figure 13.23. 1000Base-T was designed in response to those users who

had already installed this wiring for other purposes such as Fast Ethernet or telephone services.

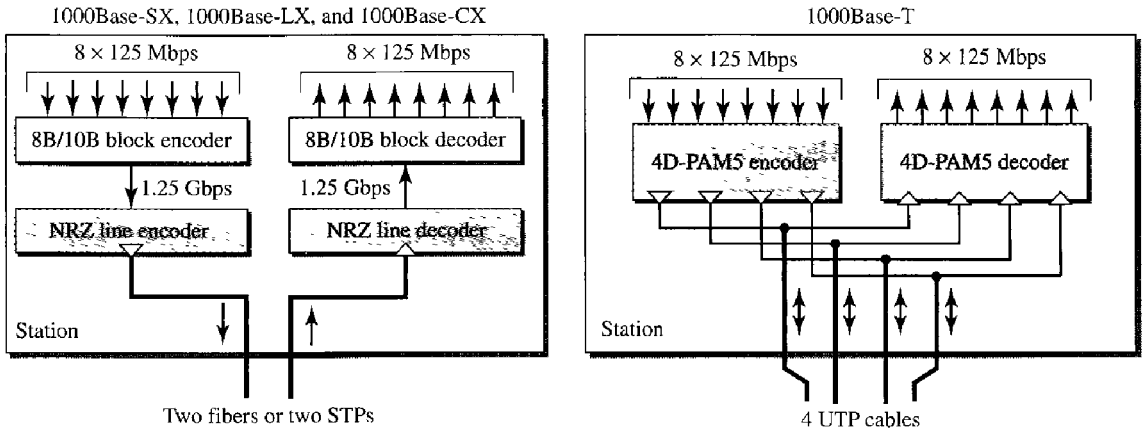
Figure 13.23 *Gigabit Ethernet implementations*



Encoding

Figure 13.24 shows the encoding/decoding schemes for the four implementations.

Figure 13.24 *Encoding in Gigabit Ethernet implementations*



Gigabit Ethernet cannot use the Manchester encoding scheme because it involves a very high bandwidth (2 GBaud). The two-wire implementations use an NRZ scheme, but NRZ does not self-synchronize properly. To synchronize bits, particularly at this high data rate, 8B/10B block encoding, discussed in Chapter 4, is used.

This block encoding prevents long sequences of 0s or 1s in the stream, but the resulting stream is 1.25 Gbps. Note that in this implementation, one wire (fiber or STP) is used for sending and one for receiving.

In the four-wire implementation it is not possible to have 2 wires for input and 2 for output, because each wire would need to carry 500 Mbps, which exceeds the capacity for category 5 UTP. As a solution, 4D-PAM5 encoding, as discussed in Chapter 4, is used to reduce the bandwidth. Thus, all four wires are involved in both input and output; each wire carries 250 Mbps, which is in the range for category 5 UTP cable.

Summary

Table 13.3 is a summary of the Gigabit Ethernet implementations.

Table 13.3 *Summary of Gigabit Ethernet implementations*

Characteristics	*1000Base-SX*	*1000Base-LX*	*1000Base-CX*	*1000Base-T*
Media	Fiber short-wave	Fiber long-wave	STP	Cat 5 UTP
Number of wires	2	2	2	4
Maximum length	550 m	5000 m	25 m	100 m
Block encoding	8B/10B	8B/10B	8B/10B	
Line encoding	NRZ	NRZ	NRZ	4D-PAM5

Ten-Gigabit Ethernet

The IEEE committee created Ten-Gigabit Ethernet and called it Standard 802.3ae. The goals of the Ten-Gigabit Ethernet design can be summarized as follows:

1. Upgrade the data rate to 10 Gbps.
2. Make it compatible with Standard, Fast, and Gigabit Ethernet.
3. Use the same 48-bit address.
4. Use the same frame format.
5. Keep the same minimum and maximum frame lengths.
6. Allow the interconnection of existing LANs into a metropolitan area network (MAN) or a wide area network (WAN).
7. Make Ethernet compatible with technologies such as Frame Relay and ATM (see Chapter 18).

MAC Sublayer

Ten-Gigabit Ethernet operates only in full duplex mode which means there is no need for contention; CSMA/CD is not used in Ten-Gigabit Ethernet.

Physical Layer

The physical layer in Ten-Gigabit Ethernet is designed for using fiber-optic cable over long distances. Three implementations are the most common: **10GBase-S, 10GBase-L,** and **10GBase-E.** Table 13.4 shows a summary of the Ten-Gigabit Ethernet implementaions.

Table 13.4 *Summary of Ten-Gigabit Ethernet implementations*

Characteristics	*10GBase-S*	*10GBase-L*	*10GBase-E*
Media	Short-wave 850-nm multimode	Long-wave 1310-nm single mode	Extended 1550-mm single mode
Maximum length	300 m	10 km	40 km

13.6 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

Ethernet is discussed in Chapters 10, 11, and 12 of [For03], Chapter 5 of [Kei02], Section 4.3 of [Tan03], and Chapters 15 and 16 of [Sta04]. [Spu00] is a book about Ethernet. A complete discussion of Gigabit Ethernet can be found in [KCK98] and [Sau98]. Chapter 2 of [Izz00] has a good comparison between different generations of Ethernet.

13.7 KEY TERMS

1000Base-CX
1000Base-LX
1000Base-SX
1000Base-T
100Base-FX
100Base-T4
100Base-TX
10Base2
10Base5
10Base-F
10Base-T
10GBase-E
10GBase-L
10GBase-S
autonegotiation
bridge
carrier extension
Cheapernet
collision domain
destination service access point (DSAP)
Fast Ethernet
frame bursting
full-duplex switched Ethernet
Gigabit Ethernet
hexadecimal notation
logical link control (LLC)
media access control (MAC)
network interface card (NIC)
preamble
Project 802
source service access point (SSAP)
Standard Ethernet
switch
switched Ethernet
Ten-Gigabit Ethernet
thick Ethernet
Thicknet
thin Ethernet
transceiver
twisted-pair Ethernet

13.8 SUMMARY

- Ethernet is the most widely used local area network protocol.
- The IEEE 802.3 Standard defines 1-persistent CSMA/CD as the access method for first-generation 10-Mbps Ethernet.
- The data link layer of Ethernet consists of the LLC sublayer and the MAC sublayer.

- The MAC sublayer is responsible for the operation of the CSMA/CD access method and framing.
- Each station on an Ethernet network has a unique 48-bit address imprinted on its network interface card (NIC).
- The minimum frame length for 10-Mbps Ethernet is 64 bytes; the maximum is 1518 bytes.
- The common implementations of 10-Mbps Ethernet are 10Base5 (thick Ethernet), 10Base2 (thin Ethernet), 10Base-T (twisted-pair Ethernet), and 10Base-F (fiber Ethernet).
- The 10Base5 implementation of Ethernet uses thick coaxial cable. 10Base2 uses thin coaxial cable. 10Base-T uses four twisted-pair cables that connect each station to a common hub. 10Base-F uses fiber-optic cable.
- A bridge can increase the bandwidth and separate the collision domains on an Ethernet LAN.
- A switch allows each station on an Ethernet LAN to have the entire capacity of the network to itself.
- Full-duplex mode doubles the capacity of each domain and removes the need for the CSMA/CD method.
- Fast Ethernet has a data rate of 100 Mbps.
- In Fast Ethernet, autonegotiation allows two devices to negotiate the mode or data rate of operation.
- The common Fast Ethernet implementations are 100Base-TX (two pairs of twisted-pair cable), 100Base-FX (two fiber-optic cables), and 100Base-T4 (four pairs of voice-grade, or higher, twisted-pair cable).
- Gigabit Ethernet has a data rate of 1000 Mbps.
- Gigabit Ethernet access methods include half-duplex mode using traditional CSMA/CD (not common) and full-duplex mode (most popular method).
- The common Gigabit Ethernet implementations are 1000Base-SX (two optical fibers and a short-wave laser source), 1000Base-LX (two optical fibers and a long-wave laser source), and 1000Base-T (four twisted pairs).
- The latest Ethernet standard is Ten-Gigabit Ethernet that operates at 10 Gbps. The three common implementations are 10GBase-S, 10GBase-L, and 10GBase-E. These implementations use fiber-optic cables in full-duplex mode.

13.9 PRACTICE SET

Review Questions

1. How is the preamble field different from the SFD field?
2. What is the purpose of an NIC?
3. What is the difference between a unicast, multicast, and broadcast address?
4. What are the advantages of dividing an Ethernet LAN with a bridge?
5. What is the relationship between a switch and a bridge?

6. Why is there no need for CSMA/CD on a full-duplex Ethernet LAN?
7. Compare the data rates for Standard Ethernet, Fast Ethernet, Gigabit Ethernet, and Ten-Gigabit Ethernet.
8. What are the common Standard Ethernet implementations?
9. What are the common Fast Ethernet implementations?
10. What are the common Gigabit Ethernet implementations?
11. What are the common Ten-Gigabit Ethernet implementations?

Exercises

12. What is the hexadecimal equivalent of the following Ethernet address?

    01011010 00010001 01010101 00011000 10101010 00001111

13. How does the Ethernet address 1A:2B:3C:4D:5E:6F appear on the line in binary?
14. If an Ethernet destination address is 07:01:02:03:04:05, what is the type of the address (unicast, multicast, or broadcast)?
15. The address 43:7B:6C:DE:10:00 has been shown as the source address in an Ethernet frame. The receiver has discarded the frame. Why?
16. An Ethernet MAC sublayer receives 42 bytes of data from the upper layer. How many bytes of padding must be added to the data?
17. An Ethernet MAC sublayer receives 1510 bytes of data from the upper layer. Can the data be encapsulated in one frame? If not, how many frames need to be sent? What is the size of the data in each frame?
18. What is the ratio of useful data to the entire packet for the smallest Ethernet frame? What is the ratio for the largest frame?
19. Suppose the length of a 10Base5 cable is 2500 m. If the speed of propagation in a thick coaxial cable is 200,000,000 m/s, how long does it take for a bit to travel from the beginning to the end of the network? Assume there are 10 μs delay in the equipment.
20. The data rate of 10Base5 is 10 Mbps. How long does it take to create the smallest frame? Show your calculation.

CHAPTER 14

Wireless LANs

Wireless communication is one of the fastest-growing technologies. The demand for connecting devices without the use of cables is increasing everywhere. **Wireless LANs** can be found on college campuses, in office buildings, and in many public areas.

In this chapter, we concentrate on two promising wireless technologies for LANs: IEEE 802.11 wireless LANs, sometimes called wireless Ethernet, and Bluetooth, a technology for small wireless LANs. Although both protocols need several layers to operate, we concentrate mostly on the physical and data link layers.

14.1 IEEE 802.11

IEEE has defined the specifications for a wireless LAN, called **IEEE 802.11,** which covers the physical and data link layers.

Architecture

The standard defines two kinds of services: the basic service set (BSS) and the extended service set (ESS).

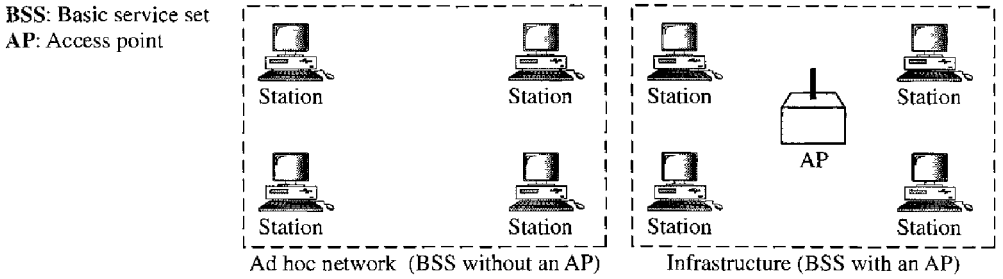
Basic Service Set

IEEE 802.11 defines the **basic service set (BSS)** as the building block of a wireless LAN. A basic service set is made of stationary or mobile wireless stations and an optional central base station, known as the **access point (AP).** Figure 14.1 shows two sets in this standard.

The BSS without an AP is a stand-alone network and cannot send data to other BSSs. It is called an *ad hoc architecture*. In this architecture, stations can form a network without the need of an AP; they can locate one another and agree to be part of a BSS. A BSS with an AP is sometimes referred to as an *infrastructure* network.

A BSS without an AP is called an ad hoc network;
a BSS with an AP is called an infrastructure network.

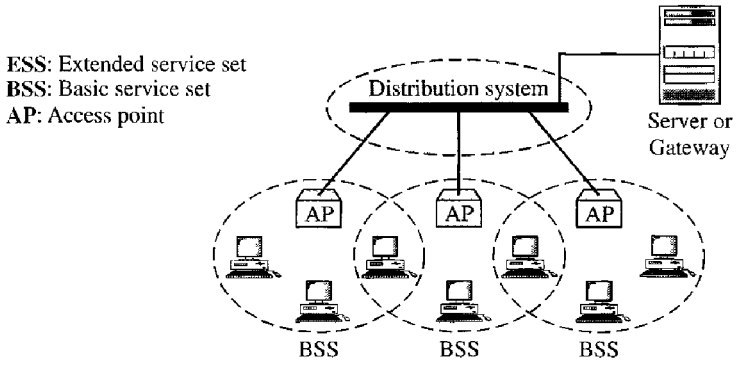
Figure 14.1 *Basic service sets (BSSs)*



Extended Service Set

An **extended service set (ESS)** is made up of two or more BSSs with APs. In this case, the BSSs are connected through a *distribution system,* which is usually a wired LAN. The distribution system connects the APs in the BSSs. IEEE 802.11 does not restrict the distribution system; it can be any IEEE LAN such as an Ethernet. Note that the extended service set uses two types of stations: mobile and stationary. The mobile stations are normal stations inside a BSS. The stationary stations are AP stations that are part of a wired LAN. Figure 14.2 shows an ESS.

Figure 14.2 *Extended service sets (ESSs)*



When BSSs are connected, the stations within reach of one another can communicate without the use of an AP. However, communication between two stations in two different BSSs usually occurs via two APs. The idea is similar to communication in a cellular network if we consider each BSS to be a cell and each AP to be a base station. Note that a mobile station can belong to more than one BSS at the same time.

Station Types

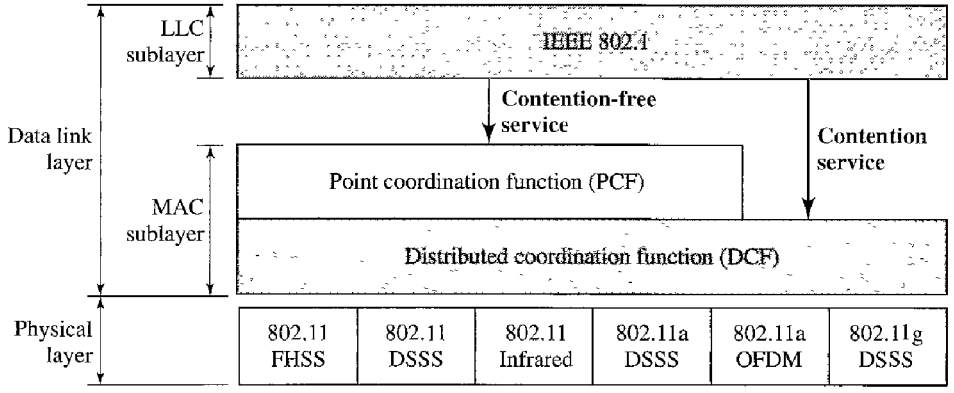
IEEE 802.11 defines three types of stations based on their mobility in a wireless LAN: **no-transition, BSS-transition,** and **ESS-transition mobility.** A station with no-transition

mobility is either stationary (not moving) or moving only inside a BSS. A station with BSS-transition mobility can move from one BSS to another, but the movement is confined inside one ESS. A station with ESS-transition mobility can move from one ESS to another. However, IEEE 802.11 does not guarantee that communication is continuous during the move.

MAC Sublayer

IEEE 802.11 defines two MAC sublayers: the distributed coordination function (DCF) and point coordination function (PCF). Figure 14.3 shows the relationship between the two MAC sublayers, the LLC sublayer, and the physical layer. We discuss the physical layer implementations later in the chapter and will now concentrate on the MAC sublayer.

Figure 14.3 *MAC layers in IEEE 802.11 standard*



Distributed Coordination Function

One of the two protocols defined by IEEE at the MAC sublayer is called the **distributed coordination function (DCF).** DCF uses CSMA/CA (as defined in Chapter 12) as the access method. Wireless LANs cannot implement CSMA/CD for three reasons:

1. For collision detection a station must be able to send data and receive collision signals at the same time. This can mean costly stations and increased bandwidth requirements.
2. Collision may not be detected because of the hidden station problem. We will discuss this problem later in the chapter.
3. The distance between stations can be great. Signal fading could prevent a station at one end from hearing a collision at the other end.

Process Flowchart Figure 14.4 shows the process flowchart for CSMA/CA as used in wireless LANs. We will explain the steps shortly.

Frame Exchange Time Line Figure 14.5 shows the exchange of data and control frames in time.

Figure 14.4 *CSMA/CA flowchart*

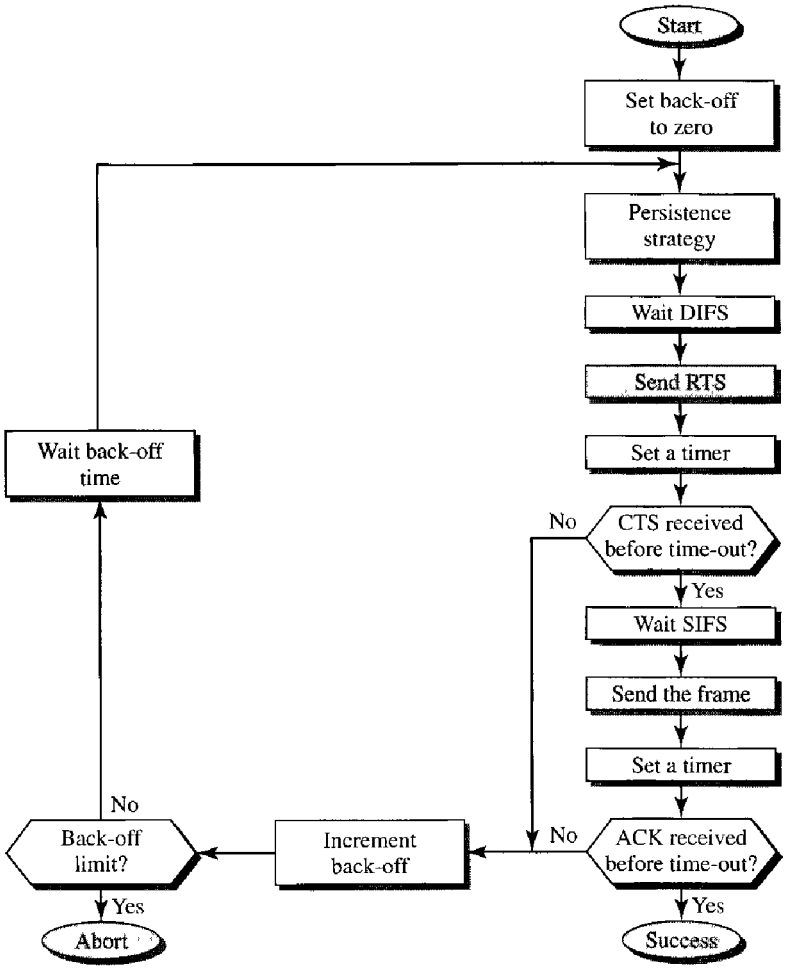
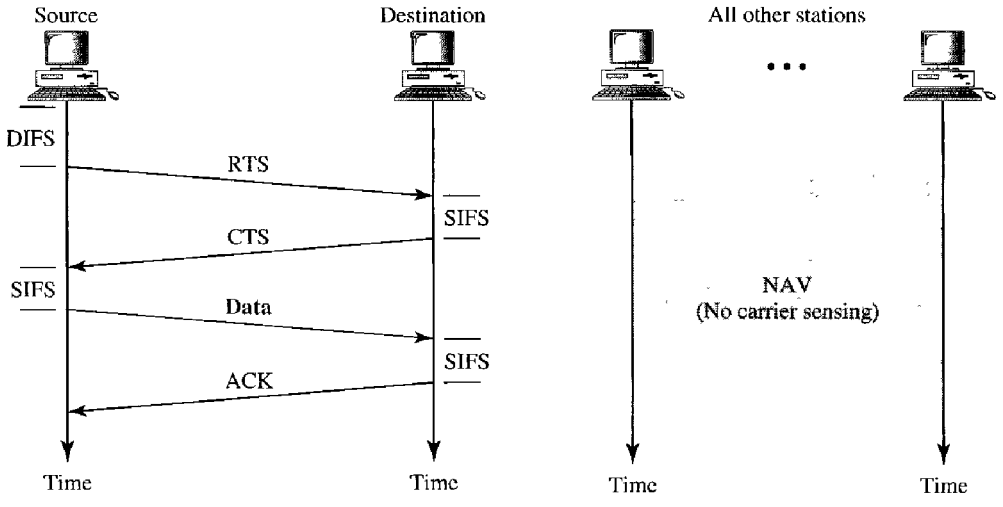


Figure 14.5 *CSMA/CA and NAV*



1. Before sending a frame, the source station senses the medium by checking the energy level at the carrier frequency.
   a. The channel uses a persistence strategy with back-off until the channel is idle.
   b. After the station is found to be idle, the station waits for a period of time called the **distributed interframe space (DIFS);** then the station sends a control frame called the request to send (RTS).
2. After receiving the RTS and waiting a period of time called the **short interframe space (SIFS),** the destination station sends a control frame, called the clear to send (CTS), to the source station. This control frame indicates that the destination station is ready to receive data.
3. The source station sends data after waiting an amount of time equal to SIFS.
4. The destination station, after waiting an amount of time equal to SIFS, sends an acknowledgment to show that the frame has been received. Acknowledgment is needed in this protocol because the station does not have any means to check for the successful arrival of its data at the destination. On the other hand, the lack of collision in CSMA/CD is a kind of indication to the source that data have arrived.

Network Allocation Vector How do other stations defer sending their data if one station acquires access? In other words, how is the *collision avoidance* aspect of this protocol accomplished? The key is a feature called NAV.

When a station sends an RTS frame, it includes the duration of time that it needs to occupy the channel. The stations that are affected by this transmission create a timer called a **network allocation vector (NAV)** that shows how much time must pass before these stations are allowed to check the channel for idleness. Each time a station accesses the system and sends an RTS frame, other stations start their NAV. In other words, each station, before sensing the physical medium to see if it is idle, first checks its NAV to see if it has expired. Figure 14.5 shows the idea of NAV.

Collision During Handshaking What happens if there is collision during the time when RTS or CTS control frames are in transition, often called the **handshaking period**? Two or more stations may try to send RTS frames at the same time. These control frames may collide. However, because there is no mechanism for collision detection, the sender assumes there has been a collision if it has not received a CTS frame from the receiver. The back-off strategy is employed, and the sender tries again.

Point Coordination Function (PCF)

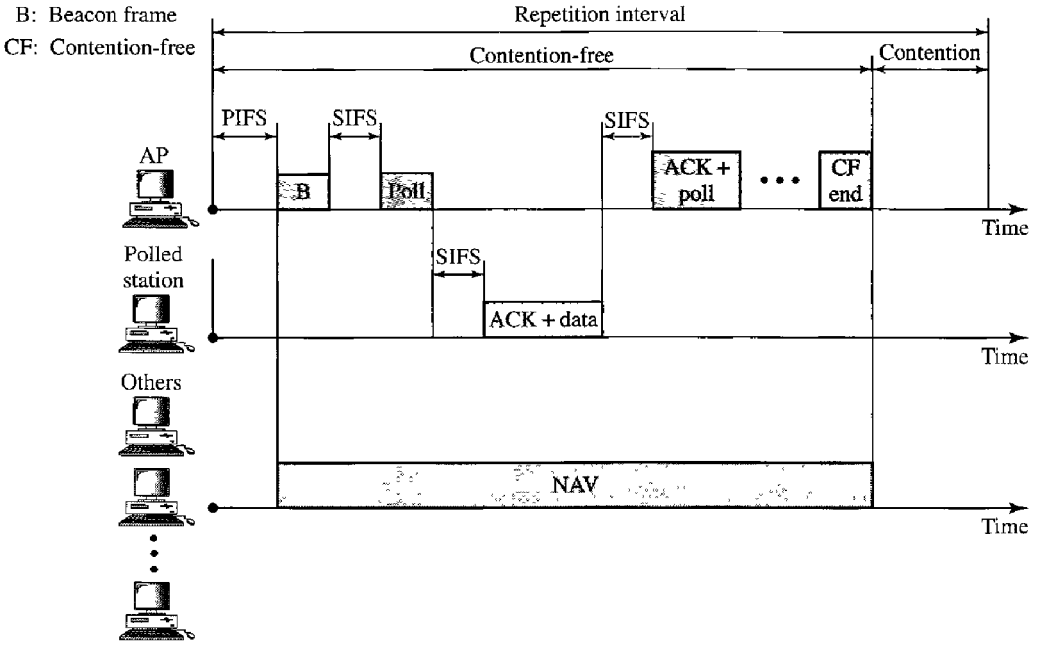
The **point coordination function (PCF)** is an optional access method that can be implemented in an infrastructure network (not in an ad hoc network). It is implemented on top of the DCF and is used mostly for time-sensitive transmission.

PCF has a centralized, contention-free polling access method. The AP performs polling for stations that are capable of being polled. The stations are polled one after another, sending any data they have to the AP.

To give priority to PCF over DCF, another set of interframe spaces has been defined: PIFS and SIFS. The SIFS is the same as that in DCF, but the PIFS (PCF IFS) is shorter than the DIFS. This means that if, at the same time, a station wants to use only DCF and an AP wants to use PCF, the AP has priority.

Due to the priority of PCF over DCF, stations that only use DCF may not gain access to the medium. To prevent this, a repetition interval has been designed to cover both contention-free (PCF) and contention-based (DCF) traffic. The **repetition interval,** which is repeated continuously, starts with a special control frame, called a **beacon frame.** When the stations hear the beacon frame, they start their NAV for the duration of the contention-free period of the repetition interval. Figure 14.6 shows an example of a repetition interval.

Figure 14.6 *Example of repetition interval*



During the repetition interval, the PC (point controller) can send a poll frame, receive data, send an ACK, receive an ACK, or do any combination of these (802.11 uses piggybacking). At the end of the contention-free period, the PC sends a CF end (contention-free end) frame to allow the contention-based stations to use the medium.

Fragmentation

The wireless environment is very noisy; a corrupt frame has to be retransmitted. The protocol, therefore, recommends fragmentation—the division of a large frame into smaller ones. It is more efficient to resend a small frame than a large one.

Frame Format

The MAC layer frame consists of nine fields, as shown in Figure 14.7.

- **Frame control (FC).** The FC field is 2 bytes long and defines the type of frame and some control information. Table 14.1 describes the subfields. We will discuss each frame type later in this chapter.

Figure 14.7 *Frame format*

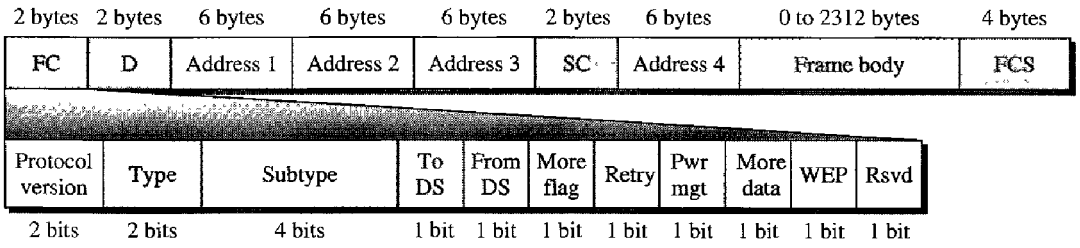


Table 14.1 *Subfields in FC field*

Field	*Explanation*
Version	Current version is 0
Type	Type of information: management (00), control (01), or data (10)
Subtype	Subtype of each type (see Table 14.2)
To DS	Defined later
From DS	Defined later
More flag	When set to 1, means more fragments
Retry	When set to 1, means retransmitted frame
Pwr mgt	When set to 1, means station is in power management mode
More data	When set to 1, means station has more data to send
WEP	Wired equivalent privacy (encryption implemented)
Rsvd	Reserved

- **D.** In all frame types except one, this field defines the duration of the transmission that is used to set the value of NAV. In one control frame, this field defines the ID of the frame.
- **Addresses.** There are four address fields, each 6 bytes long. The meaning of each address field depends on the value of the *To DS* and *From DS* subfields and will be discussed later.
- **Sequence control.** This field defines the sequence number of the frame to be used in flow control.
- **Frame body.** This field, which can be between 0 and 2312 bytes, contains information based on the type and the subtype defined in the FC field.
- **FCS.** The FCS field is 4 bytes long and contains a CRC-32 error detection sequence.

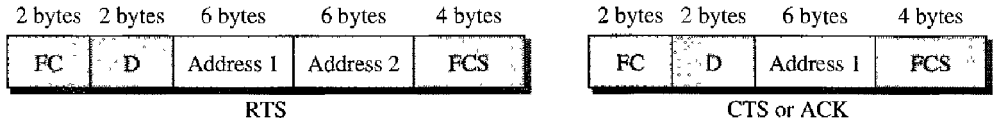
Frame Types

A wireless LAN defined by IEEE 802.11 has three categories of frames: management frames, control frames, and data frames.

Management Frames Management frames are used for the initial communication between stations and access points.

Control Frames Control frames are used for accessing the channel and acknowledging frames. Figure 14.8 shows the format.

Figure 14.8 *Control frames*



For control frames the value of the type field is 01; the values of the subtype fields for frames we have discussed are shown in Table 14.2.

Table 14.2 *Values of subfields in control frames*

Subtype	*Meaning*
1011	Request to send (RTS)
1100	Clear to send (CTS)
1101	Acknowledgment (ACK)

Data Frames Data frames are used for carrying data and control information.

Addressing Mechanism

The IEEE 802.11 addressing mechanism specifies four cases, defined by the value of the two flags in the FC field, *To DS* and *From DS*. Each flag can be either 0 or 1, resulting in four different situations. The interpretation of the four addresses (address 1 to address 4) in the MAC frame depends on the value of these flags, as shown in Table 14.3.

Table 14.3 *Addresses*

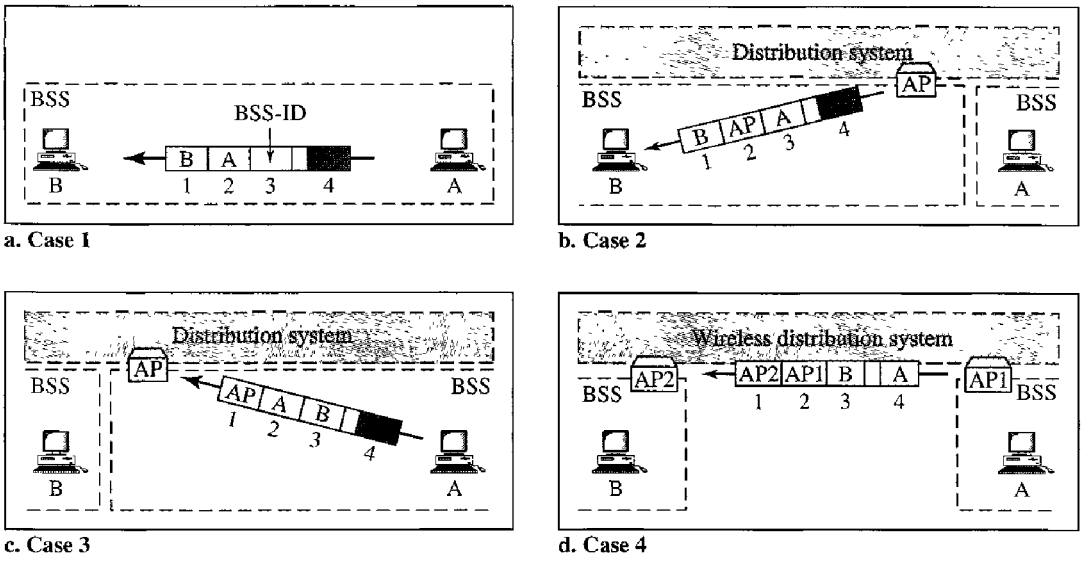
To DS	*From DS*	*Address 1*	*Address 2*	*Address 3*	*Address 4*
0	0	Destination	Source	BSS ID	N/A
0	1	Destination	Sending AP	Source	N/A
1	0	Receiving AP	Source	Destination	N/A
1	1	Receiving AP	Sending AP	Destination	Source

Note that address 1 is always the address of the next device. Address 2 is always the address of the previous device. Address 3 is the address of the final destination station if it is not defined by address 1. Address 4 is the address of the original source station if it is not the same as address 2.

- **Case 1: 00** In this case, *To DS* = 0 and *From DS* = 0. This means that the frame is not going to a distribution system (*To DS* = 0) and is not coming from a distribution

system (*From DS* = 0). The frame is going from one station in a BSS to another without passing through the distribution system. The ACK frame should be sent to the original sender. The addresses are shown in Figure 14.9.

Figure 14.9 *Addressing mechanisms*



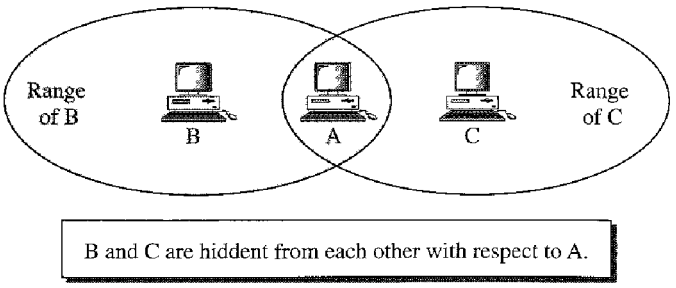
- **Case 2: 01** In this case, *To DS* = 0 and *From DS* = 1. This means that the frame is coming from a distribution system (*From DS* = 1). The frame is coming from an AP and going to a station. The ACK should be sent to the AP. The addresses are as shown in Figure 14.9. Note that address 3 contains the original sender of the frame (in another BSS).
- **Case 3: 10** In this case, *To DS* = 1 and *From DS* = 0. This means that the frame is going to a distribution system (*To DS* = 1). The frame is going from a station to an AP. The ACK is sent to the original station. The addresses are as shown in Figure 14.9. Note that address 3 contains the final destination of the frame (in another BSS).
- **Case 4:11** In this case, *To DS* = 1 and *From DS* = 1. This is the case in which the distribution system is also wireless. The frame is going from one AP to another AP in a wireless distribution system. We do not need to define addresses if the distribution system is a wired LAN because the frame in these cases has the format of a wired LAN frame (Ethernet, for example). Here, we need four addresses to define the original sender, the final destination, and two intermediate APs. Figure 14.9 shows the situation.

Hidden and Exposed Station Problems

We referred to hidden and exposed station problems in the previous section. It is time now to dicuss these problems and their effects.

Hidden Station Problem Figure 14.10 shows an example of the hidden station problem. Station B has a transmission range shown by the left oval (sphere in space); every station in this range can hear any signal transmitted by station B. Station C has

Figure 14.10 *Hidden station problem*



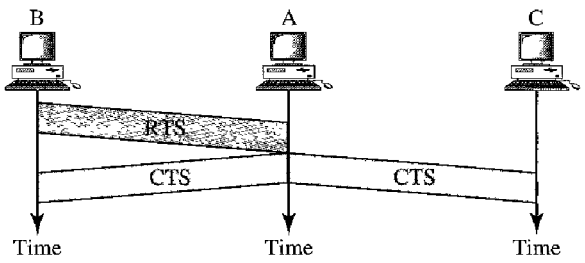
a transmission range shown by the right oval (sphere in space); every station located in this range can hear any signal transmitted by C. Station C is outside the transmission range of B; likewise, station B is outside the transmission range of C. Station A, however, is in the area covered by both B and C; it can hear any signal transmitted by B or C.

Assume that station B is sending data to station A. In the middle of this transmission, station C also has data to send to station A. However, station C is out of B's range and transmissions from B cannot reach C. Therefore C thinks the medium is free. Station C sends its data to A, which results in a collision at A because this station is receiving data from both B and C. In this case, we say that stations B and C are hidden from each other with respect to A. Hidden stations can reduce the capacity of the network because of the possibility of collision.

The solution to the hidden station problem is the use of the handshake frames (RTS and CTS) that we discussed earlier. Figure 14.11 shows that the RTS message from B reaches A, but not C. However, because both B and C are within the range of A, the CTS message, which contains the duration of data transmission from B to A reaches C. Station C knows that some hidden station is using the channel and refrains from transmitting until that duration is over.

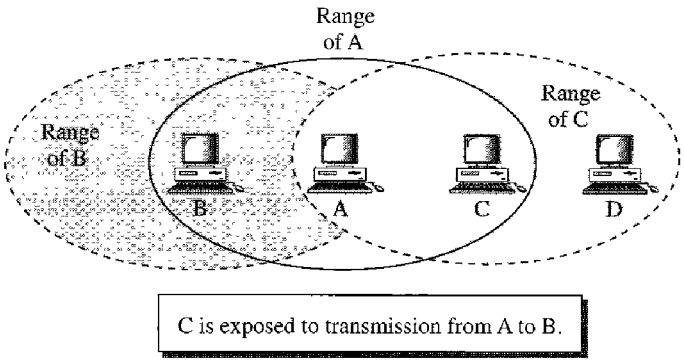
The CTS frame in CSMA/CA handshake can prevent collision from a hidden station.

Figure 14.11 *Use of handshaking to prevent hidden station problem*



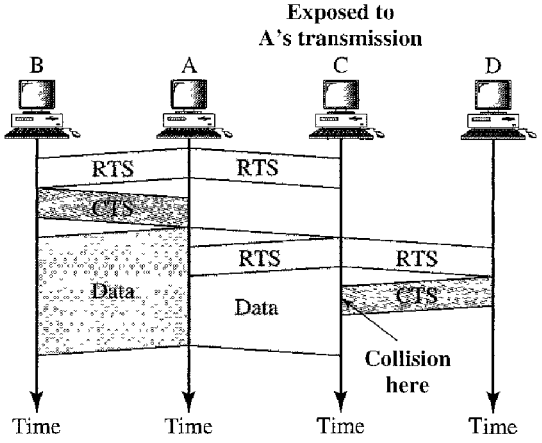
Exposed Station Problem Now consider a situation that is the inverse of the previous one: the exposed station problem. In this problem a station refrains from using a channel when it is, in fact, available. In Figure 14.12, station A is transmitting to station B. Station C has some data to send to station D, which can be sent without interfering with the transmission from A to B. However, station C is exposed to transmission from A; it hears what A is sending and thus refrains from sending. In other words, C is too conservative and wastes the capacity of the channel.

Figure 14.12 *Exposed station problem*



The handshaking messages RTS and CTS cannot help in this case, despite what you might think. Station C hears the RTS from A, but does not hear the CTS from B. Station C, after hearing the RTS from A, can wait for a time so that the CTS from B reaches A; it then sends an RTS to D to show that it needs to communicate with D. Both stations B and A may hear this RTS, but station A is in the sending state, not the receiving state. Station B, however, responds with a CTS. The problem is here. If station A has started sending its data, station C cannot hear the CTS from station D because of the collision; it cannot send its data to D. It remains exposed until A finishes sending its data as Figure 14.13 shows.

Figure 14.13 *Use of handshaking in exposed station problem*



Physical Layer

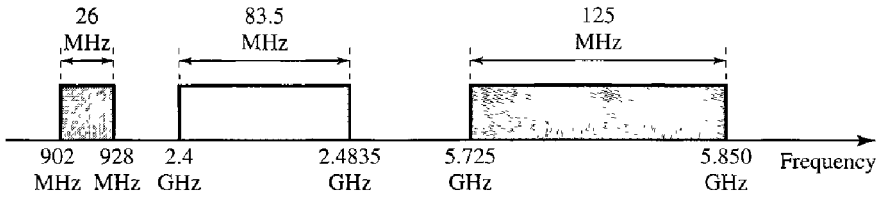
We discuss six specifications, as shown in Table 14.4.

Table 14.4 *Physical layers*

IEEE	*Technique*	*Band*	*Modulation*	*Rate (Mbps)*
802.11	FHSS	2.4 GHz	FSK	1 and 2
	DSSS	2.4 GHz	PSK	1 and 2
		Infrared	PPM	1 and 2
802.11a	OFDM	5.725 GHz	PSK or QAM	6 to 54
802.11b	DSSS	2.4 GHz	PSK	5.5 and 11
802.11g	OFDM	2.4 GHz	Different	22 and 54

All implementations, except the infrared, operate in the *industrial, scientific,* and *medical (ISM)* band, which defines three unlicensed bands in the three ranges 902–928 MHz, 2.400–4.835 GHz, and 5.725–5.850 GHz, as shown in Figure 14.14.

Figure 14.14 *Industrial, scientific, and medical (ISM) band*



IEEE 802.11 FHSS

IEEE 802.11 FHSS uses the frequency-hopping spread spectrum (FHSS) method as discussed in Chapter 6. FHSS uses the 2.4-GHz ISM band. The band is divided into 79 subbands of 1 MHz (and some guard bands). A pseudorandom number generator selects the hopping sequence. The modulation technique in this specification is either two-level FSK or four-level FSK with 1 or 2 bits/baud, which results in a data rate of 1 or 2 Mbps, as shown in Figure 14.15.

IEEE 802.11 DSSS

IEEE 802.11 DSSS uses the direct sequence spread spectrum (DSSS) method as discussed in Chapter 6. DSSS uses the 2.4-GHz ISM band. The modulation technique in this specification is PSK at 1 Mbaud/s. The system allows 1 or 2 bits/baud (BPSK or QPSK), which results in a data rate of 1 or 2 Mbps, as shown in Figure 14.16.

IEEE 802.11 Infrared

IEEE 802.11 infrared uses infrared light in the range of 800 to 950 nm. The modulation technique is called **pulse position modulation (PPM)**. For a 1-Mbps data rate, a 4-bit

Figure 14.15 *Physical layer of IEEE 802.11 FHSS*

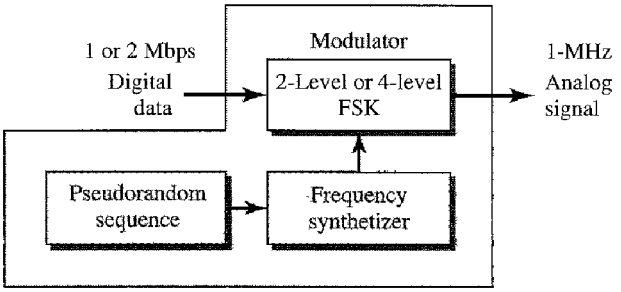
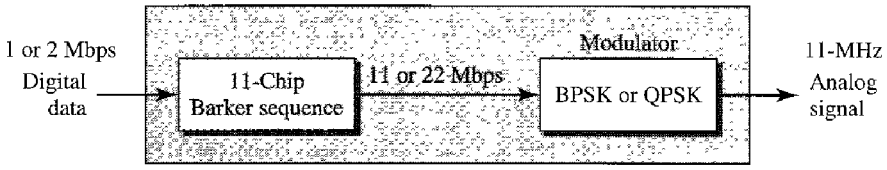
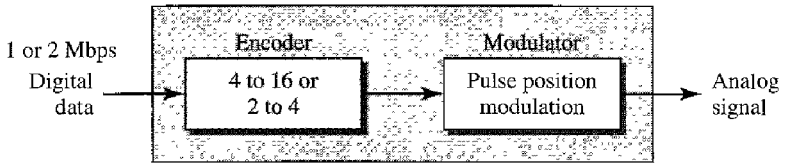


Figure 14.16 *Physical layer of IEEE 802.11 DSSS*



sequence is first mapped into a 16-bit sequence in which only one bit is set to 1 and the rest are set to 0. For a 2-Mbps data rate, a 2-bit sequence is first mapped into a 4-bit sequence in which only one bit is set to 1 and the rest are set to 0. The mapped sequences are then converted to optical signals; the presence of light specifies 1, the absence of light specifies 0. See Figure 14.17.

Figure 14.17 *Physical layer of IEEE 802.11 infrared*



IEEE 802.11a OFDM

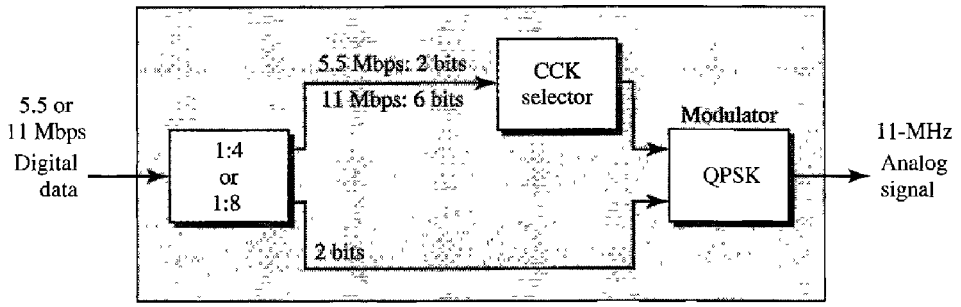
IEEE 802.11a OFDM describes the **orthogonal frequency-division multiplexing (OFDM)** method for signal generation in a 5-GHz ISM band. OFDM is similar to FDM as discussed in Chapter 6, with one major difference: All the subbands are used by one source at a given time. Sources contend with one another at the data link layer for access. The band is divided into 52 subbands, with 48 subbands for sending 48 groups of bits at a time and 4 subbands for control information. The scheme is similar to ADSL, as discussed in Chapter 9. Dividing the band into subbands diminishes the effects of interference. If the subbands are used randomly, security can also be increased.

OFDM uses PSK and QAM for modulation. The common data rates are 18 Mbps (PSK) and 54 Mbps (QAM).

IEEE 802.11b DSSS

IEEE 802.11b DSSS describes the **high-rate direct sequence spread spectrum (HR-DSSS)** method for signal generation in the 2.4-GHz ISM band. HR-DSSS is similar to DSSS except for the encoding method, which is called **complementary code keying (CCK).** CCK encodes 4 or 8 bits to one CCK symbol. To be backward compatible with DSSS, HR-DSSS defines four data rates: 1, 2, 5.5, and 11 Mbps. The first two use the same modulation techniques as DSSS. The 5.5-Mbps version uses BPSK and transmits at 1.375 Mbaud/s with 4-bit CCK encoding. The 11-Mbps version uses QPSK and transmits at 1.375 Mbps with 8-bit CCK encoding. Figure 14.18 shows the modulation technique for this standard.

Figure 14.18 *Physical layer of IEEE 802.11b*



IEEE 802.11g

This new specification defines forward error correction and OFDM using the 2.4-GHz ISM band. The modulation technique achieves a 22- or 54-Mbps data rate. It is backward-compatible with 802.11b, but the modulation technique is OFDM.

14.2 BLUETOOTH

Bluetooth is a wireless LAN technology designed to connect devices of different functions such as telephones, notebooks, computers (desktop and laptop), cameras, printers, coffee makers, and so on. A Bluetooth LAN is an ad hoc network, which means that the network is formed spontaneously; the devices, sometimes called gadgets, find each other and make a network called a piconet. A Bluetooth LAN can even be connected to the Internet if one of the gadgets has this capability. A Bluetooth LAN, by nature, cannot be large. If there are many gadgets that try to connect, there is chaos.

Bluetooth technology has several applications. Peripheral devices such as a wireless mouse or keyboard can communicate with the computer through this technology. Monitoring devices can communicate with sensor devices in a small health care center. Home security devices can use this technology to connect different sensors to the main

security controller. Conference attendees can synchronize their laptop computers at a conference.

Bluetooth was originally started as a project by the Ericsson Company. It is named for Harald Blaatand, the king of Denmark (940–981) who united Denmark and Norway. *Blaatand* translates to *Bluetooth* in English.

Today, Bluetooth technology is the implementation of a protocol defined by the IEEE 802.15 standard. The standard defines a wireless personal-area network (PAN) operable in an area the size of a room or a hall.

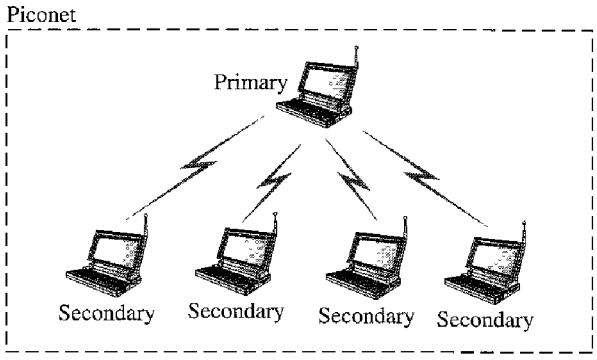
Architecture

Bluetooth defines two types of networks: piconet and scatternet.

Piconets

A Bluetooth network is called a **piconet,** or a small net. A piconet can have up to eight stations, one of which is called the **primary;**[†] the rest are called **secondaries.** All the secondary stations synchronize their clocks and hopping sequence with the primary. Note that a piconet can have only one primary station. The communication between the primary and the secondary can be one-to-one or one-to-many. Figure 14.19 shows a piconet.

Figure 14.19 *Piconet*



Although a piconet can have a maximum of seven secondaries, an additional eight secondaries can be in the *parked state*. A secondary in a parked state is synchronized with the primary, but cannot take part in communication until it is moved from the parked state. Because only eight stations can be active in a piconet, activating a station from the parked state means that an active station must go to the parked state.

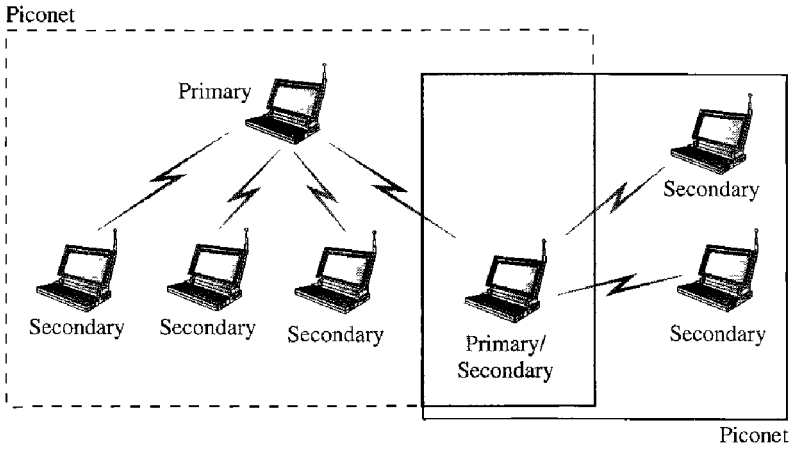
Scatternet

Piconets can be combined to form what is called a **scatternet.** A secondary station in one piconet can be the primary in another piconet. This station can receive messages

[†]The literature sometimes uses the terms *master* and *slave* instead of *primary* and *secondary*. We prefer the latter.

from the primary in the first piconet (as a secondary) and, acting as a primary, deliver them to secondaries in the second piconet. A station can be a member of two piconets. Figure 14.20 illustrates a scatternet.

Figure 14.20 *Scatternet*



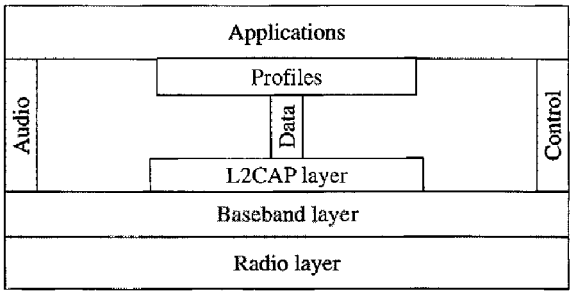
Bluetooth Devices

A Bluetooth device has a built-in short-range radio transmitter. The current data rate is 1 Mbps with a 2.4-GHz bandwidth. This means that there is a possibility of interference between the IEEE 802.11b wireless LANs and Bluetooth LANs.

Bluetooth Layers

Bluetooth uses several layers that do not exactly match those of the Internet model we have defined in this book. Figure 14.21 shows these layers.

Figure 14.21 *Bluetooth layers*



Radio Layer

The radio layer is roughly equivalent to the physical layer of the Internet model. Bluetooth devices are low-power and have a range of 10 m.

Band

Bluetooth uses a 2.4-GHz ISM band divided into 79 channels of 1 MHz each.

FHSS

Bluetooth uses the **frequency-hopping spread spectrum (FHSS)** method in the physical layer to avoid interference from other devices or other networks. Bluetooth hops 1600 times per second, which means that each device changes its modulation frequency 1600 times per second. A device uses a frequency for only 625 μs (1/1600 s) before it hops to another frequency; the dwell time is 625 μs.

Modulation

To transform bits to a signal, Bluetooth uses a sophisticated version of FSK, called GFSK (FSK with Gaussian bandwidth filtering; a discussion of this topic is beyond the scope of this book). GFSK has a carrier frequency. Bit 1 is represented by a frequency deviation above the carrier; bit 0 is represented by a frequency deviation below the carrier. The frequencies, in megahertz, are defined according to the following formula for each channel:

$$f_c = 2402 + n \qquad n = 0, 1, 2, 3, \ldots, 78$$

For example, the first channel uses carrier frequency 2402 MHz (2.402 GHz), and the second channel uses carrier frequency 2403 MHz (2.403 GHz).

Baseband Layer

The baseband layer is roughly equivalent to the MAC sublayer in LANs. The access method is TDMA (see Chapter 12). The primary and secondary communicate with each other using time slots. The length of a time slot is exactly the same as the dwell time, 625 μs. This means that during the time that one frequency is used, a sender sends a frame to a secondary, or a secondary sends a frame to the primary. Note that the communication is only between the primary and a secondary; secondaries cannot communicate directly with one another.

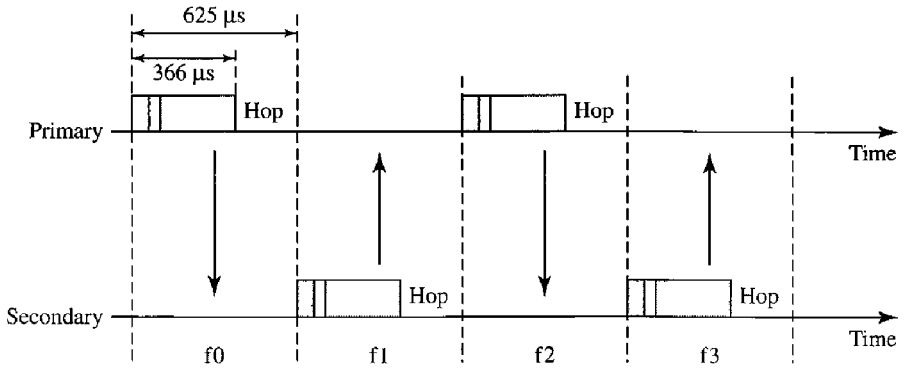
TDMA

Bluetooth uses a form of TDMA (see Chapter 12) that is called **TDD-TDMA (time-division duplex TDMA).** TDD-TDMA is a kind of half-duplex communication in which the secondary and receiver send and receive data, but not at the same time (half-duplex); however, the communication for each direction uses different hops. This is similar to walkie-talkies using different carrier frequencies.

Single-Secondary Communication If the piconet has only one secondary, the TDMA operation is very simple. The time is divided into slots of 625 μs. The primary uses even-numbered slots (0, 2, 4, . . .); the secondary uses odd-numbered slots (1, 3, 5, . . .). TDD-TDMA allows the primary and the secondary to communicate in half-duplex mode.

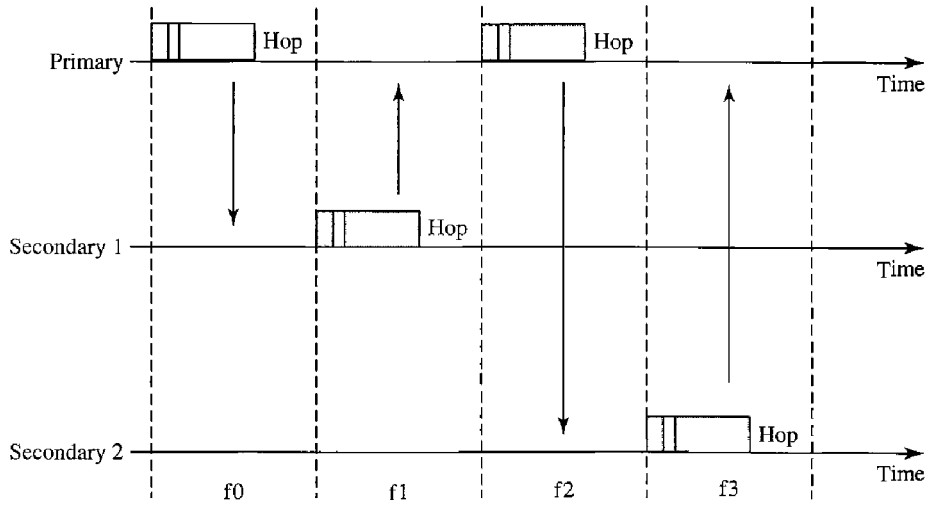
In slot 0, the primary sends, and the secondary receives; in slot 1, the secondary sends, and the primary receives. The cycle is repeated. Figure 14.22 shows the concept.

Figure 14.22 *Single-secondary communication*



Multiple-Secondary Communication The process is a little more involved if there is more than one secondary in the piconet. Again, the primary uses the even-numbered slots, but a secondary sends in the next odd-numbered slot if the packet in the previous slot was addressed to it. All secondaries listen on even-numbered slots, but only one secondary sends in any odd-numbered slot. Figure 14.23 shows a scenario.

Figure 14.23 *Multiple-secondary communication*



Let us elaborate on the figure.

1. In slot 0, the primary sends a frame to secondary 1.
2. In slot 1, only secondary 1 sends a frame to the primary because the previous frame was addressed to secondary 1; other secondaries are silent.

3. In slot 2, the primary sends a frame to secondary 2.
4. In slot 3, only secondary 2 sends a frame to the primary because the previous frame was addressed to secondary 2; other secondaries are silent.
5. The cycle continues.

We can say that this access method is similar to a poll/select operation with reservations. When the primary selects a secondary, it also polls it. The next time slot is reserved for the polled station to send its frame. If the polled secondary has no frame to send, the channel is silent.

Physical Links

Two types of links can be created between a primary and a secondary: SCO links and ACL links.

SCO A **synchronous connection-oriented (SCO)** link is used when avoiding latency (delay in data delivery) is more important than integrity (error-free delivery). In an SCO link, a physical link is created between the primary and a secondary by reserving specific slots at regular intervals. The basic unit of connection is two slots, one for each direction. If a packet is damaged, it is never retransmitted. SCO is used for real-time audio where avoiding delay is all-important. A secondary can create up to three SCO links with the primary, sending digitized audio (PCM) at 64 kbps in each link.

ACL An **asynchronous connectionless link (ACL)** is used when data integrity is more important than avoiding latency. In this type of link, if a payload encapsulated in the frame is corrupted, it is retransmitted. A secondary returns an ACL frame in the available odd-numbered slot if and only if the previous slot has been addressed to it. ACL can use one, three, or more slots and can achieve a maximum data rate of 721 kbps.

Frame Format

A frame in the baseband layer can be one of three types: one-slot, three-slot, or five-slot. A slot, as we said before, is 625 μs. However, in a one-slot frame exchange, 259 μs is needed for hopping and control mechanisms. This means that a one-slot frame can last only 625 – 259, or 366 μs. With a 1-MHz bandwidth and 1 bit/Hz, the size of a one-slot frame is 366 bits.

A three-slot frame occupies three slots. However, since 259 μs is used for hopping, the length of the frame is $3 \times 625 - 259 = 1616$ μs or 1616 bits. A device that uses a three-slot frame remains at the same hop (at the same carrier frequency) for three slots. Even though only one hop number is used, three hop numbers are consumed. That means the hop number for each frame is equal to the first slot of the frame.

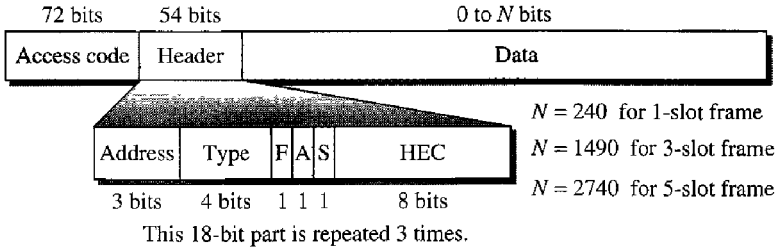
A five-slot frame also uses 259 bits for hopping, which means that the length of the frame is $5 \times 625 - 259 = 2866$ bits.

Figure 14.24 shows the format of the three frame types.

The following describes each field:

- **Access code.** This 72-bit field normally contains synchronization bits and the identifier of the primary to distinguish the frame of one piconet from another.

Figure 14.24 *Frame format types*



- **Header.** This 54-bit field is a repeated 18-bit pattern. Each pattern has the following subfields:
  1. **Address.** The 3-bit address subfield can define up to seven secondaries (1 to 7). If the address is zero, it is used for broadcast communication from the primary to all secondaries.
  2. **Type.** The 4-bit type subfield defines the type of data coming from the upper layers. We discuss these types later.
  3. **F.** This 1-bit subfield is for flow control. When set (1), it indicates that the device is unable to receive more frames (buffer is full).
  4. **A.** This 1-bit subfield is for acknowledgment. Bluetooth uses Stop-and-Wait ARQ; 1 bit is sufficient for acknowledgment.
  5. **S.** This 1-bit subfield holds a sequence number. Bluetooth uses Stop-and-Wait ARQ; 1 bit is sufficient for sequence numbering.
  6. **HEC.** The 8-bit header error correction subfield is a checksum to detect errors in each 18-bit header section.

  The header has three identical 18-bit sections. The receiver compares these three sections, bit by bit. If each of the corresponding bits is the same, the bit is accepted; if not, the majority opinion rules. This is a form of forward error correction (for the header only). This double error control is needed because the nature of the communication, via air, is very noisy. Note that there is no retransmission in this sublayer.
- **Payload.** This subfield can be 0 to 2740 bits long. It contains data or control information coming from the upper layers.

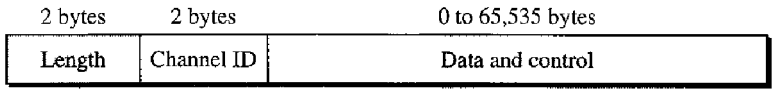
L2CAP

The **Logical Link Control and Adaptation Protocol,** or **L2CAP** (L2 here means LL), is roughly equivalent to the LLC sublayer in LANs. It is used for data exchange on an ACL link; SCO channels do not use L2CAP. Figure 14.25 shows the format of the data packet at this level.

The 16-bit length field defines the size of the data, in bytes, coming from the upper layers. Data can be up to 65,535 bytes. The channel ID (CID) defines a unique identifier for the virtual channel created at this level (see below).

The L2CAP has specific duties: multiplexing, segmentation and reassembly, quality of service (QoS), and group management.

Figure 14.25 *L2CAP data packet format*



Multiplexing

The L2CAP can do multiplexing. At the sender site, it accepts data from one of the upper-layer protocols, frames them, and delivers them to the baseband layer. At the receiver site, it accepts a frame from the baseband layer, extracts the data, and delivers them to the appropriate protocol layer. It creates a kind of virtual channel that we will discuss in later chapters on higher-level protocols.

Segmentation and Reassembly

The maximum size of the payload field in the baseband layer is 2774 bits, or 343 bytes. This includes 4 bytes to define the packet and packet length. Therefore, the size of the packet that can arrive from an upper layer can only be 339 bytes. However, application layers sometimes need to send a data packet that can be up to 65,535 bytes (an Internet packet, for example). The L2CAP divides these large packets into segments and adds extra information to define the location of the segments in the original packet. The L2CAP segments the packet at the source and reassembles them at the destination.

QoS

Bluetooth allows the stations to define a quality-of-service level. We discuss quality of service in Chapter 24. For the moment, it is sufficient to know that if no quality-of-service level is defined, Bluetooth defaults to what is called *best-effort* service; it will do its best under the circumstances.

Group Management

Another functionality of L2CAP is to allow devices to create a type of logical addressing between themselves. This is similar to multicasting. For example, two or three secondary devices can be part of a multicast group to receive data from the primary.

Other Upper Layers

Bluetooth defines several protocols for the upper layers that use the services of L2CAP; these protocols are specific for each purpose.

14.3 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

Wireless LANs and Bluetooth are discussed in several books including [Sch03] and [Gas02]. Wireless LANs are discussed in Chapter 15 of [For03], Chapter 17 of [Sta04], Chapters 13 and 14 of [Sta02], and Chapter 8 of [Kei02]. Bluetooth is discussed in Chapter 15 of [Sta02] and Chapter 15 of [For03].

14.4 KEY TERMS

access point (AP)
asynchronous connectionless link (ACL)
basic service set (BSS)
beacon frame
Bluetooth
BSS-transition mobility
complementary code keying (CCK)
direct sequence spread spectrum (DSSS)
distributed coordination function (DCF)
distributed interframe space (DIFS)
ESS-transition mobility
extended service set (ESS)
frequency-hopping spread spectrum (FHSS)
handshaking period
high-rate direct sequence spread spectrum (HR-DSSS)
IEEE 802.11
Logical Link Control and Adaptation Protocol (L2CAP)
network allocation vector (NAV)
no-transition mobility
orthogonal frequency-division multiplexing (OFDM)
piconet
point coordination function (PCF)
primary
pulse position modulation (PPM)
repetition interval
scatternet
secondary
short interframe space (SIFS)
synchronous connection-oriented (SCO)
TDD-TDMA (time-division duplex TDMA)
wireless LAN

14.5 SUMMARY

- The IEEE 802.11 standard for wireless LANs defines two services: basic service set (BSS) and extended service set (ESS).
- The access method used in the distributed coordination function (DCF) MAC sublayer is CSMA/CA.
- The access method used in the point coordination function (PCF) MAC sublayer is polling.
- The network allocation vector (NAV) is a timer used for collision avoidance.
- The MAC layer frame has nine fields. The addressing mechanism can include up to four addresses.
- Wireless LANs use management frames, control frames, and data frames.

- IEEE 802.11 defines several physical layers, with different data rates and modulating techniques.
- Bluetooth is a wireless LAN technology that connects devices (called gadgets) in a small area.
- A Bluetooth network is called a piconet. Multiple piconets form a network called a scatternet.
- A Bluetooth network consists of one primary device and up to seven secondary devices.

14.6 PRACTICE SET

Review Questions

1. What is the difference between a BSS and an ESS?
2. Discuss the three types of mobility in a wireless LAN.
3. How is OFDM different from FDM?
4. What is the access method used by wireless LANs?
5. What is the purpose of the NAV?
6. Compare a piconet and a scatternet.
7. Match the layers in Bluetooth and the Internet model.
8. What are the two types of links between a Bluetooth primary and a Bluetooth secondary?
9. In multiple-secondary communication, who uses the even-numbered slots and who uses the odd-numbered slots?
10. How much time in a Bluetooth one-slot frame is used for the hopping mechanism? What about a three-slot frame and a five-slot frame?

Exercises

11. Compare and contrast CSMA/CD with CSMA/CA.
12. Use Table 14.5 to compare and contrast the fields in IEEE 802.3 and 802.11.

Table 14.5 *Exercise 12*

Fields	*IEEE 802.3 Field Size*	*IEEE 802.11 Field Size*
Destination address		
Source address		
Address 1		
Address 2		
Address 3		
Address 4		
FC		

Table 14.5 *Exercise 12 (continued)*

Fields	*IEEE 802.3 Field Size*	*IEEE 802.11 Field Size*
D/ID		
SC		
PDU length		
Data and padding		
Frame body		
FCS (CRC)		

CHAPTER 15

Connecting LANs, Backbone Networks, and Virtual LANs

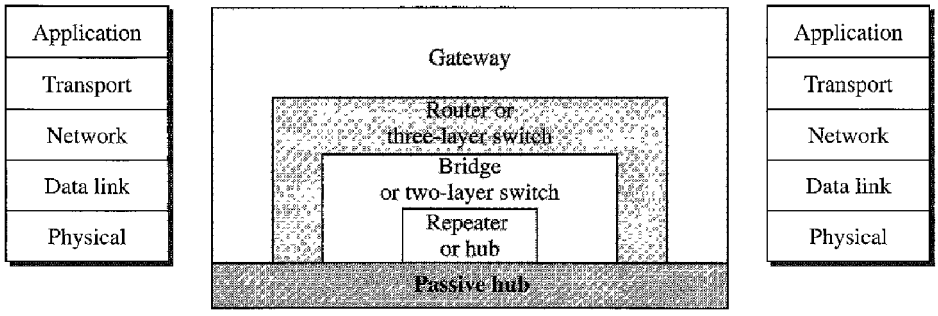
LANs do not normally operate in isolation. They are connected to one another or to the Internet. To connect LANs, or segments of LANs, we use connecting devices. Connecting devices can operate in different layers of the Internet model. In this chapter, we discuss only those that operate in the physical and data link layers; we discuss those that operate in the first three layers in Chapter 19.

After discussing some connecting devices, we show how they are used to create backbone networks. Finally, we discuss virtual local area networks (VLANs).

15.1 CONNECTING DEVICES

In this section, we divide **connecting devices** into five different categories based on the layer in which they operate in a network, as shown in Figure 15.1.

Figure 15.1 *Five categories of connecting devices*



The five categories contain devices which can be defined as

1. Those which operate below the physical layer such as a passive hub.
2. Those which operate at the physical layer (a repeater or an active hub).
3. Those which operate at the physical and data link layers (a bridge or a two-layer switch).

4. Those which operate at the physical, data link, and network layers (a router or a three-layer switch).
5. Those which can operate at all five layers (a gateway).

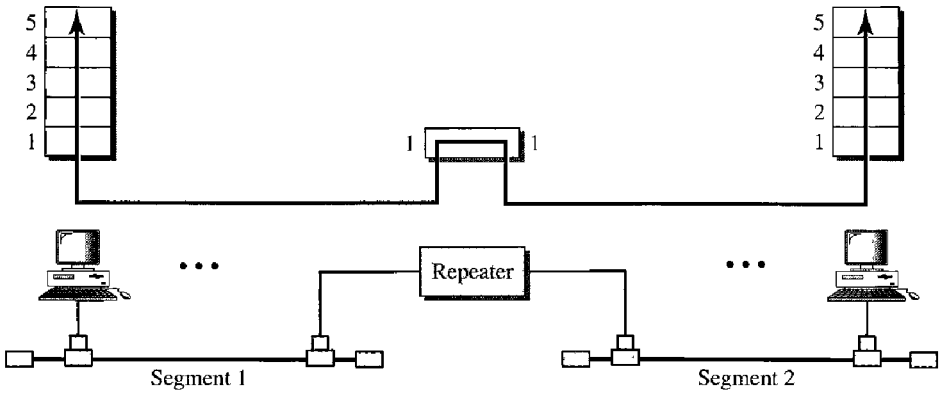
Passive Hubs

A passive hub is just a connector. It connects the wires coming from different branches. In a star-topology Ethernet LAN, a passive hub is just a point where the signals coming from different stations collide; the hub is the collision point. This type of a hub is part of the media; its location in the Internet model is below the physical layer.

Repeaters

A **repeater** is a device that operates only in the physical layer. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data. A repeater receives a signal and, before it becomes too weak or corrupted, regenerates the original bit pattern. The repeater then sends the refreshed signal. A repeater can extend the physical length of a LAN, as shown in Figure 15.2.

Figure 15.2 *A repeater connecting two segments of a LAN*



A repeater does not actually connect two LANs; it connects two segments of the same LAN. The segments connected are still part of one single LAN. A repeater is not a device that can connect two LANs of different protocols.

A repeater connects segments of a LAN.

A repeater can overcome the 10Base5 Ethernet length restriction. In this standard, the length of the cable is limited to 500 m. To extend this length, we divide the cable into segments and install repeaters between segments. Note that the whole network is still considered one LAN, but the portions of the network separated by repeaters are called **segments.** The repeater acts as a two-port node, but operates only in the physical layer. When it receives a frame from any of the ports, it regenerates and forwards it to the other port.

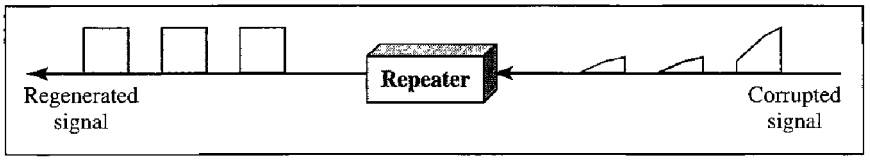
A repeater forwards every frame; it has no filtering capability.

It is tempting to compare a repeater to an amplifier, but the comparison is inaccurate. An **amplifier** cannot discriminate between the intended signal and noise; it amplifies equally everything fed into it. A repeater does not amplify the signal; it regenerates the signal. When it receives a weakened or corrupted signal, it creates a copy, bit for bit, at the original strength.

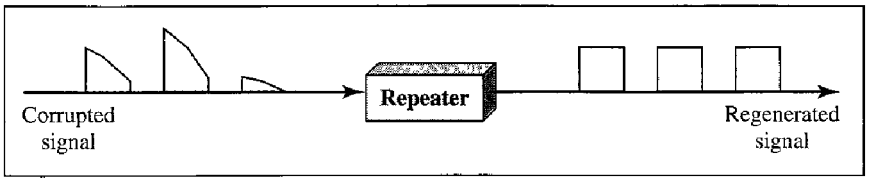
A repeater is a regenerator, not an amplifier.

The location of a repeater on a link is vital. A repeater must be placed so that a signal reaches it before any noise changes the meaning of any of its bits. A little noise can alter the precision of a bit's voltage without destroying its identity (see Figure 15.3). If the corrupted bit travels much farther, however, accumulated noise can change its meaning completely. At that point, the original voltage is not recoverable, and the error needs to be corrected. A repeater placed on the line before the legibility of the signal becomes lost can still read the signal well enough to determine the intended voltages and replicate them in their original form.

Figure 15.3 *Function of a repeater*



a. Right-to-left transmission.



b. Left-to-right transmission.

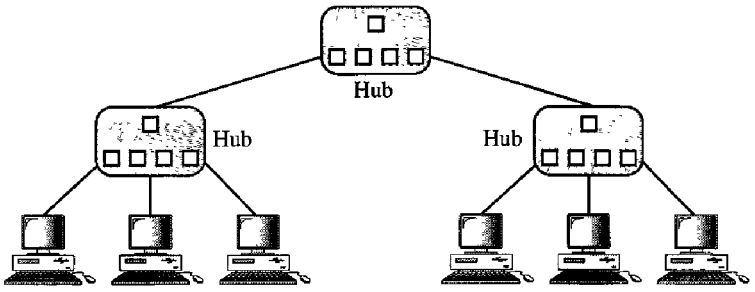
Active Hubs

An active **hub** is actually a multiport repeater. It is normally used to create connections between stations in a physical star topology. We have seen examples of hubs in some Ethernet implementations (10Base-T, for example). However, hubs can also be used to create multiple levels of hierarchy, as shown in Figure 15.4. The hierarchical use of hubs removes the length limitation of 10Base-T (100 m).

Bridges

A **bridge** operates in both the physical and the data link layer. As a physical layer device, it regenerates the signal it receives. As a data link layer device, the bridge can check the physical (MAC) addresses (source and destination) contained in the frame.

Figure 15.4 *A hierarchy of hubs*



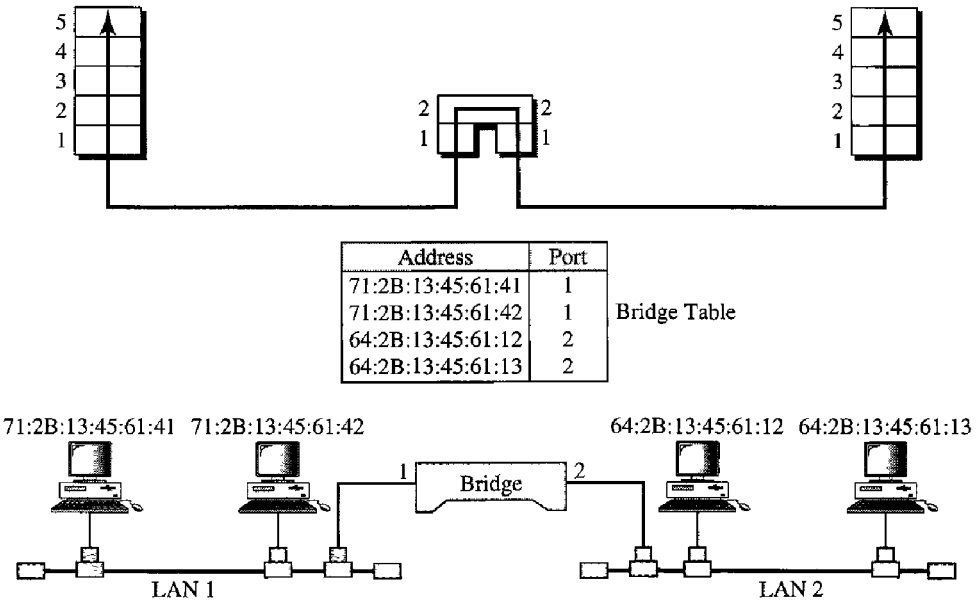
Filtering

One may ask, What is the difference in functionality between a bridge and a repeater? A bridge has **filtering** capability. It can check the destination address of a frame and decide if the frame should be forwarded or dropped. If the frame is to be forwarded, the decision must specify the port. A bridge has a table that maps addresses to ports.

A bridge has a table used in filtering decisions.

Let us give an example. In Figure 15.5, two LANs are connected by a bridge. If a frame destined for station 712B13456142 arrives at port 1, the bridge consults its table to find the departing port. According to its table, frames for 712B13456142 leave through port 1; therefore, there is no need for forwarding, and the frame is dropped. On the other hand, if a frame for 712B13456141 arrives at port 2, the departing port is port 1

Figure 15.5 *A bridge connecting two LANs*



and the frame is forwarded. In the first case, LAN 2 remains free of traffic; in the second case, both LANs have traffic. In our example, we show a two-port bridge; in reality a bridge usually has more ports.

Note also that a bridge does not change the physical addresses contained in the frame.

A bridge does not change the physical (MAC) addresses in a frame.

Transparent Bridges

A **transparent bridge** is a bridge in which the stations are completely unaware of the bridge's existence. If a bridge is added or deleted from the system, reconfiguration of the stations is unnecessary. According to the IEEE 802.1d specification, a system equipped with transparent bridges must meet three criteria:

1. Frames must be forwarded from one station to another.
2. The forwarding table is automatically made by learning frame movements in the network.
3. Loops in the system must be prevented.

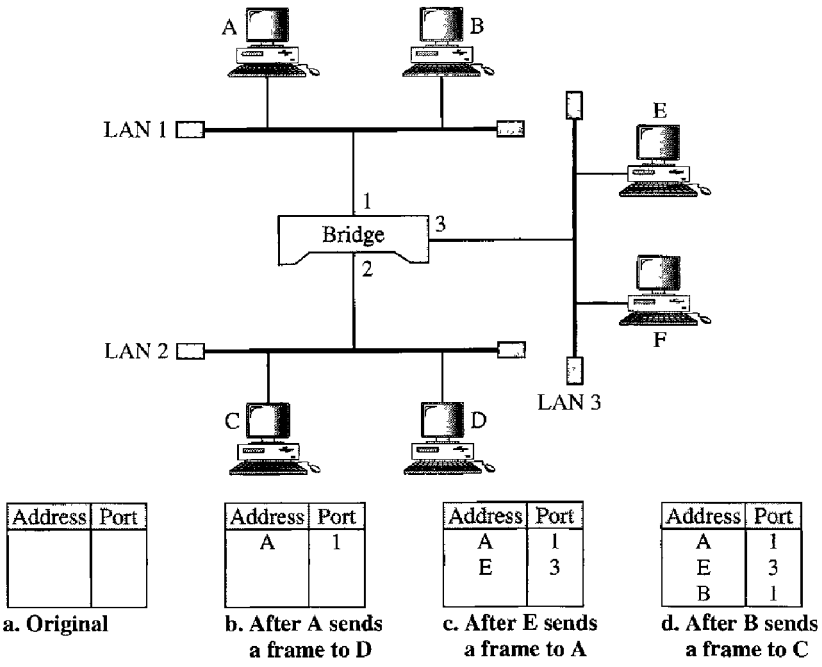
Forwarding A transparent bridge must correctly forward the frames, as discussed in the previous section.

Learning The earliest bridges had forwarding tables that were static. The systems administrator would manually enter each table entry during bridge setup. Although the process was simple, it was not practical. If a station was added or deleted, the table had to be modified manually. The same was true if a station's MAC address changed, which is not a rare event. For example, putting in a new network card means a new MAC address.

A better solution to the static table is a dynamic table that maps addresses to ports automatically. To make a table dynamic, we need a bridge that gradually learns from the frame movements. To do this, the bridge inspects both the destination and the source addresses. The destination address is used for the forwarding decision (table lookup); the source address is used for adding entries to the table and for updating purposes. Let us elaborate on this process by using Figure 15.6.

1. When station A sends a frame to station D, the bridge does not have an entry for either D or A. The frame goes out from all three ports; the frame floods the network. However, by looking at the source address, the bridge learns that station A must be located on the LAN connected to port 1. This means that frames destined for A, in the future, must be sent out through port 1. The bridge adds this entry to its table. The table has its first entry now.
2. When station E sends a frame to station A, the bridge has an entry for A, so it forwards the frame only to port 1. There is no flooding. In addition, it uses the source address of the frame, E, to add a second entry to the table.
3. When station B sends a frame to C, the bridge has no entry for C, so once again it floods the network and adds one more entry to the table.
4. The process of learning continues as the bridge forwards frames.

Figure 15.6 *A learning bridge and the process of learning*

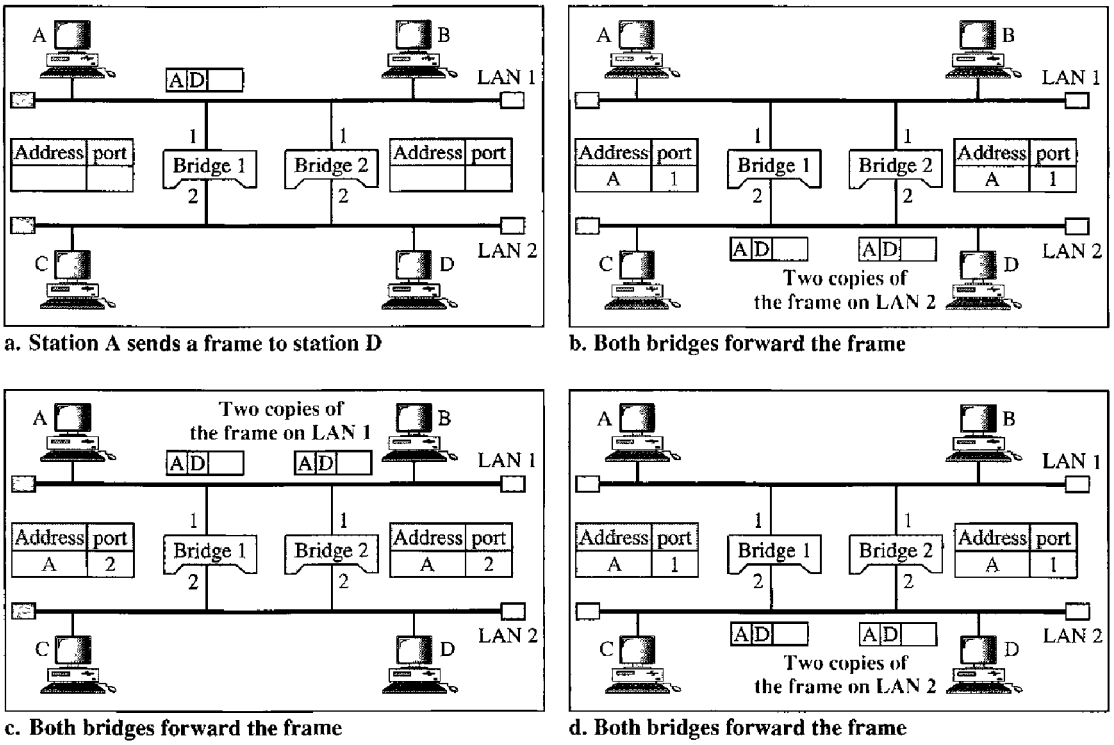


Loop Problem Transparent bridges work fine as long as there are no redundant bridges in the system. Systems administrators, however, like to have redundant bridges (more than one bridge between a pair of LANs) to make the system more reliable. If a bridge fails, another bridge takes over until the failed one is repaired or replaced. Redundancy can create loops in the system, which is very undesirable. Figure 15.7 shows a very simple example of a loop created in a system with two LANs connected by two bridges.

1. Station A sends a frame to station D. The tables of both bridges are empty. Both forward the frame and update their tables based on the source address A.
2. Now there are two copies of the frame on LAN 2. The copy sent out by bridge 1 is received by bridge 2, which does not have any information about the destination address D; it floods the bridge. The copy sent out by bridge 2 is received by bridge 1 and is sent out for lack of information about D. Note that each frame is handled separately because bridges, as two nodes on a network sharing the medium, use an access method such as CSMA/CD. The tables of both bridges are updated, but still there is no information for destination D.
3. Now there are two copies of the frame on LAN 1. Step 2 is repeated, and both copies flood the network.
4. The process continues on and on. Note that bridges are also repeaters and regenerate frames. So in each iteration, there are newly generated fresh copies of the frames.

To solve the looping problem, the IEEE specification requires that bridges use the spanning tree algorithm to create a loopless topology.

Figure 15.7 *Loop problem in a learning bridge*



a. Station A sends a frame to station D

b. Both bridges forward the frame

c. Both bridges forward the frame

d. Both bridges forward the frame

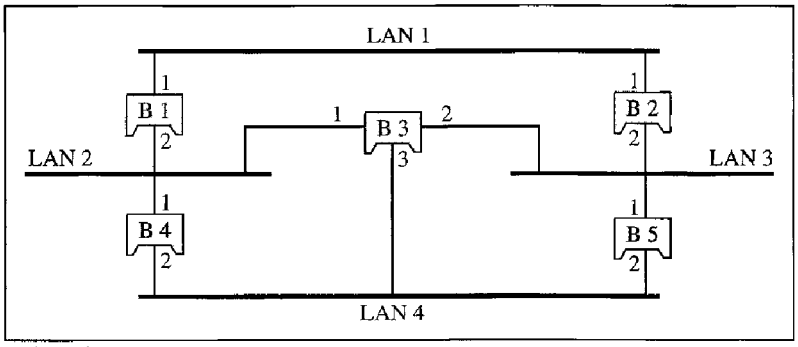
Spanning Tree

In graph theory, a **spanning tree** is a graph in which there is no loop. In a bridged LAN, this means creating a topology in which each LAN can be reached from any other LAN through one path only (no loop). We cannot change the physical topology of the system because of physical connections between cables and bridges, but we can create a logical topology that overlays the physical one. Figure 15.8 shows a system with four LANs and five bridges. We have shown the physical system and its representation in graph theory. Although some textbooks represent the LANs as nodes and the bridges as the connecting arcs, we have shown both LANs and bridges as nodes. The connecting arcs show the connection of a LAN to a bridge and vice versa. To find the spanning tree, we need to assign a cost (metric) to each arc. The interpretation of the cost is left up to the systems administrator. It may be the path with minimum hops (nodes), the path with minimum delay, or the path with maximum bandwidth. If two ports have the same shortest value, the systems administrator just chooses one. We have chosen the minimum hops. However, as we will see in Chapter 22, the hop count is normally 1 from a bridge to the LAN and 0 in the reverse direction.

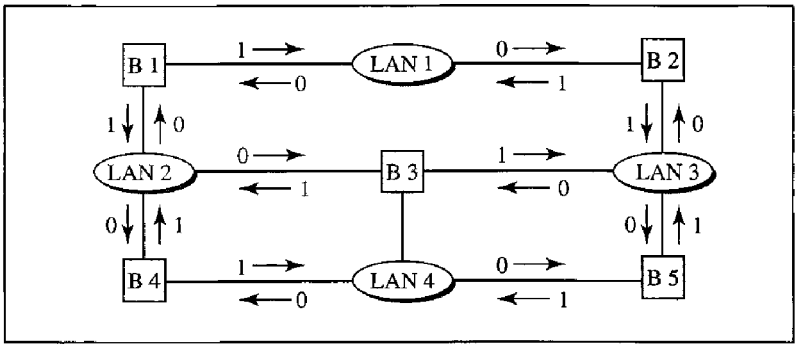
The process to find the spanning tree involves three steps:

1. Every bridge has a built-in ID (normally the serial number, which is unique). Each bridge broadcasts this ID so that all bridges know which one has the smallest ID. The bridge with the smallest ID is selected as the *root* bridge (root of the tree). We assume that bridge B1 has the smallest ID. It is, therefore, selected as the root bridge.

Figure 15.8 *A system of connected LANs and its graph representation*



a. Actual system



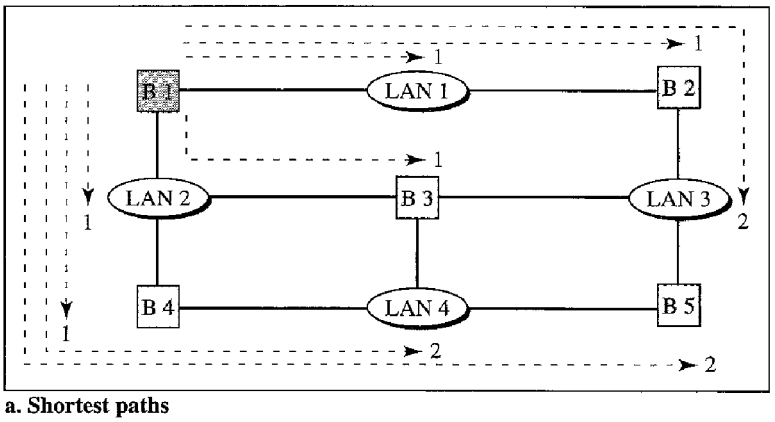
b. Graph representation with cost assigned to each arc

2. The algorithm tries to find the shortest path (a path with the shortest cost) from the root bridge to every other bridge or LAN. The shortest path can be found by examining the total cost from the root bridge to the destination. Figure 15.9 shows the shortest paths.
3. The combination of the shortest paths creates the shortest tree, which is also shown in Figure 15.9.
4. Based on the spanning tree, we mark the ports that are part of the spanning tree, the **forwarding ports,** which forward a frame that the bridge receives. We also mark those ports that are not part of the spanning tree, the **blocking ports,** which block the frames received by the bridge. Figure 15.10 shows the physical systems of LANs with forwarding points (solid lines) and blocking ports (broken lines).

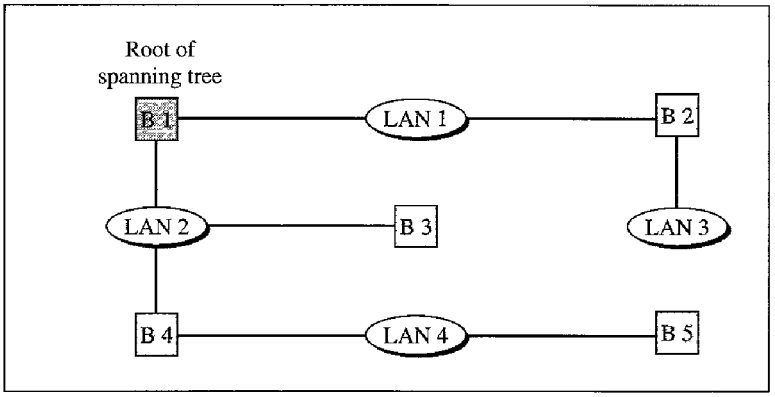
Note that there is only one single path from any LAN to any other LAN in the spanning tree system. This means there is only one single path from one LAN to any other LAN. No loops are created. You can prove to yourself that there is only one path from LAN 1 to LAN 2, LAN 3, or LAN 4. Similarly, there is only one path from LAN 2 to LAN 1, LAN 3, and LAN 4. The same is true for LAN 3 and LAN 4.

Dynamic Algorithm We have described the spanning tree algorithm as though it required manual entries. This is not true. Each bridge is equipped with a software package that carries out this process dynamically. The bridges send special messages to one another, called bridge protocol data units (BPDUs), to update the spanning tree. The spanning tree is updated when there is a change in the system such as a failure of a bridge or an addition or deletion of bridges.

Figure 15.9 *Finding the shortest paths and the spanning tree in a system of bridges*

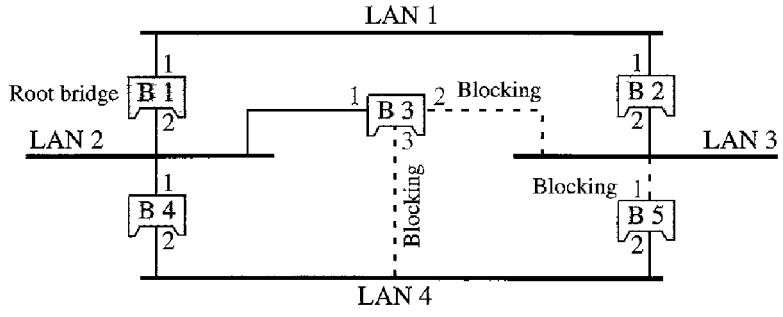


a. Shortest paths



b. Spanning tree

Figure 15.10 *Forwarding and blocking ports after using spanning tree algorithm*



Ports 2 and 3 of bridge B3 are blocking ports (no frame is sent out of these ports).
Port 1 of bridge B5 is also a blocking port (no frame is sent out of this port).

Source Routing Bridges

Another way to prevent loops in a system with redundant bridges is to use **source routing bridges.** A transparent bridge's duties include filtering frames, forwarding, and blocking. In a system that has source routing bridges, these duties are performed by the source station and, to some extent, the destination station.

In source routing, a sending station defines the bridges that the frame must visit. The addresses of these bridges are included in the frame. In other words, the frame contains not only the source and destination addresses, but also the addresses of all bridges to be visited.

The source gets these bridge addresses through the exchange of special frames with the destination prior to sending the data frame.

Source routing bridges were designed by IEEE to be used with Token Ring LANs. These LANs are not very common today.

Bridges Connecting Different LANs

Theoretically a bridge should be able to connect LANs using different protocols at the data link layer, such as an Ethernet LAN to a wireless LAN. However, there are many issues to be considered:

- **Frame format.** Each LAN type has its own frame format (compare an Ethernet frame with a wireless LAN frame).
- **Maximum data size.** If an incoming frame's size is too large for the destination LAN, the data must be fragmented into several frames. The data then need to be reassembled at the destination. However, no protocol at the data link layer allows the fragmentation and reassembly of frames. We will see in Chapter 19 that this is allowed in the network layer. The bridge must therefore discard any frames too large for its system.
- **Data rate.** Each LAN type has its own data rate. (Compare the 10-Mbps data rate of an Ethernet with the 1-Mbps data rate of a wireless LAN.) The bridge must buffer the frame to compensate for this difference.
- **Bit order.** Each LAN type has its own strategy in the sending of bits. Some send the most significant bit in a byte first; others send the least significant bit first.
- **Security.** Some LANs, such as wireless LANs, implement security measures in the data link layer. Other LANs, such as Ethernet, do not. Security often involves encryption (see Chapter 30). When a bridge receives a frame from a wireless LAN, it needs to decrypt the message before forwarding it to an Ethernet LAN.
- **Multimedia support.** Some LANs support multimedia and the quality of services needed for this type of communication; others do not.

Two-Layer Switches

When we use the term *switch,* we must be careful because a switch can mean two different things. We must clarify the term by adding the level at which the device operates. We can have a two-layer switch or a three-layer switch. A **three-layer switch** is used at the network layer; it is a kind of router. The **two-layer switch** performs at the physical and data link layers.

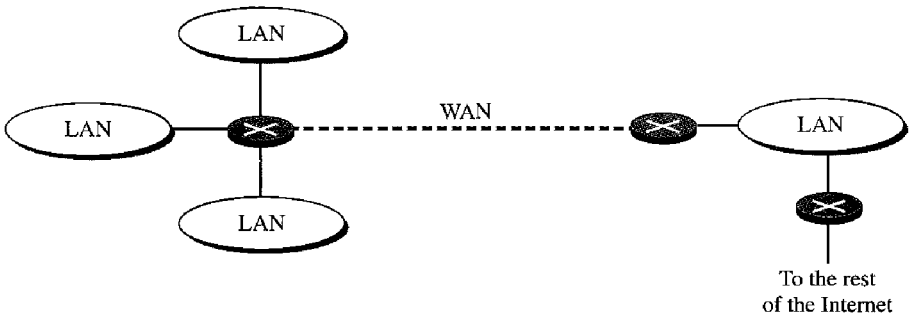
A two-layer switch is a bridge, a bridge with many ports and a design that allows better (faster) performance. A bridge with a few ports can connect a few LANs together. A bridge with many ports may be able to allocate a unique port to each station, with each station on its own independent entity. This means no competing traffic (no collision, as we saw in Ethernet).

A two-layer switch, as a bridge does, makes a filtering decision based on the MAC address of the frame it received. However, a two-layer switch can be more sophisticated. It can have a buffer to hold the frames for processing. It can have a switching factor that forwards the frames faster. Some new two-layer switches, called *cut-through* switches, have been designed to forward the frame as soon as they check the MAC addresses in the header of the frame.

Routers

A **router** is a three-layer device that routes packets based on their logical addresses (host-to-host addressing). A router normally connects LANs and WANs in the Internet and has a routing table that is used for making decisions about the route. The routing tables are normally dynamic and are updated using routing protocols. We discuss routers and routing in greater detail in Chapters 19 and 21. Figure 15.11 shows a part of the Internet that uses routers to connect LANs and WANs.

Figure 15.11 *Routers connecting independent LANs and WANs*



Three-Layer Switches

A three-layer switch is a router, but a faster and more sophisticated. The switching fabric in a three-layer switch allows faster table lookup and forwarding. In this book, we use the terms *router* and *three-layer switch* interchangeably.

Gateway

Although some textbooks use the terms *gateway* and *router* interchangeably, most of the literature distinguishes between the two. A gateway is normally a computer that operates in all five layers of the Internet or seven layers of OSI model. A gateway takes an application message, reads it, and interprets it. This means that it can be used as a connecting device between two internetworks that use different models. For example, a network designed to use the OSI model can be connected to another network using the Internet model. The gateway connecting the two systems can take a frame as it arrives from the first system, move it up to the OSI application layer, and remove the message.

Gateways can provide security. In Chapter 32, we learn that the gateway is used to filter unwanted application-layer messages.

15.2 BACKBONE NETWORKS

Some connecting devices discussed in this chapter can be used to connect LANs in a backbone network. A backbone network allows several LANs to be connected. In a backbone network, no station is directly connected to the backbone; the stations are part of a LAN, and the backbone connects the LANs. The backbone is itself a LAN that uses a LAN protocol such as Ethernet; each connection to the backbone is itself another LAN.

Although many different architectures can be used for a backbone, we discuss only the two most common: the bus and the star.

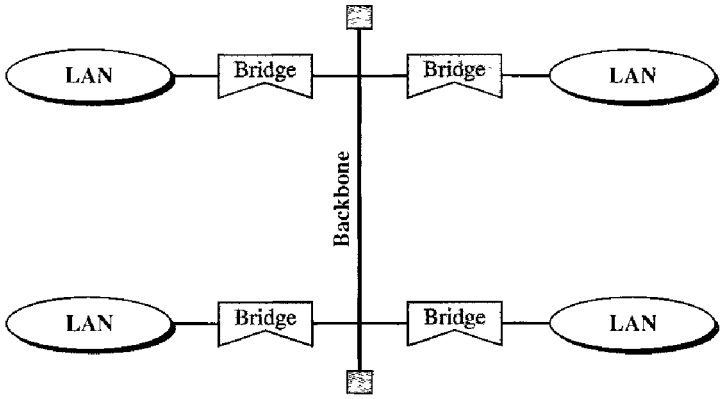
Bus Backbone

In a **bus backbone,** the topology of the backbone is a bus. The backbone itself can use one of the protocols that support a bus topology such as 10Base5 or 10Base2.

In a bus backbone, the topology of the backbone is a bus.

Bus backbones are normally used as a distribution backbone to connect different buildings in an organization. Each building can comprise either a single LAN or another backbone (normally a star backbone). A good example of a bus backbone is one that connects single- or multiple-floor buildings on a campus. Each single-floor building usually has a single LAN. Each multiple-floor building has a backbone (usually a star) that connects each LAN on a floor. A bus backbone can interconnect these LANs and backbones. Figure 15.12 shows an example of a bridge-based backbone with four LANs.

Figure 15.12 *Bus backbone*



In Figure 15.12, if a station in a LAN needs to send a frame to another station in the same LAN, the corresponding bridge blocks the frame; the frame never reaches the backbone. However, if a station needs to send a frame to a station in another LAN, the bridge passes the frame to the backbone, which is received by the appropriate bridge and is delivered to the destination LAN. Each bridge connected to the backbone has a table that shows the stations on the LAN side of the bridge. The blocking or delivery of a frame is based on the contents of this table.

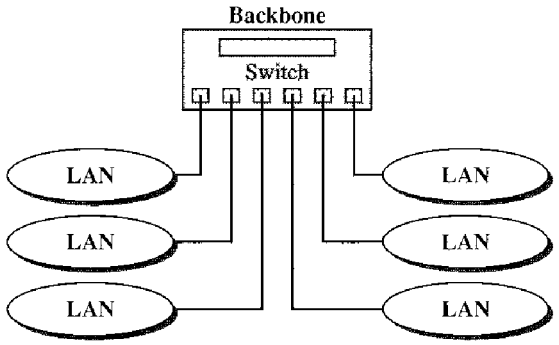
Star Backbone

In a **star backbone,** sometimes called a collapsed or switched backbone, the topology of the backbone is a star. In this configuration, the backbone is just one switch (that is why it is called, erroneously, a collapsed backbone) that connects the LANs.

In a star backbone, the topology of the backbone is a star; the backbone is just one switch.

Figure 15.13 shows a star backbone. Note that, in this configuration, the switch does the job of the backbone and at the same time connects the LANs.

Figure 15.13 *Star backbone*



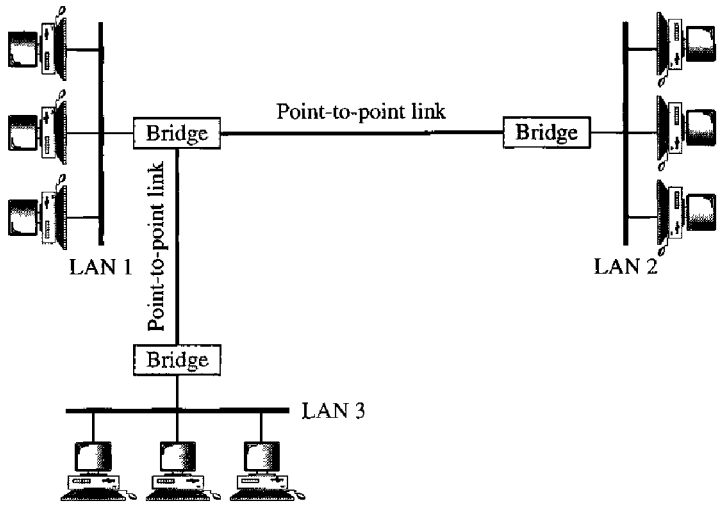
Star backbones are mostly used as a distribution backbone inside a building. In a multifloor building, we usually find one LAN that serves each particular floor. A star backbone connects these LANs. The backbone network, which is just a switch, can be installed in the basement or the first floor, and separate cables can run from the switch to each LAN. If the individual LANs have a physical star topology, either the hubs (or switches) can be installed in a closet on the corresponding floor, or all can be installed close to the switch. We often find a rack or chassis in the basement where the backbone switch and all hubs or switches are installed.

Connecting Remote LANs

Another common application for a backbone network is to connect remote LANs. This type of backbone network is useful when a company has several offices with LANs and needs to connect them. The connection can be done through bridges,

sometimes called **remote bridges.** The bridges act as connecting devices connecting LANs and point-to-point networks, such as leased telephone lines or ADSL lines. The point-to-point network in this case is considered a LAN without stations. The point-to-point link can use a protocol such as PPP. Figure 15.14 shows a backbone connecting remote LANs.

Figure 15.14 *Connecting remote LANs with bridges*



A point-to-point link acts as a LAN in a remote backbone connected by remote bridges.

15.3 VIRTUAL LANs

A station is considered part of a LAN if it physically belongs to that LAN. The criterion of membership is geographic. What happens if we need a virtual connection between two stations belonging to two different physical LANs? We can roughly define a **virtual local area network (VLAN)** as a local area network configured by software, not by physical wiring.

Let us use an example to elaborate on this definition. Figure 15.15 shows a switched LAN in an engineering firm in which 10 stations are grouped into three LANs that are connected by a switch. The first four engineers work together as the first group, the next three engineers work together as the second group, and the last three engineers work together as the third group. The LAN is configured to allow this arrangement.

But what would happen if the administrators needed to move two engineers from the first group to the third group, to speed up the project being done by the third group? The LAN configuration would need to be changed. The network technician must rewire. The problem is repeated if, in another week, the two engineers move back to their previous group. In a switched LAN, changes in the work group mean physical changes in the network configuration.

Figure 15.15 *A switch connecting three LANs*

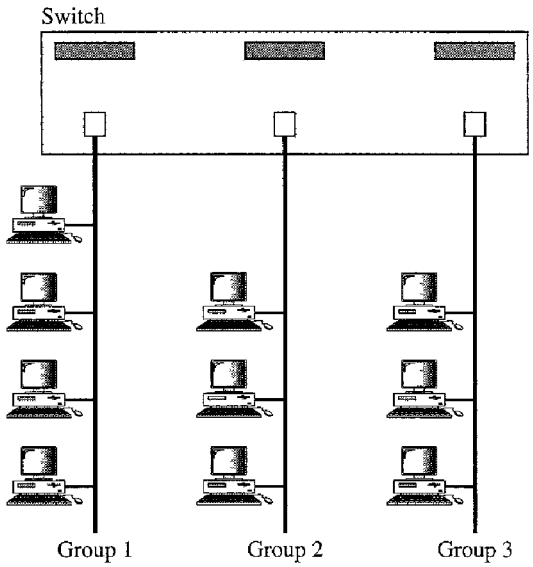
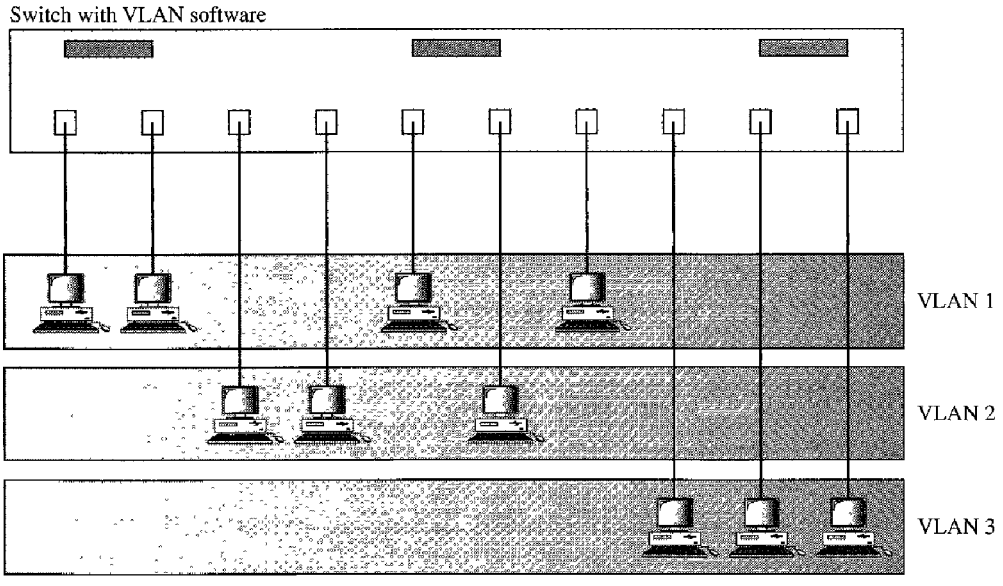


Figure 15.16 shows the same switched LAN divided into VLANs. The whole idea of VLAN technology is to divide a LAN into logical, instead of physical, segments. A LAN can be divided into several logical LANs called VLANs. Each VLAN is a work group in the organization. If a person moves from one group to another, there is no need to change the physical configuration. The group membership in VLANs is defined by software, not hardware. Any station can be logically moved to another VLAN. All members belonging to a VLAN can receive broadcast messages sent to that particular VLAN.

Figure 15.16 *A switch using VLAN software*

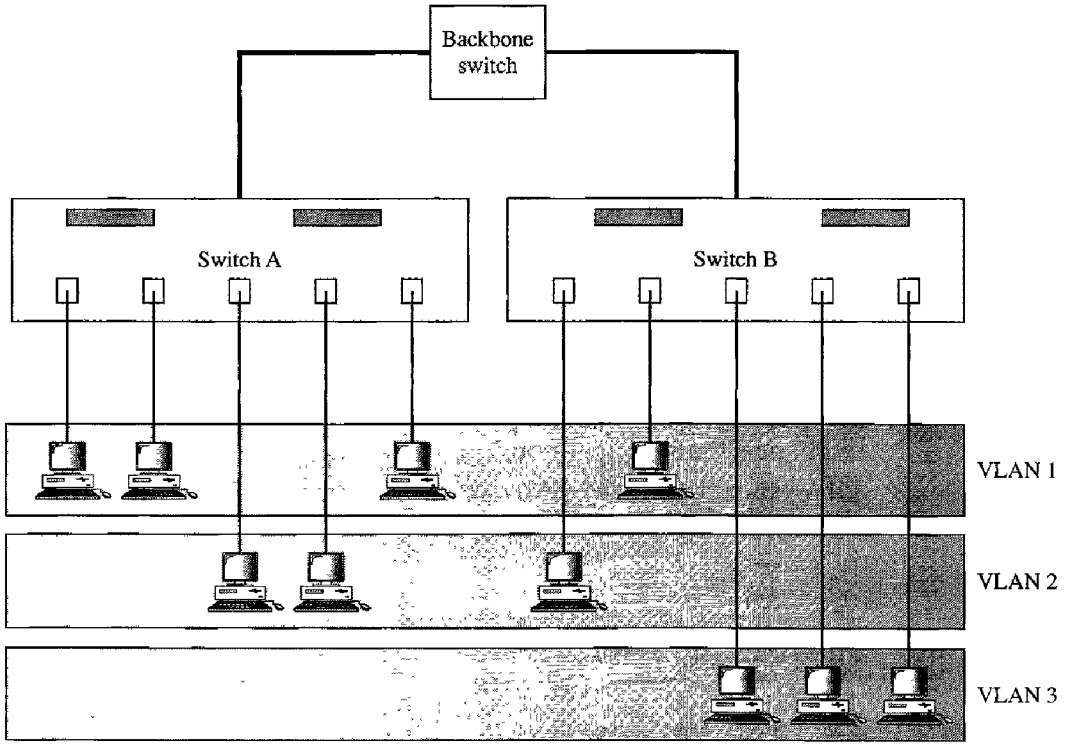


This means if a station moves from VLAN 1 to VLAN 2, it receives broadcast messages sent to VLAN 2, but no longer receives broadcast messages sent to VLAN 1.

It is obvious that the problem in our previous example can easily be solved by using VLANs. Moving engineers from one group to another through software is easier than changing the configuration of the physical network.

VLAN technology even allows the grouping of stations connected to different switches in a VLAN. Figure 15.17 shows a backbone local area network with two switches and three VLANs. Stations from switches A and B belong to each VLAN.

Figure 15.17 *Two switches in a backbone using VLAN software*



This is a good configuration for a company with two separate buildings. Each building can have its own switched LAN connected by a backbone. People in the first building and people in the second building can be in the same work group even though they are connected to different physical LANs.

From these three examples, we can define a VLAN characteristic:

VLANs create broadcast domains.

VLANs group stations belonging to one or more physical LANs into broadcast domains. The stations in a VLAN communicate with one another as though they belonged to a physical segment.

Membership

What characteristic can be used to group stations in a VLAN? Vendors use different characteristics such as port numbers, MAC addresses, IP addresses, IP multicast addresses, or a combination of two or more of these.

Port Numbers

Some VLAN vendors use switch port numbers as a membership characteristic. For example, the administrator can define that stations connecting to ports 1, 2, 3, and 7 belong to VLAN 1; stations connecting to ports 4, 10, and 12 belong to VLAN 2; and so on.

MAC Addresses

Some VLAN vendors use the 48-bit MAC address as a membership characteristic. For example, the administrator can stipulate that stations having MAC addresses E21342A12334 and F2A123BCD341 belong to VLAN 1.

IP Addresses

Some VLAN vendors use the 32-bit IP address (see Chapter 19) as a membership characteristic. For example, the administrator can stipulate that stations having IP addresses 181.34.23.67, 181.34.23.72, 181.34.23.98, and 181.34.23.112 belong to VLAN 1.

Multicast IP Addresses

Some VLAN vendors use the multicast IP address (see Chapter 19) as a membership characteristic. Multicasting at the IP layer is now translated to multicasting at the data link layer.

Combination

Recently, the software available from some vendors allows all these characteristics to be combined. The administrator can choose one or more characteristics when installing the software. In addition, the software can be reconfigured to change the settings.

Configuration

How are the stations grouped into different VLANs? Stations are configured in one of three ways: manual, semiautomatic, and automatic.

Manual Configuration

In a manual configuration, the network administrator uses the VLAN software to manually assign the stations into different VLANs at setup. Later migration from one VLAN to another is also done manually. Note that this is not a physical configuration; it is a logical configuration. The term *manually* here means that the administrator types the port numbers, the IP addresses, or other characteristics, using the VLAN software.

Automatic Configuration

In an automatic configuration, the stations are automatically connected or disconnected from a VLAN using criteria defined by the administrator. For example, the administrator can define the project number as the criterion for being a member of a group. When a user changes the project, he or she automatically migrates to a new VLAN.

Semiautomatic Configuration

A semiautomatic configuration is somewhere between a manual configuration and an automatic configuration. Usually, the initializing is done manually, with migrations done automatically.

Communication Between Switches

In a multiswitched backbone, each switch must know not only which station belongs to which VLAN, but also the membership of stations connected to other switches. For example, in Figure 15.17, switch A must know the membership status of stations connected to switch B, and switch B must know the same about switch A. Three methods have been devised for this purpose: table maintenance, frame tagging, and time-division multiplexing.

Table Maintenance

In this method, when a station sends a broadcast frame to its group members, the switch creates an entry in a table and records station membership. The switches send their tables to one another periodically for updating.

Frame Tagging

In this method, when a frame is traveling between switches, an extra header is added to the MAC frame to define the destination VLAN. The frame tag is used by the receiving switches to determine the VLANs to be receiving the broadcast message.

Time-Division Multiplexing (TDM)

In this method, the connection (trunk) between switches is divided into timeshared channels (see TDM in Chapter 6). For example, if the total number of VLANs in a backbone is five, each trunk is divided into five channels. The traffic destined for VLAN 1 travels in channel 1, the traffic destined for VLAN 2 travels in channel 2, and so on. The receiving switch determines the destination VLAN by checking the channel from which the frame arrived.

IEEE Standard

In 1996, the IEEE 802.1 subcommittee passed a standard called 802.1Q that defines the format for frame tagging. The standard also defines the format to be used in multiswitched backbones and enables the use of multivendor equipment in VLANs. IEEE 802.1Q has opened the way for further standardization in other issues related to VLANs. Most vendors have already accepted the standard.

Advantages

There are several advantages to using VLANs.

Cost and Time Reduction

VLANs can reduce the migration cost of stations going from one group to another. Physical reconfiguration takes time and is costly. Instead of physically moving one station to another segment or even to another switch, it is much easier and quicker to move it by using software.

Creating Virtual Work Groups

VLANs can be used to create virtual work groups. For example, in a campus environment, professors working on the same project can send broadcast messages to one another without the necessity of belonging to the same department. This can reduce traffic if the multicasting capability of IP was previously used.

Security

VLANs provide an extra measure of security. People belonging to the same group can send broadcast messages with the guaranteed assurance that users in other groups will not receive these messages.

15.4 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

A book devoted to connecting devices is [Per00]. Connecting devices and VLANs are discussed in Section 4.7 of [Tan03]. Switches, bridges, and hubs are discussed in [Sta03] and [Sta04].

Site

- IEEE 802 LAN/MAN Standards Committee

15.5 KEY TERMS

amplifier
blocking port
bridge
bus backbone
connecting device
filtering
forwarding port
hub
remote bridge
repeater
router
segment

source routing bridge
spanning tree
star backbone
three-layer switch
transparent bridge
two-layer switch
virtual local area network (VLAN)

15.6 SUMMARY

- A repeater is a connecting device that operates in the physical layer of the Internet model. A repeater regenerates a signal, connects segments of a LAN, and has no filtering capability.
- A bridge is a connecting device that operates in the physical and data link layers of the Internet model.
- A transparent bridge can forward and filter frames and automatically build its forwarding table.
- A bridge can use the spanning tree algorithm to create a loopless topology.
- A backbone LAN allows several LANs to be connected.
- A backbone is usually a bus or a star.
- A virtual local area network (VLAN) is configured by software, not by physical wiring.
- Membership in a VLAN can be based on port numbers, MAC addresses, IP addresses, IP multicast addresses, or a combination of these features.
- VLANs are cost- and time-efficient, can reduce network traffic, and provide an extra measure of security.

15.7 PRACTICE SET

Review Questions

1. How is a repeater different from an amplifier?
2. What do we mean when we say that a bridge can filter traffic? Why is filtering important?
3. What is a transparent bridge?
4. How does a repeater extend the length of a LAN?
5. How is a hub related to a repeater?
6. What is the difference between a forwarding port and a blocking port?
7. What is the difference between a bus backbone and a star backbone?
8. How does a VLAN save a company time and money?
9. How does a VLAN provide extra security for a network?
10. How does a VLAN reduce network traffic?
11. What is the basis for membership in a VLAN?

Exercises

12. Complete the table in Figure 15.6 after each station has sent a packet to another station.
13. Find the spanning tree for the system in Figure 15.7.
14. Find the spanning tree for the system in Figure 15.8 if bridge B5 is removed.
15. Find the spanning tree for the system in Figure 15.8 if bridge B2 is removed.
16. Find the spanning tree for the system in Figure 15.8 if B5 is selected as the root bridge.
17. In Figure 15.6, we are using a bridge. Can we replace the bridge with a router? Explain the consequences.
18. A bridge uses a filtering table; a router uses a routing table. Can you explain the difference?
19. Create a system of three LANs with four bridges. The bridges (B1 to B4) connect the LANs as follows:
    a. B1 connects LAN 1 and LAN 2.
    b. B2 connects LAN 1 and LAN 3.
    c. B3 connects LAN 2 and LAN 3.
    d. B4 connects LAN 1, LAN 2, and LAN 3.

    Choose B1 as the root bridge. Show the forwarding and blocking ports, after applying the spanning tree procedure.
20. Which one has more overhead, a bridge or a router? Explain your answer.
21. Which one has more overhead, a repeater or a bridge? Explain your answer.
22. Which one has more overhead, a router or a gateway? Explain your answer.

CHAPTER 16

Wireless WANs: Cellular Telephone and Satellite Networks

We discussed wireless LANs in Chapter 14. Wireless technology is also used in cellular telephony and satellite networks. We discuss the former in this chapter as well as examples of channelization access methods (see Chapter 12). We also briefly discuss satellite networks, a technology that eventually will be linked to cellular telephony to access the Internet directly.

16.1 CELLULAR TELEPHONY

Cellular telephony is designed to provide communications between two moving units, called mobile stations (MSs), or between one mobile unit and one stationary unit, often called a land unit. A service provider must be able to locate and track a caller, assign a channel to the call, and transfer the channel from base station to base station as the caller moves out of range.

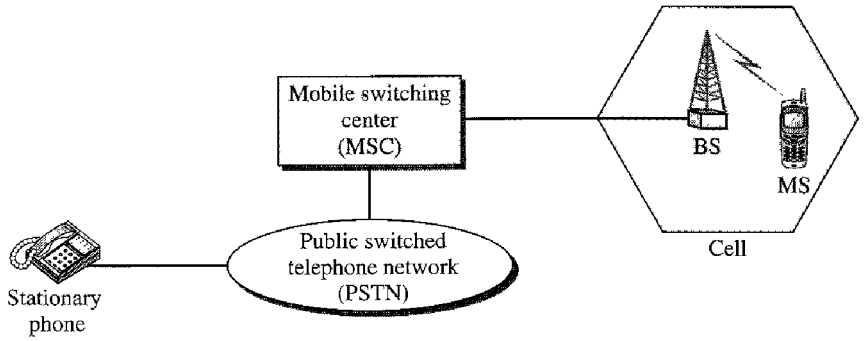
To make this tracking possible, each cellular service area is divided into small regions called cells. Each cell contains an antenna and is controlled by a solar or AC powered network station, called the base station (BS). Each base station, in turn, is controlled by a switching office, called a **mobile switching center (MSC).** The MSC coordinates communication between all the base stations and the telephone central office. It is a computerized center that is responsible for connecting calls, recording call information, and billing (see Figure 16.1).

Cell size is not fixed and can be increased or decreased depending on the population of the area. The typical radius of a cell is 1 to 12 mi. High-density areas require more, geographically smaller cells to meet traffic demands than do low-density areas. Once determined, cell size is optimized to prevent the interference of adjacent cell signals. The transmission power of each cell is kept low to prevent its signal from interfering with those of other cells.

Frequency-Reuse Principle

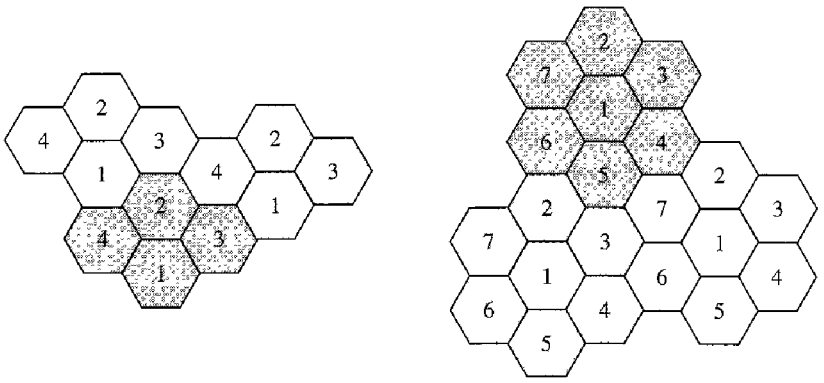
In general, neighboring cells cannot use the same set of frequencies for communication because it may create interference for the users located near the cell boundaries. However, the set of frequencies available is limited, and frequencies need to be reused. A

Figure 16.1 *Cellular system*



frequency reuse pattern is a configuration of N cells, N being the **reuse factor,** in which each cell uses a unique set of frequencies. When the pattern is repeated, the frequencies can be reused. There are several different patterns. Figure 16.2 shows two of them.

Figure 16.2 *Frequency reuse patterns*



a. Reuse factor of 4 **b. Reuse factor of 7**

The numbers in the cells define the pattern. The cells with the same number in a pattern can use the same set of frequencies. We call these cells the *reusing cells*. As Figure 16.2 shows, in a pattern with reuse factor 4, only one cell separates the cells using the same set of frequencies. In the pattern with reuse factor 7, two cells separate the reusing cells.

Transmitting

To place a call from a mobile station, the caller enters a code of 7 or 10 digits (a phone number) and presses the send button. The mobile station then scans the band, seeking a setup channel with a strong signal, and sends the data (phone number) to the closest base station using that channel. The base station relays the data to the MSC. The MSC

sends the data on to the telephone central office. If the called party is available, a connection is made and the result is relayed back to the MSC. At this point, the MSC assigns an unused voice channel to the call, and a connection is established. The mobile station automatically adjusts its tuning to the new channel, and communication can begin.

Receiving

When a mobile phone is called, the telephone central office sends the number to the MSC. The MSC searches for the location of the mobile station by sending query signals to each cell in a process called *paging*. Once the mobile station is found, the MSC transmits a ringing signal and, when the mobile station answers, assigns a voice channel to the call, allowing voice communication to begin.

Handoff

It may happen that, during a conversation, the mobile station moves from one cell to another. When it does, the signal may become weak. To solve this problem, the MSC monitors the level of the signal every few seconds. If the strength of the signal diminishes, the MSC seeks a new cell that can better accommodate the communication. The MSC then changes the channel carrying the call (hands the signal off from the old channel to a new one).

Hard Handoff Early systems used a hard **handoff.** In a hard handoff, a mobile station only communicates with one base station. When the MS moves from one cell to another, communication must first be broken with the previous base station before communication can be established with the new one. This may create a rough transition.

Soft Handoff New systems use a soft handoff. In this case, a mobile station can communicate with two base stations at the same time. This means that, during handoff, a mobile station may continue with the new base station before breaking off from the old one.

Roaming

One feature of cellular telephony is called **roaming.** Roaming means, in principle, that a user can have access to communication or can be reached where there is coverage. A service provider usually has limited coverage. Neighboring service providers can provide extended coverage through a roaming contract. The situation is similar to snail mail between countries. The charge for delivery of a letter between two countries can be divided upon agreement by the two countries.

First Generation

Cellular telephony is now in its second generation with the third on the horizon. The first generation was designed for voice communication using analog signals. We discuss one first-generation mobile system used in North America, AMPS.

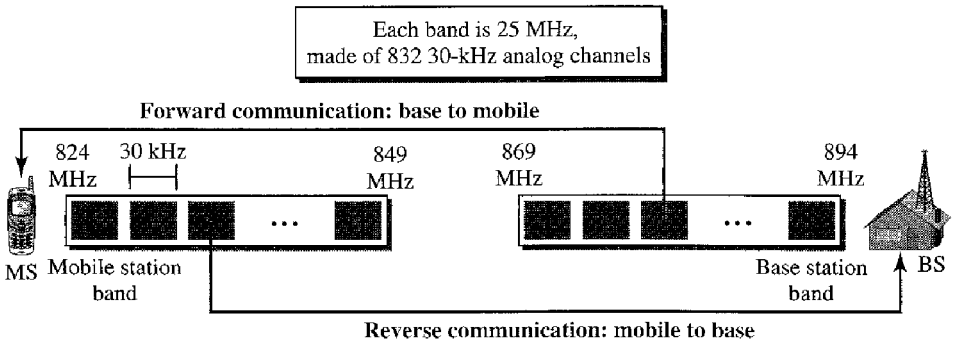
AMPS

Advanced Mobile Phone System (AMPS) is one of the leading analog cellular systems in North America. It uses FDMA (see Chapter 12) to separate channels in a link.

AMPS is an analog cellular phone system using FDMA.

Bands AMPS operates in the ISM 800-MHz band. The system uses two separate analog channels, one for forward (base station to mobile station) communication and one for reverse (mobile station to base station) communication. The band between 824 and 849 MHz carries reverse communication; the band between 869 and 894 MHz carries forward communication (see Figure 16.3).

Figure 16.3 *Cellular bands for AMPS*



Each band is divided into 832 channels. However, two providers can share an area, which means 416 channels in each cell for each provider. Out of these 416, 21 channels are used for control, which leaves 395 channels. AMPS has a frequency reuse factor of 7; this means only one-seventh of these 395 traffic channels are actually available in a cell.

Transmission AMPS uses FM and FSK for modulation. Figure 16.4 shows the transmission in the reverse direction. Voice channels are modulated using FM, and control channels use FSK to create 30-kHz analog signals. AMPS uses FDMA to divide each 25-MHz band into 30-kHz channels.

Second Generation

To provide higher-quality (less noise-prone) mobile voice communications, the second generation of the cellular phone network was developed. While the first generation was designed for analog voice communication, the second generation was mainly designed for digitized voice. Three major systems evolved in the second generation, as shown in Figure 16.5. We will discuss each system separately.

Figure 16.4 *AMPS reverse communication band*

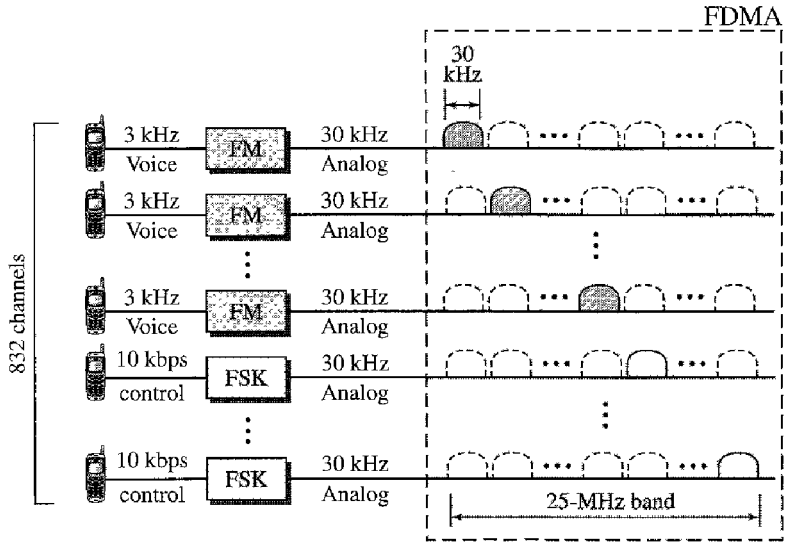
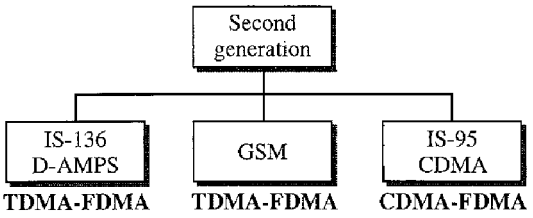


Figure 16.5 *Second-generation cellular phone systems*



D-AMPS

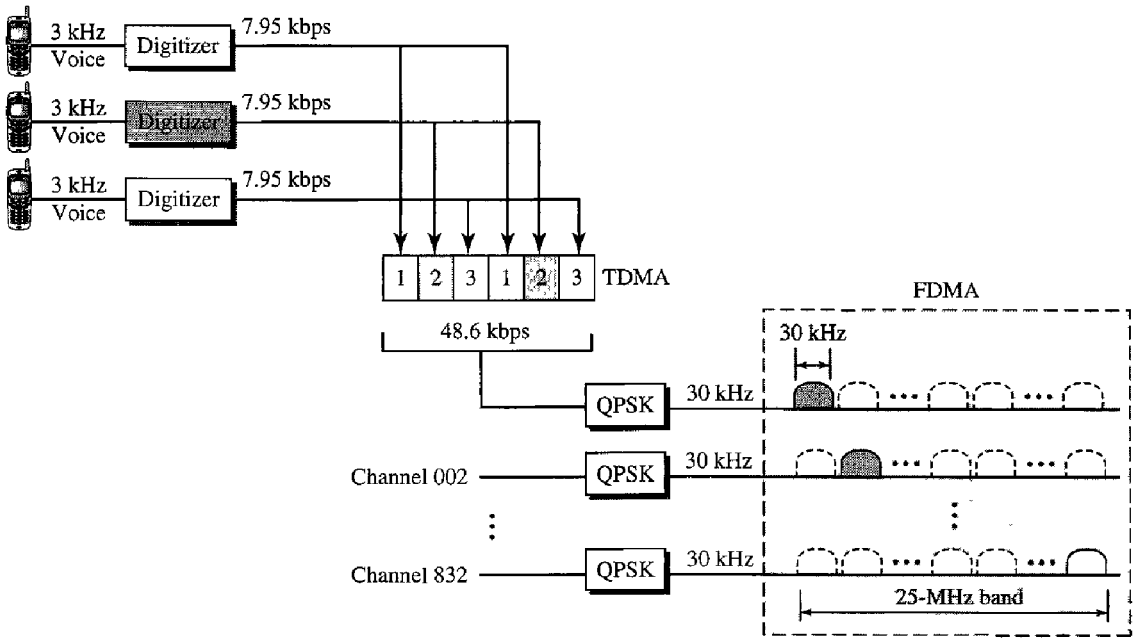
The product of the evolution of the analog AMPS into a digital system is **digital AMPS (D-AMPS).** D-AMPS was designed to be backward-compatible with AMPS. This means that in a cell, one telephone can use AMPS and another D-AMPS. D-AMPS was first defined by IS-54 (Interim Standard 54) and later revised by IS-136.

Band D-AMPS uses the same bands and channels as AMPS.

Transmission Each voice channel is digitized using a very complex PCM and compression technique. A voice channel is digitized to 7.95 kbps. Three 7.95-kbps digital voice channels are combined using TDMA. The result is 48.6 kbps of digital data; much of this is overhead. As Figure 16.6 shows, the system sends 25 frames per second, with 1944 bits per frame. Each frame lasts 40 ms (1/25) and is divided into six slots shared by three digital channels; each channel is allotted two slots.

Each slot holds 324 bits. However, only 159 bits comes from the digitized voice; 64 bits are for control and 101 bits are for error correction. In other words, each channel drops 159 bits of data into each of the two channels assigned to it. The system adds 64 control bits and 101 error-correcting bits.

Figure 16.6 *D-AMPS*



The resulting 48.6 kbps of digital data modulates a carrier using QPSK; the result is a 30-kHz analog signal. Finally, the 30-kHz analog signals share a 25-MHz band (FDMA). D-AMPS has a frequency reuse factor of 7.

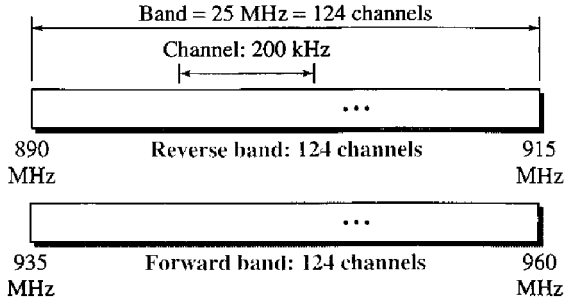
D-AMPS, or IS-136, is a digital cellular phone system using TDMA and FDMA.

GSM

The **Global System for Mobile Communication (GSM)** is a European standard that was developed to provide a common second-generation technology for all Europe. The aim was to replace a number of incompatible first-generation technologies.

Bands GSM uses two bands for duplex communication. Each band is 25 MHz in width, shifted toward 900 MHz, as shown in Figure 16.7. Each band is divided into 124 channels of 200 kHz separated by guard bands.

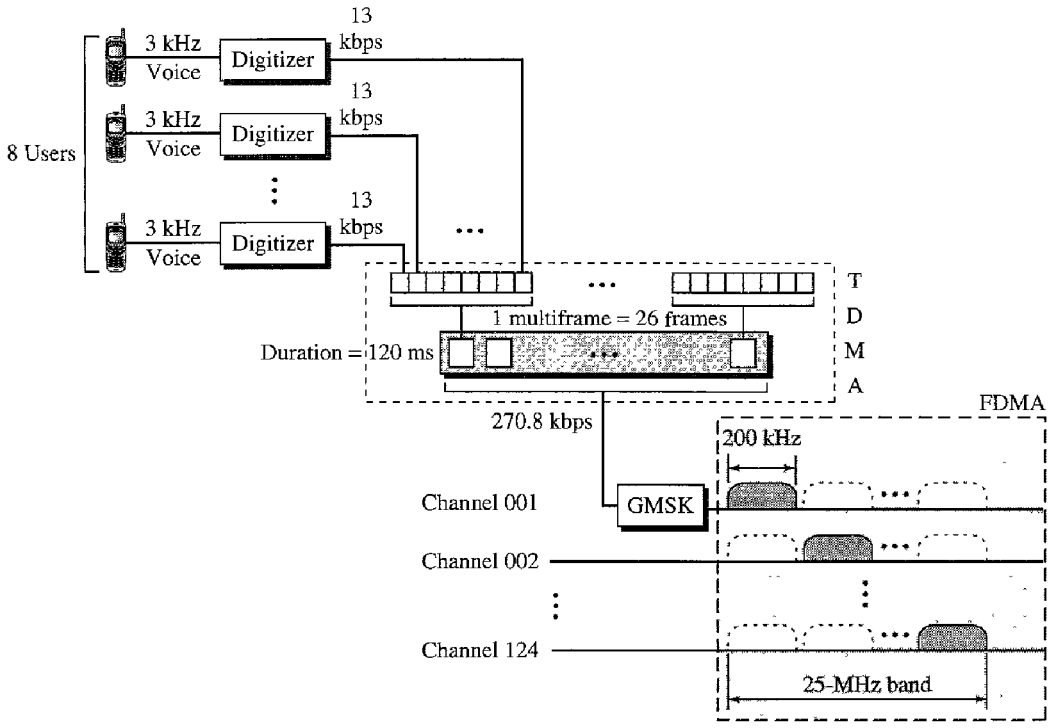
Figure 16.7 *GSM bands*



Transmission Figure 16.8 shows a GSM system. Each voice channel is digitized and compressed to a 13-kbps digital signal. Each slot carries 156.25 bits (see Figure 16.9). Eight slots share a frame (TDMA). Twenty-six frames also share a multiframe (TDMA). We can calculate the bit rate of each channel as follows:

$$\text{Channel data rate} = (1/120 \text{ ms}) \times 26 \times 8 \times 156.25 = 270.8 \text{ kbps}$$

Figure 16.8 *GSM*

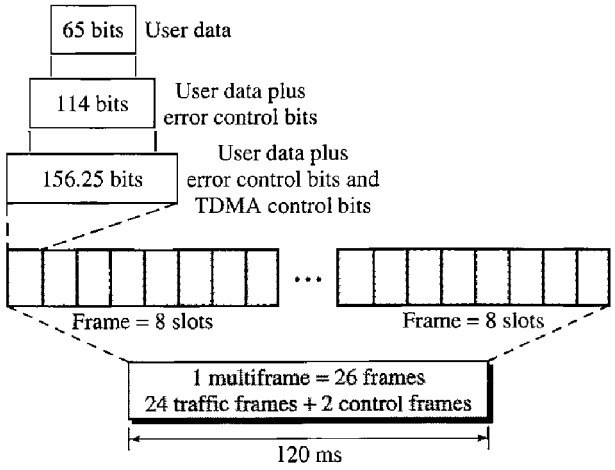


Each 270.8-kbps digital channel modulates a carrier using GMSK (a form of FSK used mainly in European systems); the result is a 200-kHz analog signal. Finally 124 analog channels of 200 kHz are combined using FDMA. The result is a 25-MHz band. Figure 16.9 shows the user data and overhead in a multiframe.

The reader may have noticed the large amount of overhead in TDMA. The user data are only 65 bits per slot. The system adds extra bits for error correction to make it 114 bits per slot. To this, control bits are added to bring it up to 156.25 bits per slot. Eight slots are encapsulated in a frame. Twenty-four traffic frames and two additional control frames make a multiframe. A multiframe has a duration of 120 ms. However, the architecture does define superframes and hyperframes that do not add any overhead; we will not discuss them here.

Reuse Factor Because of the complex error correction mechanism, GSM allows a reuse factor as low as 3.

Figure 16.9 *Multiframe components*



GSM is a digital cellular phone system using TDMA and FDMA.

IS-95

One of the dominant second-generation standards in North America is **Interim Standard 95 (IS-95).** It is based on CDMA and DSSS.

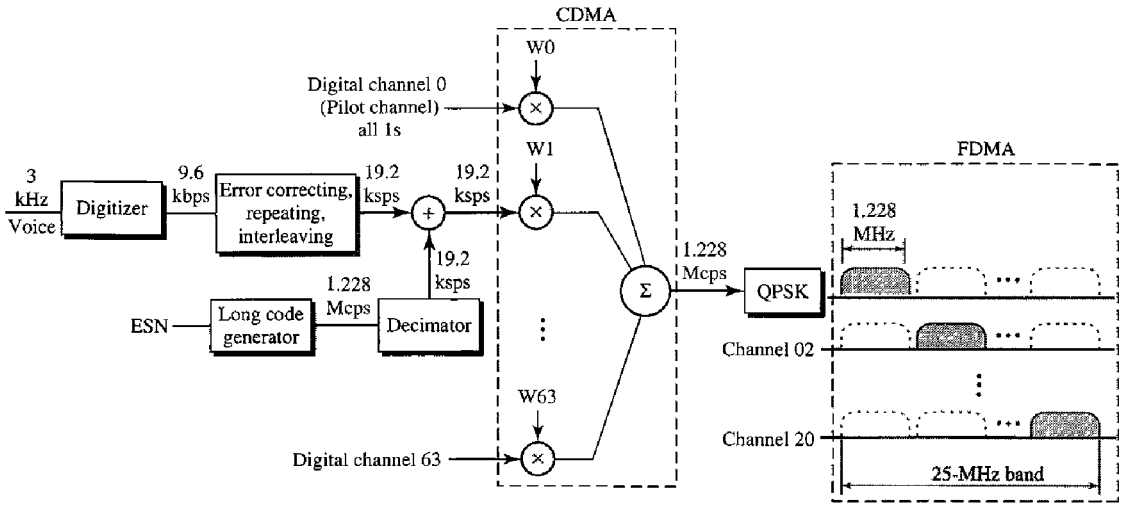
Bands and Channels IS-95 uses two bands for duplex communication. The bands can be the traditional ISM 800-MHz band or the ISM 1900-MHz band. Each band is divided into 20 channels of 1.228 MHz separated by guard bands. Each service provider is allotted 10 channels. IS-95 can be used in parallel with AMPS. Each IS-95 channel is equivalent to 41 AMPS channels (41×30 kHz = 1.23 MHz).

Synchronization All base channels need to be synchronized to use CDMA. To provide synchronization, bases use the services of GPS (Global Positioning System), a satellite system that we discuss in the next section.

Forward Transmission IS-95 has two different transmission techniques: one for use in the forward (base to mobile) direction and another for use in the reverse (mobile to base) direction. In the forward direction, communications between the base and all mobiles are synchronized; the base sends synchronized data to all mobiles. Figure 16.10 shows a simplified diagram for the forward direction.

Each voice channel is digitized, producing data at a basic rate of 9.6 kbps. After adding error-correcting and repeating bits, and interleaving, the result is a signal of 19.2 ksps (kilosignals per second). This output is now scrambled using a 19.2-ksps signal. The scrambling signal is produced from a long code generator that uses the electronic serial number (ESN) of the mobile station and generates 2^{42} pseudorandom chips, each chip having 42 bits. Note that the chips are generated pseudorandomly, not randomly, because the pattern repeats itself. The output of the long code generator is fed to a decimator, which chooses 1 bit out of 64 bits. The output of the decimator is used for scrambling. The scrambling is used to create privacy; the ESN is unique for each station.

Figure 16.10 *IS-95 forward transmission*



The result of the scrambler is combined using CDMA. For each traffic channel, one Walsh 64 × 64 row chip is selected. The result is a signal of 1.228 Mcps (megachips per second).

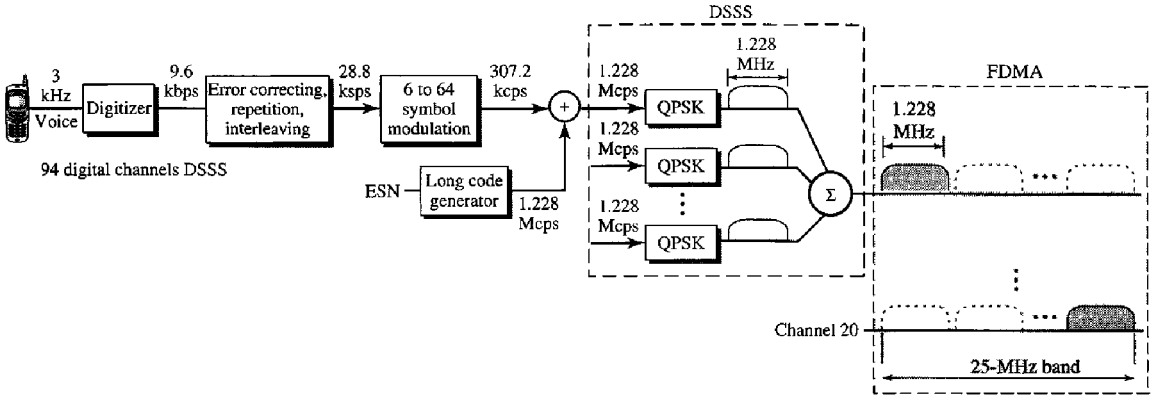
$$19.2 \text{ ksps} \times 64 \text{ cps} = 1.228 \text{ Mcps}$$

The signal is fed into a QPSK modulator to produce a signal of 1.228 MHz. The resulting bandwidth is shifted appropriately, using FDMA. An analog channel creates 64 digital channels, of which 55 channels are traffic channels (carrying digitized voice). Nine channels are used for control and synchronization:

- Channel 0 is a pilot channel. This channel sends a continuous stream of 1s to mobile stations. The stream provides bit synchronization, serves as a phase reference for demodulation, and allows the mobile station to compare the signal strength of neighboring bases for handoff decisions.
- Channel 32 gives information about the system to the mobile station.
- Channels 1 to 7 are used for paging, to send messages to one or more mobile stations.
- Channels 8 to 31 and 33 to 63 are traffic channels carrying digitized voice from the base station to the corresponding mobile station.

Reverse Transmission The use of CDMA in the forward direction is possible because the pilot channel sends a continuous sequence of 1s to synchronize transmission. The synchronization is not used in the reverse direction because we need an entity to do that, which is not feasible. Instead of CDMA, the reverse channels use DSSS (direct sequence spread spectrum), which we discussed in Chapter 8. Figure 16.11 shows a simplified diagram for reverse transmission.

Figure 16.11 *IS-95 reverse transmission*



Each voice channel is digitized, producing data at a rate of 9.6 kbps. However, after adding error-correcting and repeating bits, plus interleaving, the result is a signal of 28.8 ksps. The output is now passed through a 6/64 symbol modulator. The symbols are divided into six-symbol chunks, and each chunk is interpreted as a binary number (from 0 to 63). The binary number is used as the index to a 64 × 64 Walsh matrix for selection of a row of chips. Note that this procedure is not CDMA; each bit is not multiplied by the chips in a row. Each six-symbol chunk is replaced by a 64-chip code. This is done to provide a kind of orthogonality; it differentiates the streams of chips from the different mobile stations. The result creates a signal of 307.2 kcps or (28.8/6) × 64.

Spreading is the next step; each chip is spread into 4. Again the ESN of the mobile station creates a long code of 42 bits at a rate of 1.228 Mcps, which is 4 times 307.2. After spreading, each signal is modulated using QPSK, which is slightly different from the one used in the forward direction; we do not go into details here. Note that there is no multiple-access mechanism here; all reverse channels send their analog signal into the air, but the correct chips will be received by the base station due to spreading.

Although we can create $2^{42} - 1$ digital channels in the reverse direction (because of the long code generator), normally 94 channels are used; 62 are traffic channels, and 32 are channels used to gain access to the base station.

IS-95 is a digital cellular phone system using CDMA/DSSS and FDMA.

Two Data Rate Sets IS-95 defines two data rate sets, with four different rates in each set. The first set defines 9600, 4800, 2400, and 1200 bps. If, for example, the selected rate is 1200 bps, each bit is repeated 8 times to provide a rate of 9600 bps. The second set defines 14,400, 7200, 3600, and 1800 bps. This is possible by reducing the number of bits used for error correction. The bit rates in a set are related to the activity of the channel. If the channel is silent, only 1200 bits can be transferred, which improves the spreading by repeating each bit 8 times.

Frequency-Reuse Factor In an IS-95 system, the frequency-reuse factor is normally 1 because the interference from neighboring cells cannot affect CDMA or DSSS transmission.

Soft Handoff Every base station continuously broadcasts signals using its pilot channel. This means a mobile station can detect the pilot signal from its cell and neighboring cells. This enables a mobile station to do a soft handoff in contrast to a hard handoff.

PCS

Before we leave the discussion of second-generation cellular telephones, let us explain a term generally heard in relation to this generation: PCS. **Personal communications system (PCS)** does not refer to a single technology such as GSM, IS-136, or IS-95. It is a generic name for a commercial system that offers several kinds of communication services. Common features of these systems can be summarized:

1. They may use any second-generation technology (GSM, IS-136, or IS-95).
2. They use the 1900-MHz band, which means that a mobile station needs more power because higher frequencies have a shorter range than lower ones. However, since a station's power is limited by the FCC, the base station and the mobile station need to be close to each other (smaller cells).
3. They offer communication services such as short message service (SMS) and limited Internet access.

Third Generation

The third generation of cellular telephony refers to a combination of technologies that provide a variety of services. Ideally, when it matures, the third generation can provide both digital data and voice communication. Using a small portable device, a person should be able to talk to anyone else in the world with a voice quality similar to that of the existing fixed telephone network. A person can download and watch a movie, can download and listen to music, can surf the Internet or play games, can have a video conference, and can do much more. One of the interesting characteristics of a third-generation system is that the portable device is always connected; you do not need to dial a number to connect to the Internet.

The third-generation concept started in 1992, when ITU issued a blueprint called the **Internet Mobile Communication 2000 (IMT-2000).** The blueprint defines some criteria for third-generation technology as outlined below:

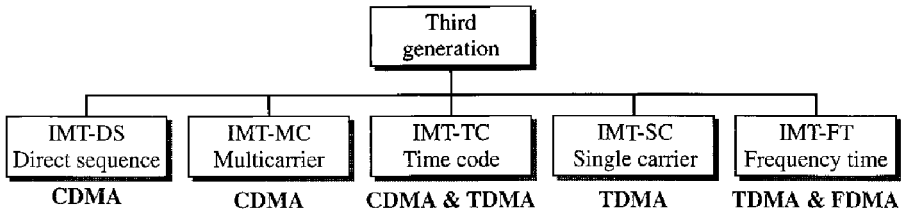
- ❑ Voice quality comparable to that of the existing public telephone network.
- ❑ Data rate of 144 kbps for access in a moving vehicle (car), 384 kbps for access as the user walks (pedestrians), and 2 Mbps for the stationary user (office or home).
- ❑ Support for packet-switched and circuit-switched data services.
- ❑ A band of 2 GHz.
- ❑ Bandwidths of 2 MHz.
- ❑ Interface to the Internet.

The main goal of third-generation cellular telephony is to provide universal personal communication.

IMT-2000 Radio Interface

Figure 16.12 shows the radio interfaces (wireless standards) adopted by IMT-2000. All five are developed from second-generation technologies. The first two evolve from CDMA technology. The third evolves from a combination of CDMA and TDMA. The fourth evolves from TDMA, and the last evolves from both FDMA and TDMA.

Figure 16.12 *IMT-2000 radio interfaces*



IMT-DS This approach uses a version of CDMA called wideband CDMA or W-CDMA. W-CDMA uses a 5-MHz bandwidth. It was developed in Europe, and it is compatible with the CDMA used in IS-95.

IMT-MC This approach was developed in North America and is known as CDMA 2000. It is an evolution of CDMA technology used in IS-95 channels. It combines the new wideband (15-MHz) spread spectrum with the narrowband (1.25-MHz) CDMA of IS-95. It is backward-compatible with IS-95. It allows communication on multiple 1.25-MHz channels (1, 3, 6, 9, 12 times), up to 15 MHz. The use of the wider channels allows it to reach the 2-Mbps data rate defined for the third generation.

IMT-TC This standard uses a combination of W-CDMA and TDMA. The standard tries to reach the IMT-2000 goals by adding TDMA multiplexing to W-CDMA.

IMT-SC This standard only uses TDMA.

IMT-FT This standard uses a combination of FDMA and TDMA.

16.2 SATELLITE NETWORKS

A **satellite network** is a combination of nodes, some of which are satellites, that provides communication from one point on the Earth to another. A node in the network can be a satellite, an Earth station, or an end-user terminal or telephone. Although a natural satellite, such as the Moon, can be used as a relaying node in the network, the use of artificial satellites is preferred because we can install electronic equipment on the satellite to regenerate the signal that has lost its energy during travel. Another restriction on using natural satellites is their distances from the Earth, which create a long delay in communication.

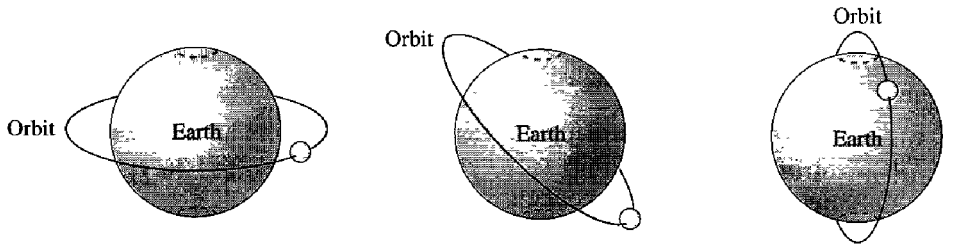
Satellite networks are like cellular networks in that they divide the planet into cells. Satellites can provide transmission capability to and from any location on Earth, no matter how remote. This advantage makes high-quality communication available to

undeveloped parts of the world without requiring a huge investment in ground-based infrastructure.

Orbits

An artificial satellite needs to have an **orbit,** the path in which it travels around the Earth. The orbit can be equatorial, inclined, or polar, as shown in Figure 16.13.

Figure 16.13 *Satellite orbits*



a. Equatorial-orbit satellite b. Inclined-orbit satellite c. Polar-orbit satellite

The period of a satellite, the time required for a satellite to make a complete trip around the Earth, is determined by Kepler's law, which defines the period as a function of the distance of the satellite from the center of the Earth.

Example 16.1

What is the period of the Moon, according to Kepler's law?

$$\text{Period} = C \times \text{distance}^{1.5}$$

Here C is a constant approximately equal to 1/100. The period is in seconds and the distance in kilometers.

Solution

The Moon is located approximately 384,000 km above the Earth. The radius of the Earth is 6378 km. Applying the formula, we get

$$\text{Period} = \frac{1}{100}(384{,}000 + 6378)^{1.5} = 2{,}439{,}090 \text{ s} = 1 \text{ month}$$

Example 16.2

According to Kepler's law, what is the period of a satellite that is located at an orbit approximately 35,786 km above the Earth?

Solution

Applying the formula, we get

$$\text{Period} = \frac{1}{100}(35{,}786 + 6378)^{1.5} = 86{,}579 \text{ s} = 24 \text{ h}$$

This means that a satellite located at 35,786 km has a period of 24 h, which is the same as the rotation period of the Earth. A satellite like this is said to be *stationary* to the Earth. The orbit, as we will see, is called a geosynchronous orbit.

Footprint

Satellites process microwaves with bidirectional antennas (line-of-sight). Therefore, the signal from a satellite is normally aimed at a specific area called the **footprint.** The signal power at the center of the footprint is maximum. The power decreases as we move out from the footprint center. The boundary of the footprint is the location where the power level is at a predefined threshold.

Three Categories of Satellites

Based on the location of the orbit, satellites can be divided into three categories: **geostationary Earth orbit (GEO), low-Earth-orbit (LEO),** and **middle-Earth-orbit (MEO).** Figure 16.14 shows the taxonomy.

Figure 16.14 *Satellite categories*

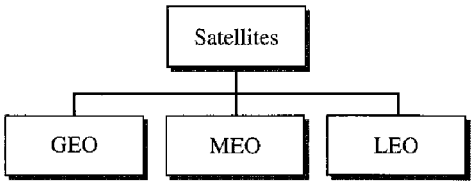
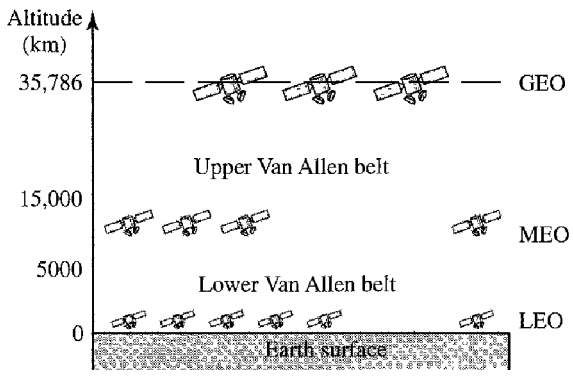


Figure 16.15 shows the satellite altitudes with respect to the surface of the Earth. There is only one orbit, at an altitude of 35,786 km for the GEO satellite. MEO satellites are located at altitudes between 5000 and 15,000 km. LEO satellites are normally below an altitude of 2000 km.

Figure 16.15 *Satellite orbit altitudes*



One reason for having different orbits is due to the existence of two Van Allen belts. A Van Allen belt is a layer that contains charged particles. A satellite orbiting in one of these two belts would be totally destroyed by the energetic charged particles. The MEO orbits are located between these two belts.

Frequency Bands for Satellite Communication

The frequencies reserved for satellite microwave communication are in the gigahertz (GHz) range. Each satellite sends and receives over two different bands. Transmission from the Earth to the satellite is called the **uplink.** Transmission from the satellite to the Earth is called the **downlink.** Table 16.1 gives the band names and frequencies for each range.

Table 16.1 *Satellite frequency bands*

Band	*Downlink, GHz*	*Uplink, GHz*	*Bandwidth, MHz*
L	1.5	1.6	15
S	1.9	2.2	70
C	4.0	6.0	500
Ku	11.0	14.0	500
Ka	20.0	30.0	3500

GEO Satellites

Line-of-sight propagation requires that the sending and receiving antennas be locked onto each other's location at all times (one antenna must have the other in sight). For this reason, a satellite that moves faster or slower than the Earth's rotation is useful only for short periods. To ensure constant communication, the satellite must move at the same speed as the Earth so that it seems to remain fixed above a certain spot. Such satellites are called *geostationary.*

Because orbital speed is based on the distance from the planet, only one orbit can be geostationary. This orbit occurs at the equatorial plane and is approximately 22,000 mi from the surface of the Earth.

But one geostationary satellite cannot cover the whole Earth. One satellite in orbit has line-of-sight contact with a vast number of stations, but the curvature of the Earth still keeps much of the planet out of sight. It takes a minimum of three satellites equidistant from each other in geostationary Earth orbit (GEO) to provide full global transmission. Figure 16.16 shows three satellites, each 120° from another in geosynchronous orbit around the equator. The view is from the North Pole.

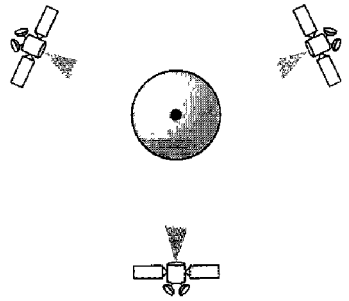
MEO Satellites

Medium-Earth-orbit (MEO) satellites are positioned between the two Van Allen belts. A satellite at this orbit takes approximately 6–8 hours to circle the Earth.

Global Positioning System

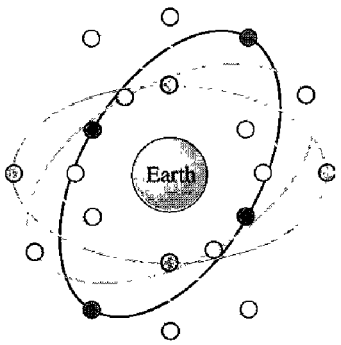
One example of a MEO satellite system is the **Global Positioning System (GPS),** constracted and operated by the US Department of Defense, orbiting at an altitude about

Figure 16.16 *Satellites in geostationary orbit*



18,000 km (11,000 mi) above the Earth. The system consists of 24 satellites and is used for land, sea, and air navigation to provide time and locations for vehicles and ships. GPS uses 24 satellites in six orbits, as shown in Figure 16.17. The orbits and the locations of the satellites in each orbit are designed in such a way that, at any time, four satellites are visible from any point on Earth. A GPS receiver has an almanac that tells the current position of each satellite.

Figure 16.17 *Orbits for global positioning system (GPS) satellites*

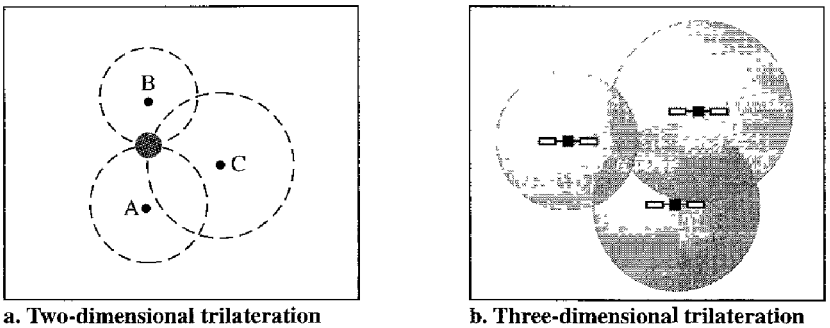


Trilateration GPS is based on a principle called **trilateration.**[†] On a plane, if we know our distance from three points, we know exactly where we are. Let us say that we are 10 miles away from point A, 12 miles away from point B, and 15 miles away from point C. If we draw three circles with the centers at A, B, and C, we must be somewhere on circle A, somewhere on circle B, and somewhere on circle C. These three circles meet at one single point (if our distances are correct), our position. Figure 16.18a shows the concept.

In three-dimensional space, the situation is different. Three spheres meet in two points as shown in Figure 16.18b. We need at least four spheres to find our exact position in space (longitude, latitude, and altitude). However, if we have additional facts about our location (for example, we know that we are not inside the ocean or somewhere in

[†]The terms *trilateration* and *triangulation* are normally used interchangeably. We use the word *trilateration*, which means using three distances, instead of **triangulation,** which may mean using three angles.

Figure 16.18 *Trilateration on a plane*



a. Two-dimensional trilateration

b. Three-dimensional trilateration

space), three spheres are enough, because one of the two points, where the spheres meet, is so improbable that the other can be selected without a doubt.

Measuring the Distance The trilateration principle can find our location on the earth if we know our distance from three satellites and know the position of each satellite. The position of each satellite can be calculated by a GPS receiver (using the predetermined path of the satellites). The GPS receiver, then, needs to find its distance from at least three GPS satellites (center of the spheres). Measuring the distance is done using a principle called one-way ranging. For the moment, let us assume that all GPS satellites and the receiver on the Earth are synchronized. Each of 24 satellites synchronously transmits a complex signal each having a unique pattern. The computer on the receiver measures the delay between the signals from the satellites and its copy of signals to determine the distances to the satellites.

Synchronization The previous discussion was based on the assumption that the satellites' clock are synchronized with each other and with the receiver's clock. Satellites use atomic clock that are precise and can function synchronously with each other. The receiver's clock however, is a normal quartz clock (an atomic clock costs more that $50,000), and there is no way to synchronize it with the satellite clocks. There is an unknown offset between the satellite clocks and the receiver clock that introduces a corresponding offset in the distance calculation. Because of this offset, the measured distance is called a *pseudorange*.

GPS uses an elegant solution to the clock offset problem, by recognizing that the offset's value is the same for all satellite being used. The calculation of position becomes finding four unknowns: the x_r, y_r, z_r coordinates of the receiver, and common clock offset *dt*. For finding these four unknown values, we need at least four equations. This means that we need to measure pesudoranges from four satellite instead of three. If we call the four measured pseudoranges PR1, PR2, PR3 and PR4 and the coordinates of each satellite x_i, y_i, and z_i (for i =1 to 4), we can find the four previously mentioned unknown values using the following four equations (the four unknown values are shown in color).

$$\mathbf{PR_1} = [(x_1 - x_r)^2 + (y_1 - y_r)^2 + (z_1 - z_r)^2]^{1/2} + c \times dt$$
$$\mathbf{PR_2} = [(x_2 - x_r)^2 + (y_2 - y_r)^2 + (z_2 - z_r)^2]^{1/2} + c \times dt$$
$$\mathbf{PR_3} = [(x_3 - x_r)^2 + (y_3 - y_r)^2 + (z_3 - z_r)^2]^{1/2} + c \times dt$$
$$\mathbf{PR_4} = [(x_4 - x_r)^2 + (y_4 - y_r)^2 + (z_4 - z_r)^2]^{1/2} + c \times dt$$

The coordinates used in the above formulas are in an Earth-Centered Earth-Fixed (ECEF) reference frame, which means that the origin of the coordinate space is at the center of the Earth and the coordinate space rotate with the Earth. This implies that the ECEF coordinates of a fixed point on the surface of the earth do not change.

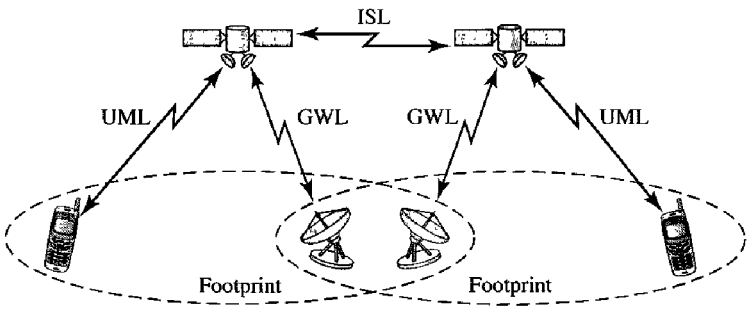
Application GPS is used by military forces. For example, thousands of portable GPS receivers were used during the Persian Gulf war by foot soldiers, vehicles, and helicopters. Another use of GPS is in navigation. The driver of a car can find the location of the car. The driver can then consult a database in the memory of the automobile to be directed to the destination. In other words, GPS gives the location of the car, and the database uses this information to find a path to the destination. A very interesting application is clock synchronization. As we mentioned previously, the IS-95 cellular telephone system uses GPS to create time synchronization between the base stations.

LEO Satellites

Low-Earth-orbit (LEO) satellites have polar orbits. The altitude is between 500 and 2000 km, with a rotation period of 90 to 120 min. The satellite has a speed of 20,000 to 25,000 km/h. An LEO system usually has a cellular type of access, similar to the cellular telephone system. The footprint normally has a diameter of 8000 km. Because LEO satellites are close to Earth, the round-trip time propagation delay is normally less than 20 ms, which is acceptable for audio communication.

An LEO system is made of a constellation of satellites that work together as a network; each satellite acts as a switch. Satellites that are close to each other are connected through intersatellite links (ISLs). A mobile system communicates with the satellite through a user mobile link (UML). A satellite can also communicate with an Earth station (gateway) through a gateway link (GWL). Figure 16.19 shows a typical LEO satellite network.

Figure 16.19 *LEO satellite system*



LEO satellites can be divided into three categories: little LEOs, big LEOs, and broadband LEOs. The little LEOs operate under 1 GHz. They are mostly used for low-data-rate messaging. The big LEOs operate between 1 and 3 GHz. Globalstar and Iridium systems are examples of big LEOs. The broadband LEOs provide communication similar to fiber-optic networks. The first broadband LEO system was Teledesic.

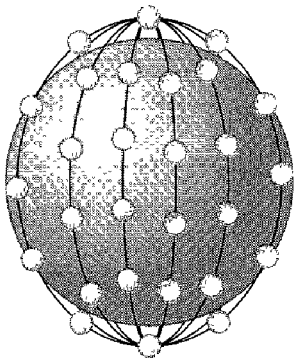
Iridium System

The concept of the **Iridium** system, a 77-satellite network, was started by Motorola in 1990. The project took eight years to materialize. During this period, the number of satellites was reduced. Finally, in 1998, the service was started with 66 satellites. The original name, Iridium, came from the name of the 77th chemical element; a more appropriate name is Dysprosium (the name of element 66).

Iridium has gone through rough times. The system was halted in 1999 due to financial problems; it was sold and restarted in 2001 under new ownership.

The system has 66 satellites divided into six orbits, with 11 satellites in each orbit. The orbits are at an altitude of 750 km. The satellites in each orbit are separated from one another by approximately 32° of latitude. Figure 16.20 shows a schematic diagram of the constellation.

Figure 16.20 *Iridium constellation*



The Iridium system has 66 satellites in six LEO orbits, each at an altitude of 750 km.

Since each satellite has 48 spot beams, the system can have up to 3168 beams. However, some of the beams are turned off as the satellite approaches the pole. The number of active spot beams at any moment is approximately 2000. Each spot beam covers a cell on Earth, which means that Earth is divided into approximately 2000 (overlapping) cells.

In the Iridium system, communication between two users takes place through satellites. When a user calls another user, the call can go through several satellites before reaching the destination. This means that relaying is done in space and each satellite needs to be sophisticated enough to do relaying. This strategy eliminates the need for many terrestrial stations.

The whole purpose of Iridium is to provide direct worldwide communication using handheld terminals (same concept as cellular telephony). The system can be used for voice, data, paging, fax, and even navigation. The system can provide connectivity between users at locations where other types of communication are not possible. The system provides 2.4- to 4.8-kbps voice and data transmission between portable telephones. Transmission occurs in the 1.616- to 1.6126-GHz frequency band. Intersatellite communication occurs in the 23.18- to 23.38-GHz frequency band.

Iridium is designed to provide direct worldwide voice and data communication using handheld terminals, a service similar to cellular telephony but on a global scale.

Globalstar

Globalstar is another LEO satellite system. The system uses 48 satellites in six polar orbits with each orbit hosting eight satellites. The orbits are located at an altitude of almost 1400 km.

The Globalstar system is similar to the Iridium system; the main difference is the relaying mechanism. Communication between two distant users in the Iridium system requires relaying between several satellites; Globalstar communication requires both satellites and Earth stations, which means that ground stations can create more powerful signals.

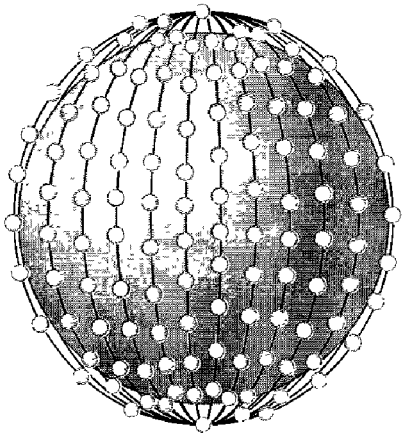
Teledesic

Teledesic is a system of satellites that provides fiber-optic-like (broadband channels, low error rate, and low delay) communication. Its main purpose is to provide broadband Internet access for users all over the world. It is sometimes called "Internet in the sky."

The project was started in 1990 by Craig McCaw and Bill Gates; later, other investors joined the consortium. The project is scheduled to be fully functional in the near future.

Constellation Teledesic provides 288 satellites in 12 polar orbits with each orbit hosting 24 satellites. The orbits are at an altitude of 1350 km, as shown in Figure 16.21.

Figure 16.21 *Teledesic*



Teledesic has 288 satellites in 12 LEO orbits, each at an altitude of 1350 km.

Communication The system provides three types of communication. Intersatellite communication allows eight neighboring satellites to communicate with one another. Communication is also possible between a satellite and an Earth gateway station. Users can communicate directly with the network using terminals. Earth is divided into tens of thousands of cells. Each cell is assigned a time slot, and the satellite focuses its beam to the cell

at the corresponding time slot. The terminal can send data during its time slot. A terminal receives all packets intended for the cell, but selects only those intended for its address.

Bands Transmission occurs in the Ka bands.

Data Rate The data rate is up to 155 Mbps for the uplink and up to 1.2 Gbps for the downlink.

16.3 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

Wireless WANs are completely covered in [Sta02], [Jam03], [AZ03], and [Sch03]. Communication satellites are discussed in Section 2.4 of [Tan03] and Section 8.5 of [Cou01]. Mobile telephone system is discussed in Section 2.6 of [Tan03] and Section 8.8 of [Cou01].

16.4 KEY TERMS

Advanced Mobile Phone System (AMPS)
cellular telephony
digital AMPS (D-AMPS)
downlink
footprint
geostationary Earth orbit (GEO)
Global Positioning System (GPS)
Global System for Mobile Communication (GSM)
Globalstar
handoff
Interim Standard 95 (IS-95)
Internet Mobile Communication 2000 (IMT-2000)
Iridium
low-Earth-orbit (LEO)
medium-Earth-orbit (MEO)
mobile switching center (MSC)
orbit
personal communications system (PCS)
reuse factor
roaming
satellite network
Teledesic
triangulation
trilateration
uplink

16.5 SUMMARY

- Cellular telephony provides communication between two devices. One or both may be mobile.
- A cellular service area is divided into cells.
- Advanced Mobile Phone System (AMPS) is a first-generation cellular phone system.

- Digital AMPS (D-AMPS) is a second-generation cellular phone system that is a digital version of AMPS.
- Global System for Mobile Communication (GSM) is a second-generation cellular phone system used in Europe.
- Interim Standard 95 (IS-95) is a second-generation cellular phone system based on CDMA and DSSS.
- The third-generation cellular phone system will provide universal personal communication.
- A satellite network uses satellites to provide communication between any points on Earth.
- A geostationary Earth orbit (GEO) is at the equatorial plane and revolves in phase with Earth.
- Global Positioning System (GPS) satellites are medium-Earth-orbit (MEO) satellites that provide time and location information for vehicles and ships.
- Iridium satellites are low-Earth-orbit (LEO) satellites that provide direct universal voice and data communications for handheld terminals.
- Teledesic satellites are low-Earth-orbit satellites that will provide universal broadband Internet access.

16.6 PRACTICE SET

Review Questions

1. What is the relationship between a base station and a mobile switching center?
2. What are the functions of a mobile switching center?
3. Which is better, a low reuse factor or a high reuse factor? Explain your answer.
4. What is the difference between a hard handoff and a soft handoff?
5. What is AMPS?
6. What is the relationship between D-AMPS and AMPS?
7. What is GSM?
8. What is the function of the CDMA in IS-95?
9. What are the three types of orbits?
10. Which type of orbit does a GEO satellite have? Explain your answer.
11. What is a footprint?
12. What is the relationship between the Van Allen belts and satellites?
13. Compare an uplink with a downlink.
14. What is the purpose of GPS?
15. What is the main difference between Iridium and Globalstar?

Exercises

16. Draw a cell pattern with a frequency-reuse factor of 3.
17. What is the maximum number of callers in each cell in AMPS?

18. What is the maximum number of simultaneous calls in each cell in an IS-136 (D-AMPS) system, assuming no analog control channels?
19. What is the maximum number of simultaneous calls in each cell in a GSM assuming no analog control channels?
20. What is the maximum number of callers in each cell in an IS-95 system?
21. Find the efficiency of AMPS in terms of simultaneous calls per megahertz of bandwidth. In other words, find the number of calls that can be used in 1-MHz bandwidth allocation.
22. Repeat Exercise 21 for D-AMPS.
23. Repeat Exercise 21 for GSM.
24. Repeat Exercise 21 for IS-95.
25. Guess the relationship between a 3-kHz voice channel and a 30-kHz modulated channel in a system using AMPS.
26. How many slots are sent each second in a channel using D-AMPS? How many slots are sent by each user in 1 s?
27. Use Kepler's formula to check the accuracy of a given period and altitude for a GPS satellite.
28. Use Kepler's formula to check the accuracy of a given period and altitude for an Iridium satellite.
29. Use Kepler's formula to check the accuracy of a given period and altitude for a Globalstar satellite.

CHAPTER 17

SONET/SDH

In this chapter, we introduce a wide area network (WAN), SONET, that is used as a transport network to carry loads from other WANs. We first discuss SONET as a protocol, and we then show how SONET networks can be constructed from the standards defined in the protocol.

The high bandwidths of fiber-optic cable are suitable for today's high-data-rate technologies (such as video conferencing) and for carrying large numbers of lower-rate technologies at the same time. For this reason, the importance of fiber optics grows in conjunction with the development of technologies requiring high data rates or wide bandwidths for transmission. With their prominence came a need for standardization. The United States (ANSI) and Europe (ITU-T) have responded by defining standards that, though independent, are fundamentally similar and ultimately compatible. The ANSI standard is called the **Synchronous Optical Network (SONET).** The ITU-T standard is called the **Synchronous Digital Hierarchy (SDH).**

SONET was developed by ANSI; SDH was developed by ITU-T.

SONET/SDH is a synchronous network using synchronous TDM multiplexing. All clocks in the system are locked to a master clock.

17.1 ARCHITECTURE

Let us first introduce the architecture of a SONET system: signals, devices, and connections.

Signals

SONET defines a hierarchy of electrical signaling levels called **synchronous transport signals (STSs).** Each STS level (STS-1 to STS-192) supports a certain data rate, specified in megabits per second (see Table 17.1). The corresponding optical signals are called **optical carriers (OCs).** SDH specifies a similar system called a **synchronous transport module (STM).** STM is intended to be compatible with existing European

hierarchies, such as E lines, and with STS levels. To this end, the lowest STM level, STM-1, is defined as 155.520 Mbps, which is exactly equal to STS-3.

Table 17.1 *SONET/SDH rates*

STS	*OC*	*Rate (Mbps)*	*STM*
STS-1	OC-1	51.840	
STS-3	OC-3	155.520	**STM-1**
STS-9	OC-9	466.560	**STM-3**
STS-12	OC-12	622.080	**STM-4**
STS-18	OC-18	933.120	**STM-6**
STS-24	OC-24	1244.160	**STM-8**
STS-36	OC-36	1866.230	**STM-12**
STS-48	OC-48	2488.320	**STM-16**
STS-96	OC-96	4976.640	**STM-32**
STS-192	OC-192	9953.280	**STM-64**

A glance through Table 17.1 reveals some interesting points. First, the lowest level in this hierarchy has a data rate of 51.840 Mbps, which is greater than that of the DS-3 service (44.736 Mbps). In fact, the STS-1 is designed to accommodate data rates equivalent to those of the DS-3. The difference in capacity is provided to handle the overhead needs of the optical system.

Second, the STS-3 rate is exactly three times the STS-1 rate; and the STS-9 rate is exactly one-half the STS-18 rate. These relationships mean that 18 STS-1 channels can be multiplexed into one STS-18, six STS-3 channels can be multiplexed into one STS-18, and so on.

SONET Devices

Figure 17.1 shows a simple link using SONET devices. SONET transmission relies on three basic devices: STS multiplexers/demultiplexers, regenerators, add/drop multiplexers and terminals.

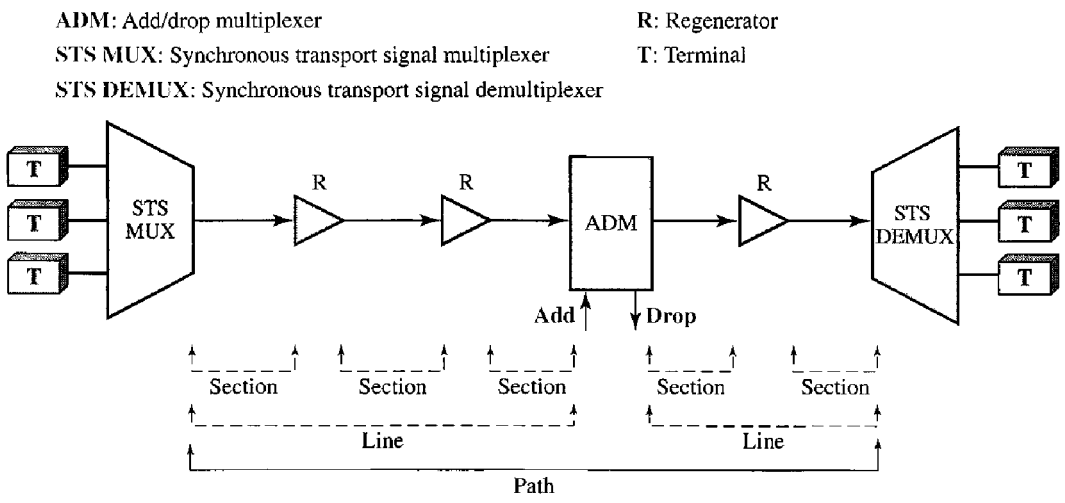
STS Multiplexer/Demultiplexer

STS multiplexers/demultiplexers mark the beginning points and endpoints of a SONET link. They provide the interface between an electrical tributary network and the optical network. An **STS multiplexer** multiplexes signals from multiple electrical sources and creates the corresponding OC signal. An **STS demultiplexer** demultiplexes an optical OC signal into corresponding electric signals.

Regenerator

Regenerators extend the length of the links. A **regenerator** is a repeater (see Chapter 15) that takes a received optical signal (OC-*n*), demodulates it into the corresponding electric signal (STS-*n*), regenerates the electric signal, and finally modulates the electric

Figure 17.1 *A simple network using SONET equipment*



signal into its correspondent OC-*n* signal. A SONET regenerator replaces some of the existing overhead information (header information) with new information.

Add/drop Multiplexer

Add/drop multiplexers allow insertion and extraction of signals. An **add/drop multiplexer (ADM)** can add STSs coming from different sources into a given path or can remove a desired signal from a path and redirect it without demultiplexing the entire signal. Instead of relying on timing and bit positions, add/drop multiplexers use header information such as addresses and pointers (described later in this section) to identify individual streams.

In the simple configuration shown by Figure 17.1, a number of incoming electronic signals are fed into an STS multiplexer, where they are combined into a single optical signal. The optical signal is transmitted to a regenerator, where it is recreated without the noise it has picked up in transit. The regenerated signals from a number of sources are then fed into an add/drop multiplexer. The add/drop multiplexer reorganizes these signals, if necessary, and sends them out as directed by information in the data frames. These remultiplexed signals are sent to another regenerator and from there to the receiving STS demultiplexer, where they are returned to a format usable by the receiving links.

Terminals

A **terminal** is a device that uses the services of a SONET network. For example, in the Internet, a terminal can be a router that needs to send packets to another router at the other side of a SONET network.

Connections

The devices defined in the previous section are connected using *sections, lines,* and *paths.*

Sections

A **section** is the optical link connecting two neighbor devices: multiplexer to multiplexer, multiplexer to regenerator, or regenerator to regenerator.

Lines

A **line** is the portion of the network between two multiplexers: STS multiplexer to add/drop multiplexer, two add/drop multiplexers, or two STS multiplexers.

Paths

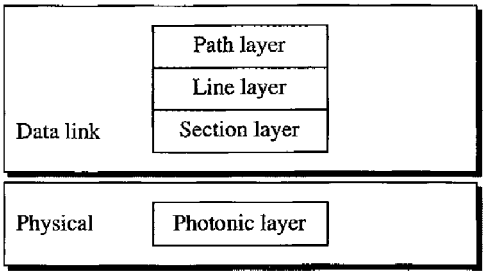
A **path** is the end-to-end portion of the network between two STS multiplexers. In a simple SONET of two STS multiplexers linked directly to each other, the section, line, and path are the same.

17.2 SONET LAYERS

The SONET standard includes four functional layers: the photonic, the section, the line, and the path layer. They correspond to both the physical and the data link layers (see Figure 17.2). The headers added to the frame at the various layers are discussed later in this chapter.

SONET defines four layers: path, line, section, and photonic.

Figure 17.2 *SONET layers compared with OSI or the Internet layers*



Path Layer

The **path layer** is responsible for the movement of a signal from its optical source to its optical destination. At the optical source, the signal is changed from an electronic form into an optical form, multiplexed with other signals, and encapsulated in a frame. At the optical destination, the received frame is demultiplexed, and the individual optical signals are changed back into their electronic forms. Path layer overhead is added at this layer. STS multiplexers provide path layer functions.

Line Layer

The **line layer** is responsible for the movement of a signal across a physical line. Line layer overhead is added to the frame at this layer. STS multiplexers and add/drop multiplexers provide line layer functions.

Section Layer

The **section layer** is responsible for the movement of a signal across a physical section. It handles framing, scrambling, and error control. Section layer overhead is added to the frame at this layer.

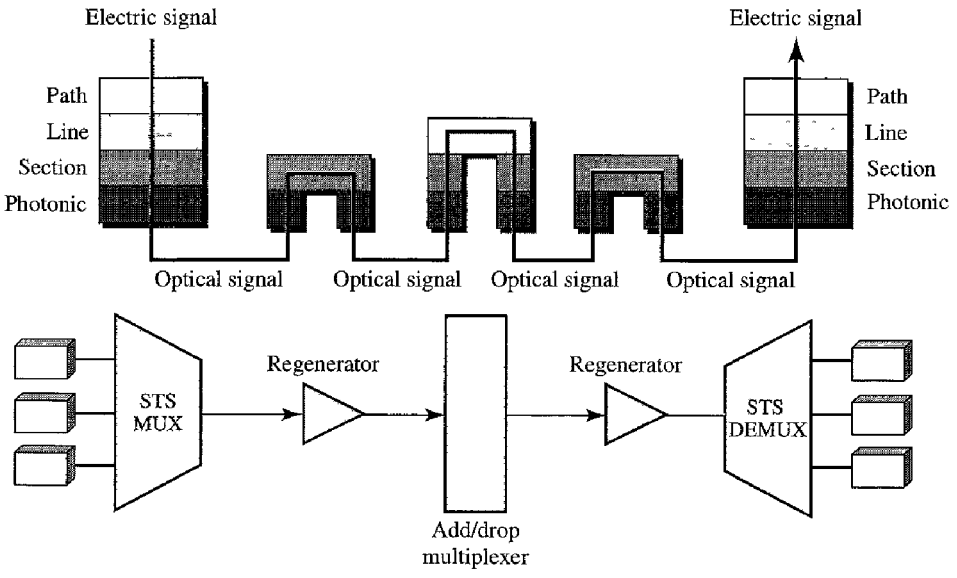
Photonic Layer

The **photonic layer** corresponds to the physical layer of the OSI model. It includes physical specifications for the optical fiber channel, the sensitivity of the receiver, multiplexing functions, and so on. SONET uses NRZ encoding with the presence of light representing 1 and the absence of light representing 0.

Device–Layer Relationships

Figure 17.3 shows the relationship between the devices used in SONET transmission and the four layers of the standard. As you can see, an STS multiplexer is a four-layer device. An add/drop multiplexer is a three-layer device. A regenerator is a two-layer device.

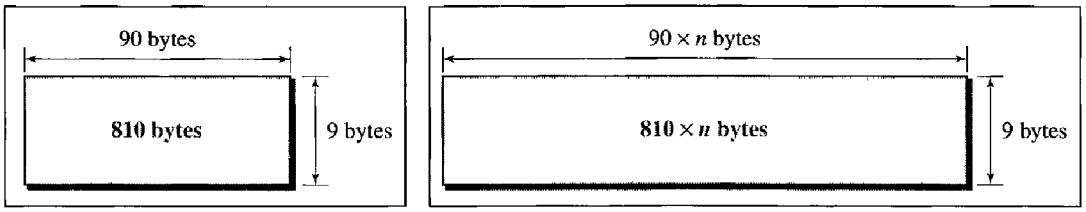
Figure 17.3 *Device–layer relationship in SONET*



17.3 SONET FRAMES

Each synchronous transfer signal STS-*n* is composed of 8000 frames. Each frame is a two-dimensional matrix of bytes with 9 rows by 90 × *n* columns. For example, STS-1 frame is 9 rows by 90 columns (810 bytes), and an STS-3 is 9 rows by 270 columns (2430 bytes). Figure 17.4 shows the general format of an STS-1 and an STS-*n*.

Figure 17.4 *An STS-1 and an STS-n frame*

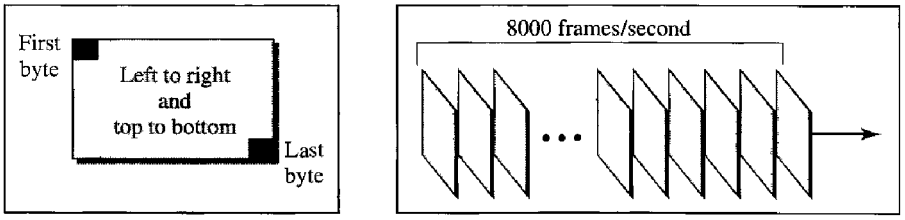


a. STS-1 frame b. STS-*n* frame

Frame, Byte, and Bit Transmission

One of the interesting points about SONET is that each STS-*n* signal is transmitted at a fixed rate of 8000 frames per second. This is the rate at which voice is digitized (see Chapter 4). For each frame the bytes are transmitted from the left to the right, top to the bottom. For each byte, the bits are transmitted from the most significant to the least significant (left to right). Figure 17.5 shows the order of frame and byte transmission.

Figure 17.5 *STS-1 frames in transition*



a. Byte transmission b. Frame transmission

A SONET STS-*n* signal is transmitted at 8000 frames per second.

If we sample a voice signal and use 8 bits (1 byte) for each sample, we can say that each byte in a SONET frame can carry information from a digitized voice channel. In other words, an STS-1 signal can carry 774 voice channels simultaneously (810 minus required bytes for overhead).

Each byte in a SONET frame can carry a digitized voice channel.

Example 17.1

Find the data rate of an STS-1 signal.

Solution

STS-1, like other STS signals, sends 8000 frames per second. Each STS-1 frame is made of 9 by (1 × 90) bytes. Each byte is made of 8 bits. The data rate is

$$\text{STS-1 data rate} = 8000 \times 9 \times (1 \times 90) \times 8 = 51.840 \text{ Mbps}$$

Example 17.2

Find the data rate of an STS-3 signal.

Solution

STS-3, like other STS signals, sends 8000 frames per second. Each STS-3 frame is made of 9 by (3 × 90) bytes. Each byte is made of 8 bits. The data rate is

$$\text{STS-3 data rate} = 8000 \times 9 \times (3 \times 90) \times 8 = 155.52 \text{ Mbps}$$

Note that in SONET, there is an exact relationship between the data rates of different STS signals. We could have found the data rate of STS-3 by using the data rate of STS-1 (multiply the latter by 3).

In SONET, the data rate of an STS-*n* signal is *n* times the data rate of an STS-1 signals.

Example 17.3

What is the duration of an STS-1 frame? STS-3 frame? STS-*n* frame?

Solution

In SONET, 8000 frames are sent per second. This means that the duration of an STS-1, STS-3, or STS-*n* frame is the same and equal to 1/8000 s, or 125 μs.

In SONET, the duration of any frame is 125 μs.

STS-1 Frame Format

The basic format of an STS-1 frame is shown in Figure 17.6. As we said before, a SONET frame is a matrix of 9 rows of 90 bytes (octets) each, for a total of 810 bytes.

The first three columns of the frame are used for section and line overhead. The upper three rows of the first three columns are used for **section overhead (SOH).** The lower six are **line overhead (LOH).** The rest of the frame is called the synchronous payload envelope (SPE). It contains user data and **path overhead (POH)** needed at the user data level. We will discuss the format of the SPE shortly.

Section Overhead

The section overhead consists of nine octets. The labels, functions, and organization of these octets are shown in Figure 17.7.

Figure 17.6 *STS-1 frame overheads*

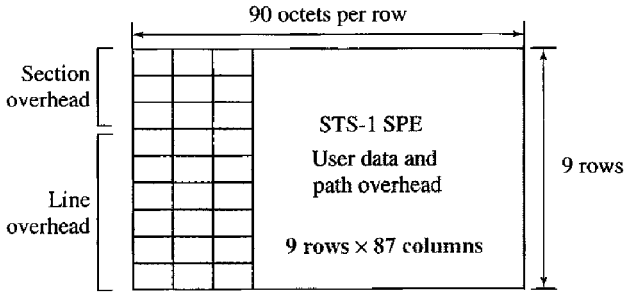
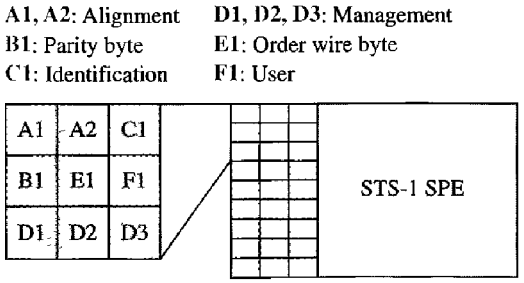


Figure 17.7 *STS-1 frame: section overhead*



- **Alignment bytes (A1 and A2).** Bytes A1 and A2 are used for framing and synchronization and are called alignment bytes. These bytes alert a receiver that a frame is arriving and give the receiver a predetermined bit pattern on which to synchronize. The bit patterns for these two bytes in hexadecimal are 0xF628. The bytes serve as a flag.
- **Section parity byte (B1).** Byte B1 is for bit interleaved parity (BIP-8). Its value is calculated over all bytes of the previous frame. In other words, the *i*th bit of this byte is the parity bit calculated over all *i*th bits of the previous STS-*n* frame. The value of this byte is filled only for the first STS-1 in an STS-*n* frame. In other words, although an STS-*n* frame has *n* B1 bytes, as we will see later, only the first byte has this value; the rest are filled with 0s.
- **Identification byte (C1).** Byte C1 carries the identity of the STS-1 frame. This byte is necessary when multiple STS-1s are multiplexed to create a higher-rate STS (STS-3, STS-9, STS-12, etc.). Information in this byte allows the various signals to be recognized easily upon demultiplexing. For example, in an STS-3 signal, the value of the C1 byte is 1 for the first STS-1; it is 2 for the second; and it is 3 for the third.
- **Management bytes (D1, D2, and D3).** Bytes D1, D2, and D3 together form a 192-kbps channel ($3 \times 8000 \times 8$) called the data communication channel. This channel is required for operation, administration, and maintenance (OA&M) signaling.
- **Order wire byte (E1).** Byte E1 is the order wire byte. Order wire bytes in consecutive frames form a channel of 64 kbps (8000 frames per second times 8 bits per

frame). This channel is used for communication between regenerators, or between terminals and regenerators.

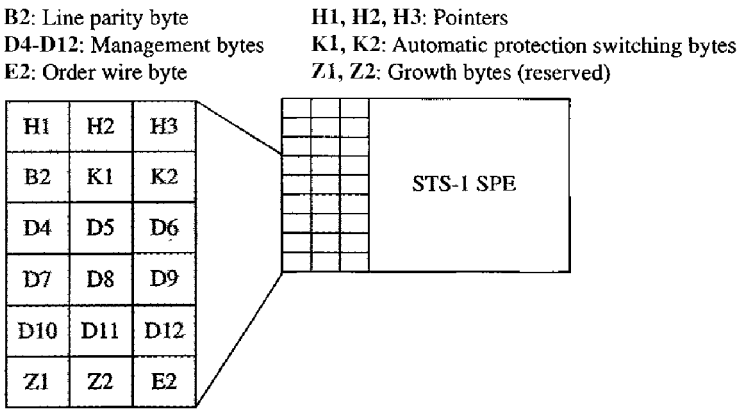
- **User's byte (F1).** The F1 bytes in consecutive frames form a 64-kbps channel that is reserved for user needs at the section level.

Section overhead is recalculated for each SONET device (regenerators and multiplexers).

Line Overhead

Line overhead consists of 18 bytes. The labels, functions, and arrangement of these bytes are shown in Figure 17.8.

Figure 17.8 *STS-1 frame: line overhead*



- **Line parity byte (B2).** Byte B2 is for bit interleaved parity. It is for error checking of the frame over a line (between two multiplexers). In an STS-*n* frame, B2 is calculated for all bytes in the previous STS-1 frame and inserted at the B2 byte for that frame. In other words, in a STS-3 frame, there are three B2 bytes, each calculated for one STS-1 frame. Contrast this byte with B1 in the section overhead.
- **Data communication channel bytes (D4 to D12).** The line overhead D bytes (D4 to D12) in consecutive frames form a 576-kbps channel that provides the same service as the D1–D3 bytes (OA&M), but at the line rather than the section level (between multiplexers).
- **Order wire byte (E2).** The E2 bytes in consecutive frames form a 64-kbps channel that provides the same functions as the E1 order wire byte, but at the line level.
- **Pointer bytes (H1, H2, and H3).** Bytes H1, H2, and H3 are pointers. The first two bytes are used to show the offset of the SPE in the frame; the third is used for justification. We show the use of these bytes later.
- **Automatic protection switching bytes (K1 and K2).** The K1 and K2 bytes in consecutive frames form a 128-kbps channel used for automatic detection of problems in

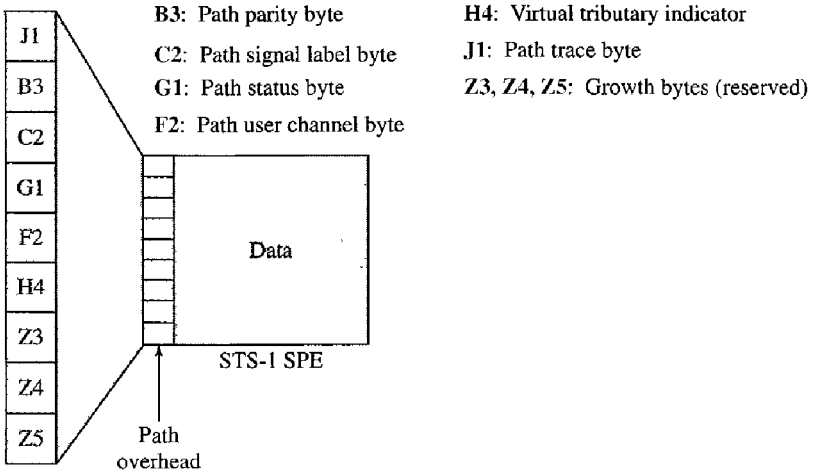
line-terminating equipment. We discuss automatic protection switching (APS) later in the chapter.

- **Growth bytes (Z1 and Z2).** The Z1 and Z2 bytes are reserved for future use.

Synchronous Payload Envelope

The **synchronous payload envelope (SPE)** contains the user data and the overhead related to the user data (path overhead). One SPE does not necessarily fit it into one STS-1 frame; it may be split between two frames, as we will see shortly. This means that the path overhead, the leftmost column of an SPE, does not necessarily align with the section or line overhead. The path overhead must be added first to the user data to create an SPE, and then an SPE can be inserted into one or two frames. Path overhead consists of 9 bytes. The labels, functions, and arrangement of these bytes are shown in Figure 17.9.

Figure 17.9 *STS-1 frame: path overhead*



- **Path parity byte (B3).** Byte B3 is for bit interleaved parity, like bytes B1 and B2, but calculated over SPE bits. It is actually calculated over the previous SPE in the stream.
- **Path signal label byte (C2).** Byte C2 is the path identification byte. It is used to identify different protocols used at higher levels (such as IP or ATM) whose data are being carried in the SPE.
- **Path user channel byte (F2).** The F2 bytes in consecutive frames, like the F1 bytes, form a 64-kbps channel that is reserved for user needs, but at the path level.
- **Path status byte (G1).** Byte G1 is sent by the receiver to communicate its status to the sender. It is sent on the reverse channel when the communication is duplex. We will see its use in the linear or ring networks later in the chapter.
- **Multiframe indicator (H4).** Byte H4 is the multiframe indicator. It indicates payloads that cannot fit into a single frame. For example, virtual tributaries can be

combined to form a frame that is larger than an SPE frame and need to be divided into different frames. Virtual tributaries are discussed in the next section.

- **Path trace byte (J1).** The J1 bytes in consecutive frames form a 64-kbps channel used for tracking the path. The J1 byte sends a continuous 64-byte string to verify the connection. The choice of the string is left to the application program. The receiver compares each pattern with the previous one to ensure nothing is wrong with the communication at the path layer.
- **Growth bytes (Z3, Z4, and Z5).** Bytes Z3, Z4, and Z5 are reserved for future use.

Path overhead is only calculated for end-to-end (at STS multiplexers).

Overhead Summary

Table 17.2 compares and summarizes the overheads used in a section, line, and path.

Table 17.2 *SONET/SDH rates*

Byte Function	*Section*	*Line*	*Path*
Alignment	A1, A2		
Parity	B1	B2	B3
Identifier	C1		C2
OA&M	D1–D3	D4–D12	
Order wire	E1		
User	F1		F2
Status			G1
Pointers		H1– H3	H4
Trace			J1
Failure tolerance		K1, K2	
Growth (reserved for future)		Z1, Z2	Z3–Z5

Example 17.4

What is the user data rate of an STS-1 frame (without considering the overheads)?

Solution

The user data part in an STS-1 frame is made of 9 rows and 86 columns. So we have

$$\text{STS-1 user data rate} = 8000 \times 9 \times (1 \times 86) \times 8 = 49.536 \text{ Mbps}$$

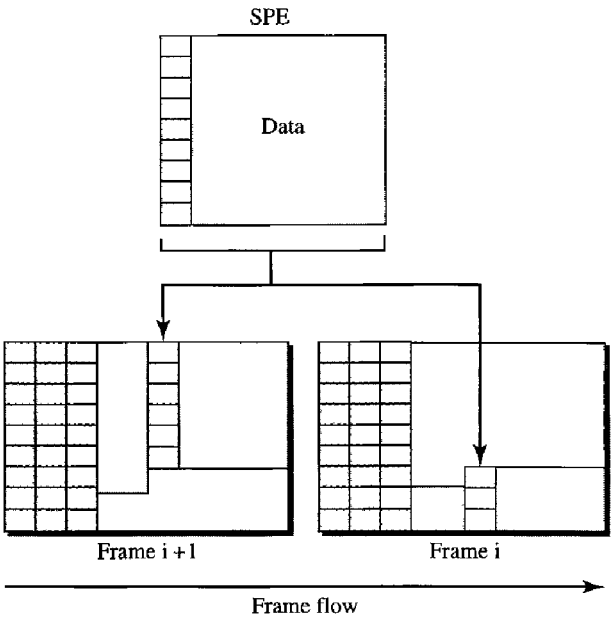
Encapsulation

The previous discussion reveals that an SPE needs to be encapsulated in an STS-1 frame. Encapsulation may create two problems that are handled elegantly by SONET using pointers (H1 to H3). We discuss the use of these bytes in this section.

Offsetting

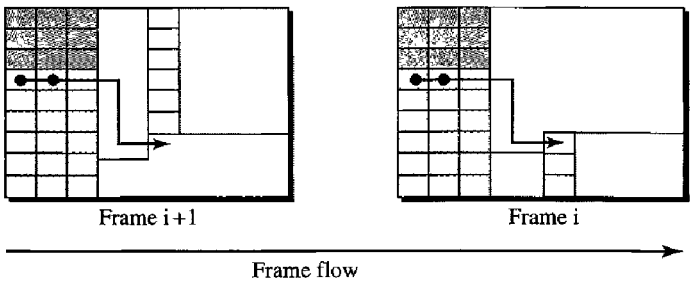
SONET allows one SPE to span two frames, part of the SPE is in the first frame and part is in the second. This may happen when one SPE that is to be encapsulated is not aligned time-wise with the passing synchronized frames. Figure 17.10 shows this situation. SPE bytes are divided between the two frames. The first set of bytes is encapsulated in the first frame; the second set is encapsulated in the second frame. The figure also shows the path overhead, which is aligned with the section/line overhead of any frame. The question is, How does the SONET multiplexer know where the SPE starts or ends in the frame? The solution is the use of pointers H1 and H2 to define the beginning of the SPE; the end can be found because each SPE has a fixed number of bytes. SONET allows the offsetting of an SPE with respect to an STS-1 frame.

Figure 17.10 *Offsetting of SPE related to frame boundary*



To find the beginning of each SPE in a frame, we need two pointers H1 and H2 in the line overhead. Note that these pointers are located in the line overhead because the encapsulation occurs at a multiplexer. Figure 17.11 shows how these 2 bytes point to

Figure 17.11 *The use of H1 and H2 pointers to show the start of an SPE in a frame*



the beginning of the SPEs. Note that we need 2 bytes to define the position of a byte in a frame; a frame has 810 bytes, which cannot be defined using 1 byte.

Example 17.5

What are the values of H1 and H2 if an SPE starts at byte number 650?

Solution

The number 650 can be expressed in four hexadecimal digits as 0x028A. This means the value of H1 is 0x02 and the value of H2 is 0x8A.

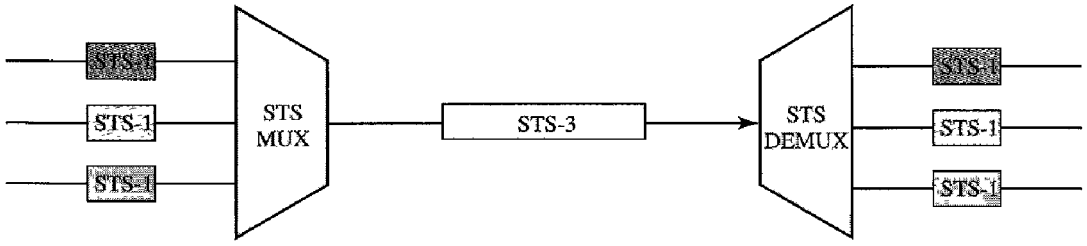
Justification

Now suppose the transmission rate of the payload is just slightly different from the transmission rate of SONET. First, assume that the rate of the payload is higher. This means that occasionally there is 1 extra byte that cannot fit in the frame. In this case, SONET allows this extra byte to be inserted in the H3 byte. Now, assume that the rate of the payload is lower. This means that occasionally 1 byte needs to be left empty in the frame. SONET allows this byte to be the byte after the H3 byte.

17.4 STS MULTIPLEXING

In SONET, frames of lower rate can be synchronously time-division multiplexed into a higher-rate frame. For example, three STS-1 signals (channels) can be combined into one STS-3 signal (channel), four STS-3s can be multiplexed into one STS-12, and so on, as shown in Figure 17.12.

Figure 17.12 *STS multiplexing/demultiplexing*



Multiplexing is synchronous TDM, and all clocks in the network are locked to a master clock to achieve synchronization.

In SONET, all clocks in the network are locked to a master clock.

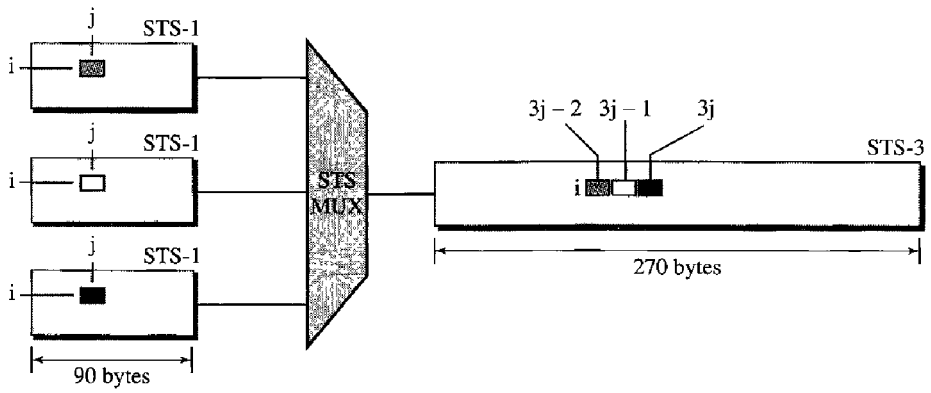
We need to mention that multiplexing can also take place at the higher data rates. For example, four STS-3 signals can be multiplexed into an STS-12 signal. However, the STS-3 signals need to first be demultiplexed into 12 STS-1 signals, and then these

twelve signals need to be multiplexed into an STS-12 signal. The reason for this extra work will be clear after our discussion on byte interleaving.

Byte Interleaving

Synchronous TDM multiplexing in SONET is achieved by using **byte interleaving.** For example, when three STS-1 signals are multliplexed into one STS-3 signal, each set of 3 bytes in the STS-3 signal is associated with 1 byte from each STS-1 signal. Figure 17.13 shows the interleaving.

Figure 17.13 *Byte interleaving*



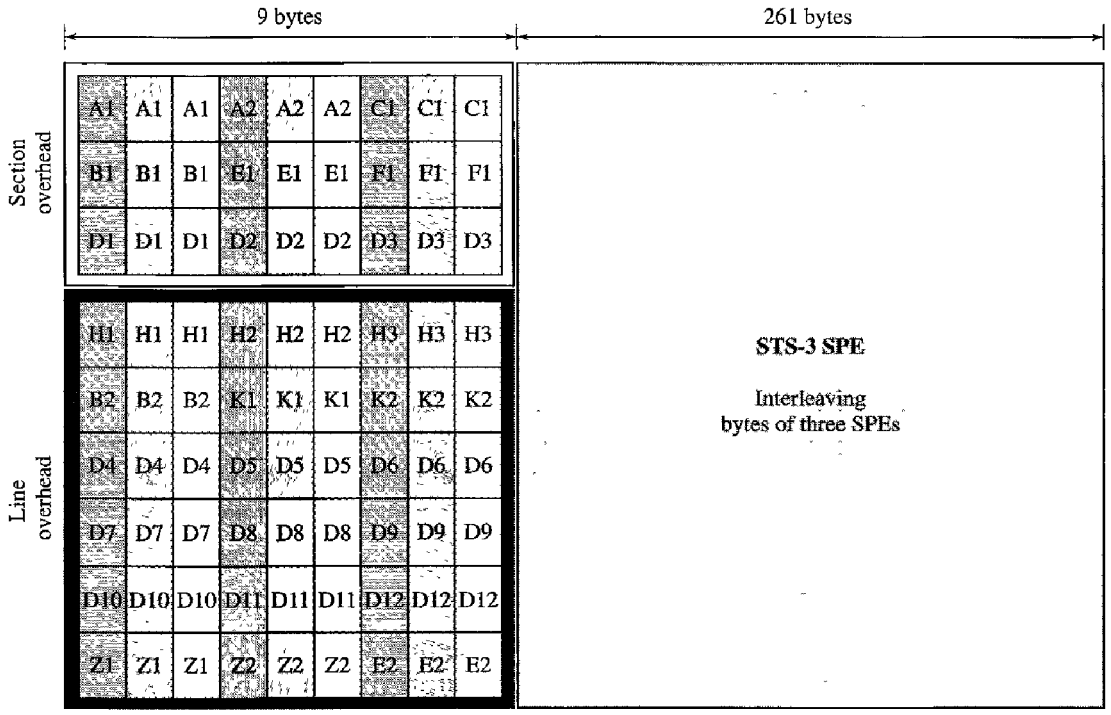
Note that a byte in an STS-1 frame keeps its row position, but it is moved into a different column. The reason is that while all signal frames have the same number of rows (9), the number of columns changes. The number of columns in an STS-*n* signal frame is *n* times the number of columns in an STS-1 frame. One STS-*n* row, therefore, can accommodate all *n* rows in the STS-1 frames.

Byte interleaving also preserves the corresponding section and line overhead as shown in Figure 17.14. As the figure shows, the section overheads from three STS-1 frames are interleaved together to create a section overhead for an STS-1 frame. The same is true for the line overheads. Each channel, however, keeps the corresponding bytes that are used to control that channel. In other words, the sections and lines keep their own control bytes for each multiplexed channel. This interesting feature will allow the use of add/drop multiplexers, as discussed shortly. As the figure shows, there are three A1 bytes, one belonging to each of the three multiplexed signals. There are also three A2 bytes, three B1 bytes, and so on.

Demultiplexing here is easier than in the statistical TDM we discussed in Chapter 6 because the demultiplexer, with no regard to the function of the bytes, removes the first A1 and assigns it to the first STS-1, removes the second A1, and assigns it to second STS-1, and removes the third A1 and assigns it to the third STS-1. In other words, the demultiplexer deals only with the position of the byte, not its function.

What we said about the section and line overheads does not exactly apply to the path overhead. This is because the path overhead is part of the SPE that may have splitted into two STS-1 frames. The byte interleaving, however, is the same for the data section of SPEs.

Figure 17.14 *An STS-3 frame*



The byte interleaving process makes the multiplexing at higher data rates a little bit more complex. How can we multiplex four STS-3 signals into one STS-12 signal? This can be done in two steps: First, the STS-3 signals must be demultiplexed to create 12 STS-1 signals. The 12 STS-1 signals are then multiplexed to create an STS-12 signal.

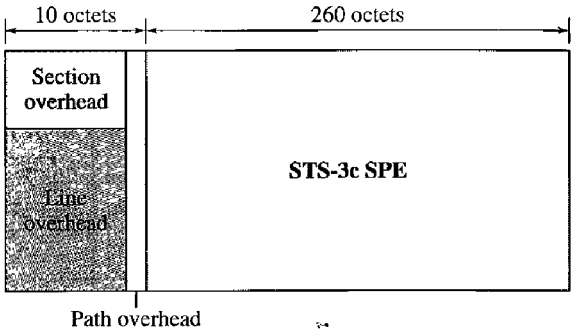
Concatenated Signal

In normal operation of the SONET, an STS-n signal is made of n multiplexed STS-1 signals. Sometimes, we have a signal with a data rate higher than what an STS-1 can carry. In this case, SONET allows us to create an STS-n signal which is not considered as n STS-1 signals; it is one STS-n signal (channel) that cannot be demultiplexed into n STS-1 signals. To specify that the signal cannot be demultiplexed, the suffix c (for concatenated) is added to the name of the signal. For example, STS-3c is a signal that cannot be demultiplexed into three STS-1 signals. However, we need to know that the whole payload in an STS-3c signal is one SPE, which means that we have only one column (9 bytes) of path overhead. The used data in this case occupy 260 columns, as shown in Figure 17.15.

Concatenated Signals Carrying ATM Cells

We will discuss ATM and ATM cells in Chapter 18. An ATM network is a cell network in which each cell has a fixed size of 53 bytes. The SPE of an STS-3c signal can be a carrier of ATM cells. The SPE of an STS-3c can carry $9 \times 260 = 2340$ bytes, which can accommodate approximately 44 ATM cells, each of 53 bytes.

Figure 17.15 *A concatenated STS-3c signal*

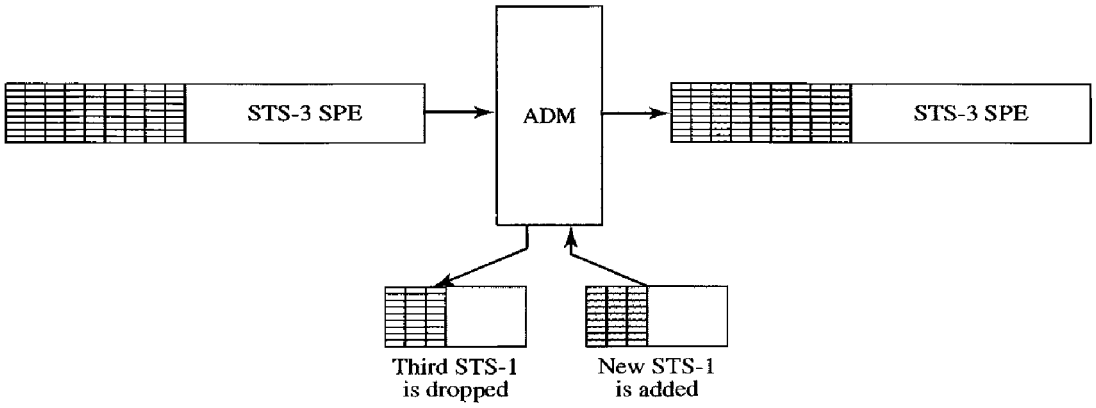


An STS-3c signal can carry 44 ATM cells as its SPE.

Add/Drop Multiplexer

Multiplexing of several STS-1 signals into an STS-*n* signal is done at the STS multiplexer (at the path layer). Demultiplexing of an STS-*n* signal into STS-1 components is done at the STS demultiplexer. In between, however, SONET uses add/drop multiplexers that can replace a signal with another one. We need to know that this is not demultiplexing/multiplexing in the conventional sense. An add/drop multiplexer operates at the line layer. An add/drop multiplexer does not create section, line, or path overhead. It almost acts as a switch; it removes one STS-1 signal and adds another one. The type of signal at the input and output of an add/drop multiplexer is the same (both STS-3 or both STS-12, for example). The add/drop multiplexer (ADM) only removes the corresponding bytes and replaces them with the new bytes (including the bytes in the section and line overhead). Figure 17.16 shows the operation of an ADM.

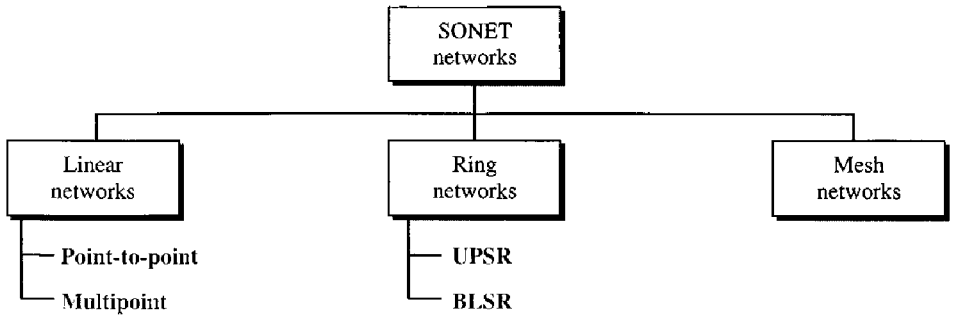
Figure 17.16 *Dropping and adding STS-1 frames in an add/drop multiplexer*



17.5 SONET NETWORKS

Using SONET equipment, we can create a SONET network that can be used as a high-speed backbone carrying loads from other networks such as ATM (Chapter 18) or IP (Chapter 20). We can roughly divide SONET networks into three categories: linear, ring, and mesh networks, as shown in Figure 17.17.

Figure 17.17 *Taxonomy of SONET networks*



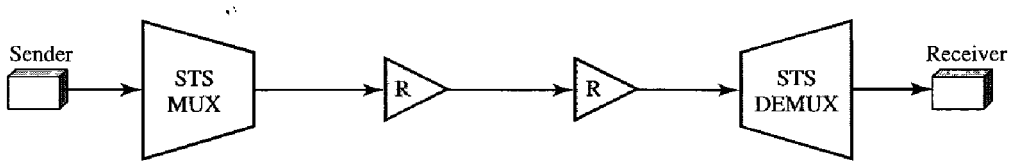
Linear Networks

A linear SONET network can be point-to-point or multipoint.

Point-to-Point Network

A point-to-point network is normally made of an STS multiplexer, an STS demultiplexer, and zero or more regenerators with no add/drop multiplexers, as shown in Figure 17.18. The signal flow can be unidirectional or bidirectional, although Figure 17.18 shows only unidirectional for simplicity.

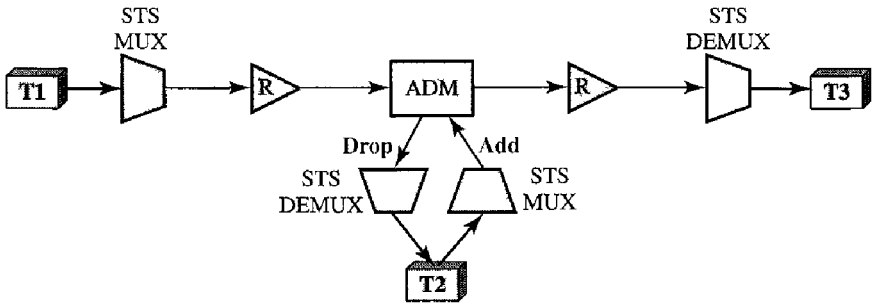
Figure 17.18 *A point-to-point SONET network*



Multipoint Network

A multipoint network uses ADMs to allow the communications between several terminals. An ADM removes the signal belonging to the terminal connected to it and adds the signal transmitted from another terminal. Each terminal can send data to one or more downstream terminals. Figure 17.19 shows a unidirectional scheme in which each terminal can send data only to the downstream terminals, but the a multipoint network can be bidirectional, too.

Figure 17.19 *A multipoint SONET network*

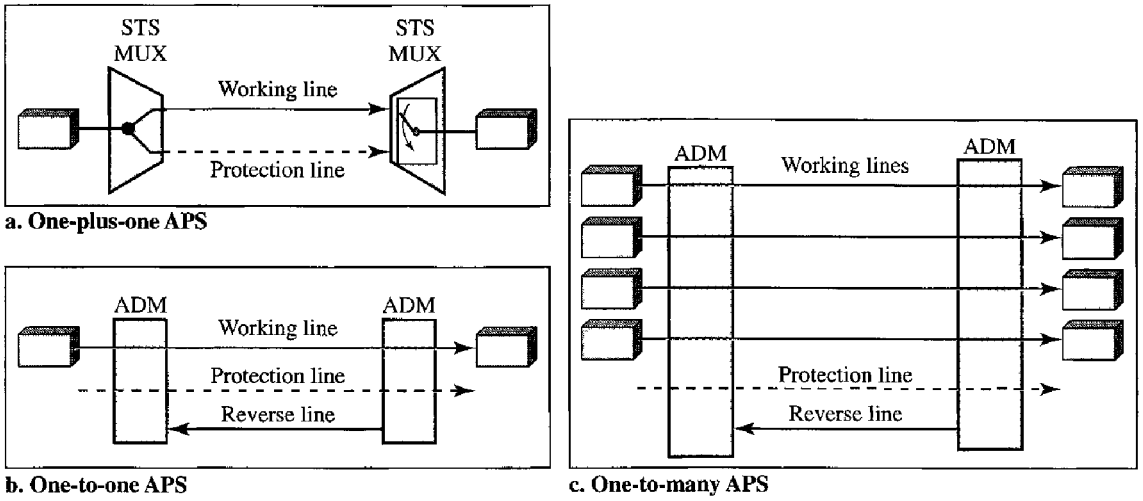


In Figure 17.19, T1 can send data to T2 and T3 simultaneously. T2, however, can send data only to T3. The figure shows a very simple configuration; in normal situations, we have more ADMs and more terminals.

Automatic Protection Switching

To create protection against failure in linear networks, SONET defines **automatic protection switching (APS).** APS in linear networks is defined at the line layer, which means the protection is between two ADMs or a pair of STS multiplexer/demultiplexers. The idea is to provide redundancy; a redundant line (fiber) can be used in case of failure in the main one. The main line is referred to as the work line and the redundant line as the protection line. Three schemes are common for protection in linear channels: one-plus-one, one-to-one, and one-to-many. Figure 17.20 shows all three schemes.

Figure 17.20 *Automatic protection switching in linear networks*



One-Plus-One APS In this scheme, there are normally two lines: one working line and one protection line. Both lines are active all the time. The sending multiplexer

sends the same data on both lines; the receiver multiplexer monitors the line and chooses the one with the better quality. If one of the lines fails, it loses its signal, and, of course, the other line is selected at the receiver. Although, the failure recovery for this scheme is instantaneous, the scheme is inefficient because two times the bandwidth is required. Note that one-plus-one switching is done at the path layer.

One-to-One APS In this scheme, which looks like the one-plus-one scheme, there is also one working line and one protection line. However, the data are normally sent on the working line until it fails. At this time, the receiver, using the reverse channel, informs the sender to use the protection line instead. Obviously, the failure recovery is slower than that of the one-plus-scheme, but this scheme is more efficient because the protection line can be used for data transfer when it is not used to replace the working line. Note that the one-to-one switching is done at the line layer.

One-to-Many APS This scheme is similar to the one-to-one scheme except that there is only one protection line for many working lines. When a failure occurs in one of the working lines, the protection line takes control until the failed line is repaired. It is not as secure as the one-to-one scheme because if more than one working line fails at the same time, the protection line can replace only one of them. Note that one-to-many APS is done at the line layer.

Ring Networks

ADMs make it possible to have SONET ring networks. SONET rings can be used in either a unidirectional or a bidirectional configuration. In each case, we can add extra rings to make the network self-healing, capable of self-recovery from line failure.

Unidirectional Path Switching Ring

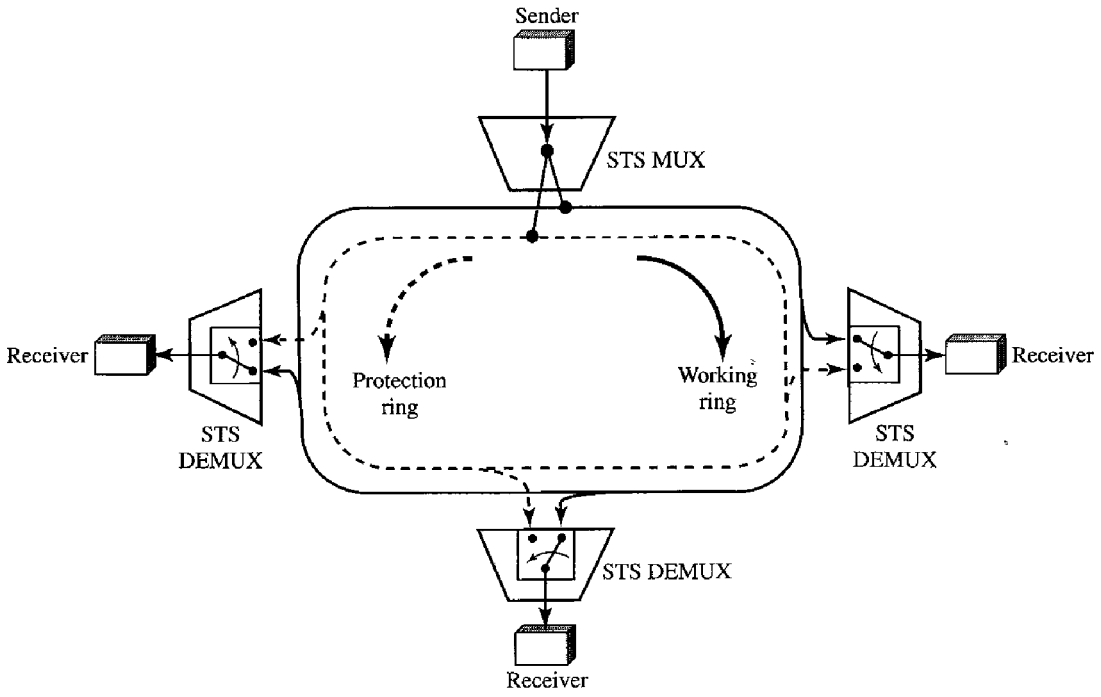
A **unidirectional path switching ring (UPSR)** is a unidirectional network with two rings: one ring used as the working ring and the other as the protection ring. The idea is similar to the one-plus-one APS scheme we discussed in a linear network. The same signal flows through both rings, one clockwise and the other counterclockwise. It is called UPSR because monitoring is done at the path layer. A node receives two copies of the electrical signals at the path layer, compares them, and chooses the one with the better quality. If part of a ring between two ADMs fails, the other ring still can guarantee the continuation of data flow. UPSR, like the one-plus-one scheme, has fast failure recovery, but it is not efficient because we need to have two rings that do the job of one. Half of the bandwidth is wasted. Figure 17.21 shows a UPSR network.

Although we have chosen one sender and three receivers in the figure, there can be many other configurations. The sender uses a two-way connection to send data to both rings simultaneously; the receiver uses selecting switches to select the ring with better signal quality. We have used one STS multiplexer and three STS demultiplexers to emphasize that nodes operate on the path layer.

Bidirectional Line Switching Ring

Another alternative in a SONET ring network is **bidirectional line switching ring (BLSR)**. In this case, communication is bidirectional, which means that we need

Figure 17.21 *A unidirectional path switching ring*



two rings for working lines. We also need two rings for protection lines. This means BLSR uses four rings. The operation, however, is similar to the one-to-one APS scheme. If a working ring in one direction between two nodes fails, the receiving node can use the reverse ring to inform the upstream node in the failed direction to use the protection ring. The network can recover in several different failure situations that we do not discuss here. Note that the discovery of a failure in BLSR is at the line layer, not the path layer. The ADMs find the failure and inform the adjacent nodes to use the protection rings. Figure 17.22 shows a BLSR ring.

Combination of Rings

SONET networks today use a combination of interconnected rings to create services in a wide area. For example, a SONET network may have a regional ring, several local rings, and many site rings to give services to a wide area. These rings can be UPSR, BLSR, or a combination of both. Figure 17.23 shows the idea of such a wide-area ring network.

Mesh Networks

One problem with ring networks is the lack of scalability. When the traffic in a ring increases, we need to upgrade not only the lines, but also the ADMs. In this situation, a mesh network with switches probably give better performance. A switch in a network mesh is called a cross-connect. A cross-connect, like other switches we have seen, has input and output ports. In an input port, the switch takes an OC-*n* signal, changes it to an STS-*n* signal, demultiplexes it into the corresponding STS-1 signals, and sends each

Figure 17.22 *A bidirectional line switching ring*

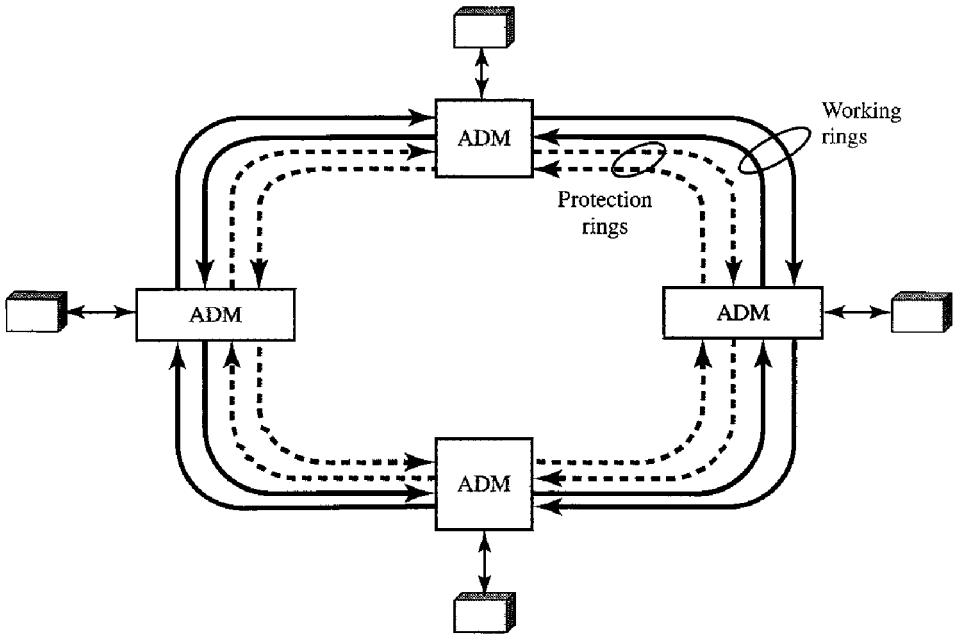
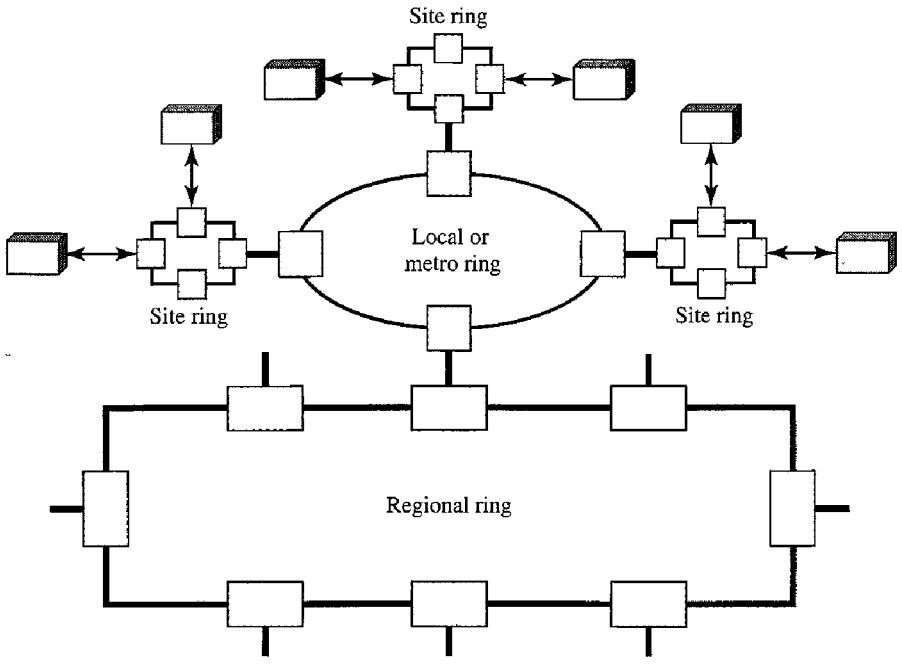
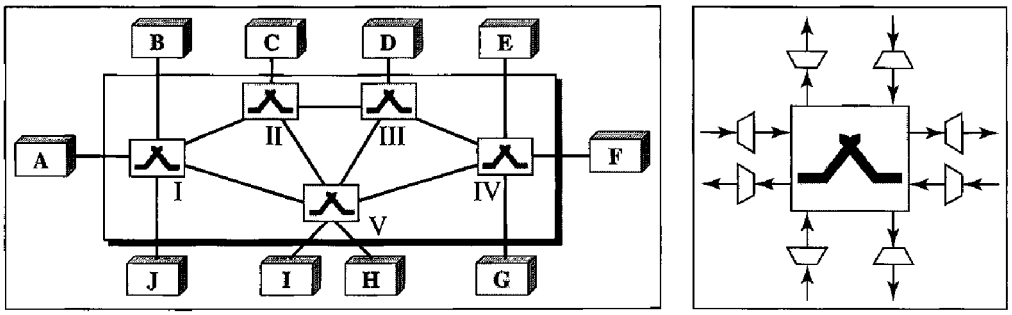


Figure 17.23 *A combination of rings in a SONET network*



STS-1 signal to the appropriate output port. An output port takes STS-1 signals coming from different input ports, multiplexes them into an STS-n signal, and makes an OC-n signal for transmission. Figure 17.24 shows a mesh SONET network, and the structure of a switch.

Figure 17.24 *A mesh SONET network*



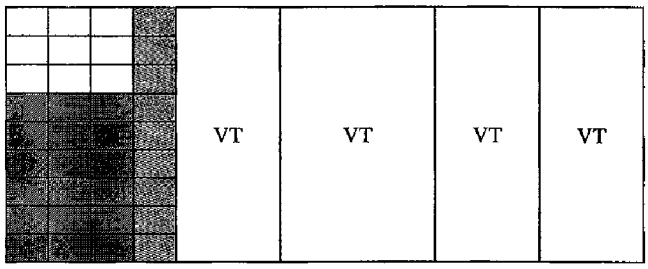
a. SONET mesh network

b. Cross-connect switch

17.6 VIRTUAL TRIBUTARIES

SONET is designed to carry broadband payloads. Current digital hierarchy data rates (DS-1 to DS-3), however, are lower than STS-1. To make SONET backward-compatible with the current hierarchy, its frame design includes a system of **virtual tributaries (VTs)** (see Figure 17.25). A virtual tributary is a partial payload that can be inserted into an STS-1 and combined with other partial payloads to fill out the frame. Instead of using all 86 payload columns of an STS-1 frame for data from one source, we can subdivide the SPE and call each component a VT.

Figure 17.25 *Virtual tributaries*

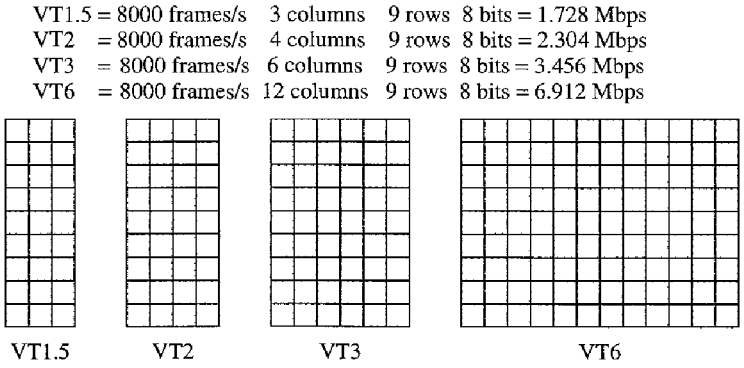


Types of VTs

Four types of VTs have been defined to accommodate existing digital hierarchies (see Figure 17.26). Notice that the number of columns allowed for each type of VT can be determined by doubling the type identification number (VT1.5 gets three columns, VT2 gets four columns, etc.).

- **VT1.5** accommodates the U.S. DS-1 service (1.544 Mbps).
- **VT2** accommodates the European CEPT-1 service (2.048 Mbps).
- **VT3** accommodates the DS-1C service (fractional DS-1, 3.152 Mbps).
- **VT6** accommodates the DS-2 service (6.312 Mbps).

Figure 17.26 *Virtual tributary types*



When two or more tributaries are inserted into a single STS-1 frame, they are interleaved column by column. SONET provides mechanisms for identifying each VT and separating them without demultiplexing the entire stream. Discussion of these mechanisms and the control issues behind them is beyond the scope of this book.

17.7 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

SONET is discussed in Section 2.5 of [Tan03], Section 15.2 of [Kes97], Sections 4.2 and 4.3 of [GW04], Section 8.2 of [Sta04], and Section 5.2 of [WV00].

17.8 KEY TERMS

add/drop multiplexer (ADM)
automatic protection switching (APS)
bidirectional line switching ring (BLSR)
byte interleaving
line
line layer
line overhead (LOH)
optical carrier (OC)
path
path layer
path overhead (POH)
photonic layer
regenerator
section
section layer
section overhead (SOH)
STS demultiplexer
STS multiplexer
Synchronous Digital Hierarchy (SDH)
Synchronous Optical Network (SONET)

synchronous payload envelope (SPE)
synchronous transport module (STM)
synchronous transport signal (STS)
terminal
unidirectional path switching ring (UPSR)
virtual tributary (VT)

17.9 SUMMARY

- Synchronous Optical Network (SONET) is a standard developed by ANSI for fiber-optic networks: Synchronous Digital Hierarchy (SDH) is a similar standard developed by ITU-T.
- SONET has defined a hierarchy of signals called synchronous transport signals (STSs). SDH has defined a similar hierarchy of signals called synchronous transfer modules (STMs).
- An OC-n signal is the optical modulation of an STS-n (or STM-n) signal.
- SONET defines four layers: path, line, section, and photonic.
- SONET is a synchronous TDM system in which all clocks are locked to a master clock.
- A SONET system can use the following equipment:
  1. STS multiplexers
  2. STS demultiplexers
  3. Regenerators
  4. Add/drop multiplexers
  5. Terminals
- SONET sends 8000 frames per second; each frame lasts 125 μs.
- An STS-1 frame is made of 9 rows and 90 columns; an STS-n frame is made of 9 rows and $n \times 90$ columns.
- STSs can be multiplexed to get a new STS with a higher data rate.
- SONET network topologies can be linear, ring, or mesh.
- A linear SONET network can be either point-to-point or multipoint.
- A ring SONET network can be unidirectional or bidirectional.
- To make SONET backward-compatible with the current hierarchy, its frame design includes a system of virtual tributaries (VTs).

17.10 PRACTICE SET

Review Questions

1. What is the relationship between SONET and SDH?
2. What is the relationship between STS and STM?
3. How is an STS multiplexer different from an add/drop multiplexer since both can add signals together?

4. What is the relationship between STS signals and OC signals?
5. What is the purpose of the pointer in the line overhead?
6. Why is SONET called a synchronous network?
7. What is the function of a SONET regenerator?
8. What are the four SONET layers?
9. Discuss the functions of each SONET layer.
10. What is a virtual tributary?

Exercises

11. What are the user data rates of STS-3, STS-9, and STS-12?
12. Show how STS-9's can be multiplexed to create an STS-36. Is there any extra overhead involved in this type of multiplexing?
13. A stream of data is being carried by STS-1 frames. If the data rate of the stream is 49.540 Mbps, how many STS-1 frames per second must let their H3 bytes carry data?
14. A stream of data is being carried by STS-1 frames. If the data rate of the stream is 49.530 Mbps, how many frames per second should leave one empty byte after the H3 byte?
15. Table 17.2 shows that the overhead bytes can be categorized as A, B, C, D, E, F, G, H, J, K, and Z bytes.
    a. Why are there no A bytes in the LOH or POH?
    b. Why are there no C bytes in the LOH?
    c. Why are there no D bytes in the POH?
    d. Why are there no E bytes in the LOH or POH?
    e. Why are there no F bytes in the LOH or POH?
    f. Why are there no G bytes in the SOH or LOH?
    g. Why are there no H bytes in the SOH?
    h. Why are there no J bytes in the SOH or LOH?
    i. Why are there no K bytes in the SOH or POH?
    j. Why are there no Z bytes in the SOH?
16. Why are B bytes present in all three headers?

Down

CHAPTER 18

Virtual-Circuit Networks: Frame Relay and ATM

In Chapter 8, we discussed switching techniques. We said that there are three types of switching: circuit switching, packet switching, and message switching. We also mentioned that packet switching can use two approaches: the virtual-circuit approach and the datagram approach.

In this chapter, we show how the virtual-circuit approach can be used in wide-area networks. Two common WAN technologies use virtual-circuit switching. Frame Relay is a relatively high-speed protocol that can provide some services not available in other WAN technologies such as DSL, cable TV, and T lines. ATM, as a high-speed protocol, can be the superhighway of communication when it deploys physical layer carriers such as SONET.

We first discuss Frame Relay. We then discuss ATM in greater detail. Finally, we show how ATM technology, which was originally designed as a WAN technology, can also be used in LAN technology, ATM LANs.

18.1 FRAME RELAY

Frame Relay is a virtual-circuit wide-area network that was designed in response to demands for a new type of WAN in the late 1980s and early 1990s.

1. Prior to Frame Relay, some organizations were using a virtual-circuit switching network called **X.25** that performed switching at the network layer. For example, the Internet, which needs wide-area networks to carry its packets from one place to another, used X.25. And X.25 is still being used by the Internet, but it is being replaced by other WANs. However, X.25 has several drawbacks:
   a. X.25 has a low 64-kbps data rate. By the 1990s, there was a need for higher-data-rate WANs.
   b. X.25 has extensive flow and error control at both the data link layer and the network layer. This was so because X.25 was designed in the 1970s, when the available transmission media were more prone to errors. Flow and error control at both layers create a large overhead and slow down transmissions. X.25 requires acknowledgments for both data link layer frames and network layer packets that are sent between nodes and between source and destination.

c. Originally X.25 was designed for private use, not for the Internet. X.25 has its own network layer. This means that the user's data are encapsulated in the network layer packets of X.25. The Internet, however, has its own network layer, which means if the Internet wants to use X.25, the Internet must deliver its network layer packet, called a datagram, to X.25 for encapsulation in the X.25 packet. This doubles the overhead.

2. Disappointed with X.25, some organizations started their own private WAN by leasing T-1 or T-3 lines from public service providers. This approach also has some drawbacks.
   a. If an organization has n branches spread over an area, it needs $n(n-1)/2$ T-1 or T-3 lines. The organization pays for all these lines although it may use the lines only 10 percent of the time. This can be very costly.
   b. The services provided by T-1 and T-3 lines assume that the user has fixed-rate data all the time. For example, a T-1 line is designed for a user who wants to use the line at a consistent 1.544 Mbps. This type of service is not suitable for the many users today that need to send **bursty data.** For example, a user may want to send data at 6 Mbps for 2 s, 0 Mbps (nothing) for 7 s, and 3.44 Mbps for 1 s for a total of 15.44 Mbits during a period of 10 s. Although the average data rate is still 1.544 Mbps, the T-1 line cannot accept this type of demand because it is designed for fixed-rate data, not bursty data. Bursty data require what is called **bandwidth on demand.** The user needs different bandwidth allocations at different times.

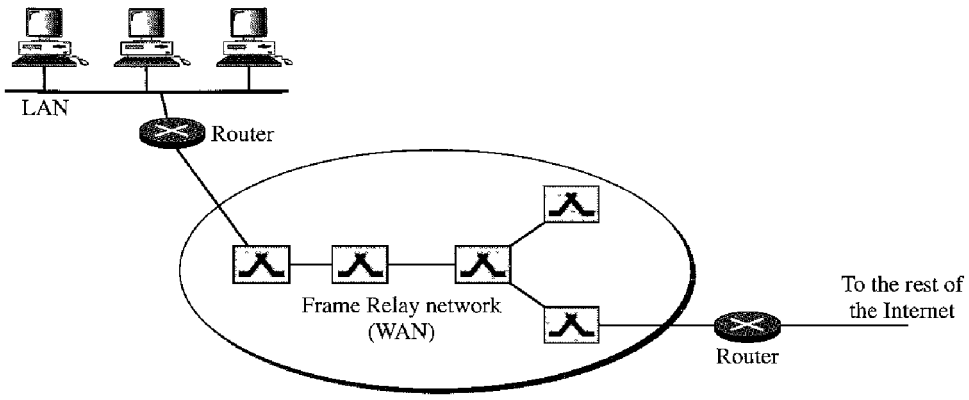
In response to the above drawbacks, Frame Relay was designed. Frame Relay is a wide-area network with the following features:

1. Frame Relay operates at a higher speed (1.544 Mbps and recently 44.376 Mbps). This means that it can easily be used instead of a mesh of T-1 or T-3 lines.
2. Frame Relay operates in just the physical and data link layers. This means it can easily be used as a backbone network to provide services to protocols that already have a network layer protocol, such as the Internet.
3. Frame Relay allows bursty data.
4. Frame Relay allows a frame size of 9000 bytes, which can accommodate all local-area network frame sizes.
5. Frame Relay is less expensive than other traditional WANs.
6. Frame Relay has error detection at the data link layer only. There is no flow control or error control. There is not even a retransmission policy if a frame is damaged; it is silently dropped. Frame Relay was designed in this way to provide fast transmission capability for more reliable media and for those protocols that have flow and error control at the higher layers.

Architecture

Frame Relay provides permanent virtual circuits and switched virtual circuits. Figure 18.1 shows an example of a Frame Relay network connected to the Internet. The routers are used, as we will see in Chapter 22, to connect LANs and WANs in the Internet. In the figure, the Frame Relay WAN is used as one link in the global Internet.

Figure 18.1 *Frame Relay network*



Virtual Circuits

Frame Relay is a virtual circuit network. A virtual circuit in Frame Relay is identified by a number called a **data link connection identifier (DLCI).**

VCIs in Frame Relay are called DLCIs.

Permanent Versus Switched Virtual Circuits

A source and a destination may choose to have a **permanent virtual circuit (PVC).** In this case, the connection setup is simple. The corresponding table entry is recorded for all switches by the administrator (remotely and electronically, of course). An outgoing DLCI is given to the source, and an incoming DLCI is given to the destination.

PVC connections have two drawbacks. First, they are costly because two parties pay for the connection all the time even when it is not in use. Second, a connection is created from one source to one single destination. If a source needs connections with several destinations, it needs a PVC for each connection. An alternate approach is the **switched virtual circuit (SVC).** The SVC creates a temporary, short connection that exists only when data are being transferred between source and destination. An SVC requires establishing and terminating phases as discussed in Chapter 8.

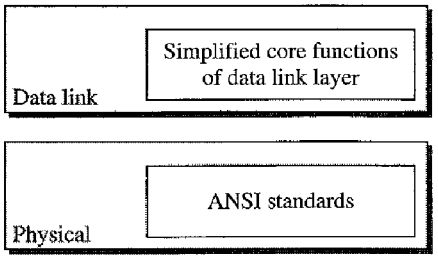
Switches

Each switch in a Frame Relay network has a table to route frames. The table matches an incoming port–DLCI combination with an outgoing port–DLCI combination as we described for general virtual-circuit networks in Chapter 8. The only difference is that VCIs are replaced by DLCIs.

Frame Relay Layers

Figure 18.2 shows the Frame Relay layers. Frame Relay has only physical and data link layers.

Figure 18.2 *Frame Relay layers*



Frame Relay operates only at the physical and data link layers.

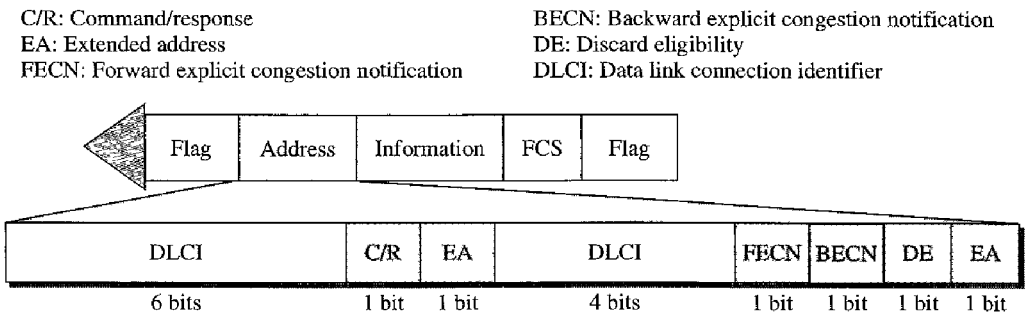
Physical Layer

No specific protocol is defined for the physical layer in Frame Relay. Instead, it is left to the implementer to use whatever is available. Frame Relay supports any of the protocols recognized by ANSI.

Data Link Layer

At the data link layer, Frame Relay uses a simple protocol that does not support flow or error control. It only has an error detection mechanism. Figure 18.3 shows the format of a Frame Relay frame.The address field defines the DLCI as well as some bits used to control congestion.

Figure 18.3 *Frame Relay frame*



The descriptions of the fields are as follows:

- **Address (DLCI) field.** The first 6 bits of the first byte makes up the first part of the DLCI. The second part of the DLCI uses the first 4 bits of the second byte. These bits are part of the 10-bit data link connection identifier defined by the standard. We will discuss extended addressing at the end of this section.

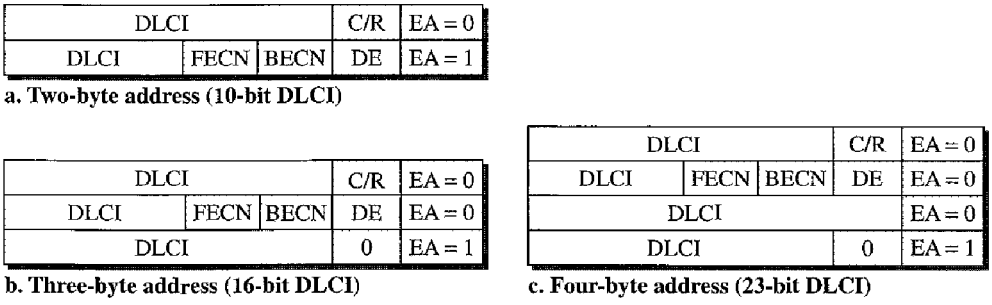
- **Command/response (C/R).** The command/response (C/R) bit is provided to allow upper layers to identify a frame as either a command or a response. It is not used by the Frame Relay protocol.
- **Extended address (EA).** The extended address (EA) bit indicates whether the current byte is the final byte of the address. An EA of 0 means that another address byte is to follow (extended addressing is discussed later). An EA of 1 means that the current byte is the final one.
- **Forward explicit congestion notification (FECN).** The **forward explicit congestion notification (FECN)** bit can be set by any switch to indicate that traffic is congested. This bit informs the destination that congestion has occurred. In this way, the destination knows that it should expect delay or a loss of packets. We will discuss the use of this bit when we discuss congestion control in Chapter 24.
- **Backward explicit congestion notification (BECN).** The **backward explicit congestion notification (BECN)** bit is set (in frames that travel in the other direction) to indicate a congestion problem in the network. This bit informs the sender that congestion has occurred. In this way, the source knows it needs to slow down to prevent the loss of packets. We will discuss the use of this bit when we discuss congestion control in Chapter 24.
- **Discard eligibility (DE).** The **discard eligibility (DE)** bit indicates the priority level of the frame. In emergency situations, switches may have to discard frames to relieve bottlenecks and keep the network from collapsing due to overload. When set (DE 1), this bit tells the network to discard this frame if there is congestion. This bit can be set either by the sender of the frames (user) or by any switch in the network.

Frame Relay does not provide flow or error control; they must be provided by the upper-layer protocols.

Extended Address

To increase the range of DLCIs, the Frame Relay address has been extended from the original 2-byte address to 3- or 4-byte addresses. Figure 18.4 shows the different addresses. Note that the EA field defines the number of bytes; it is 1 in the last byte of the address, and it is 0 in the other bytes. Note that in the 3- and 4-byte formats, the bit before the last bit is set to 0.

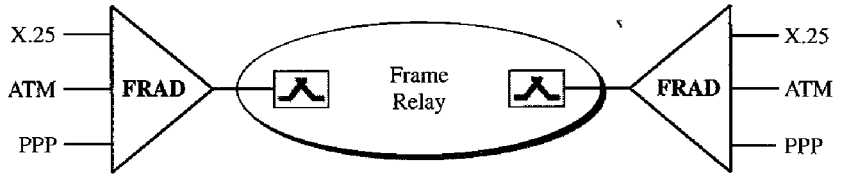
Figure 18.4 *Three address formats*



FRADs

To handle frames arriving from other protocols, Frame Relay uses a device called a **Frame Relay assembler/disassembler (FRAD).** A FRAD assembles and disassembles frames coming from other protocols to allow them to be carried by Frame Relay frames. A FRAD can be implemented as a separate device or as part of a switch. Figure 18.5 shows two FRADs connected to a Frame Relay network.

Figure 18.5 *FRAD*



VOFR

Frame Relay networks offer an option called **Voice Over Frame Relay (VOFR)** that sends voice through the network. Voice is digitized using PCM and then compressed. The result is sent as data frames over the network. This feature allows the inexpensive sending of voice over long distances. However, note that the quality of voice is not as good as voice over a circuit-switched network such as the telephone network. Also, the varying delay mentioned earlier sometimes corrupts real-time voice.

LMI

Frame Relay was originally designed to provide PVC connections. There was not, therefore, a provision for controlling or managing interfaces. **Local Management Information (LMI)** is a protocol added recently to the Frame Relay protocol to provide more management features. In particular, LMI can provide

- A keep-alive mechanism to check if data are flowing.
- A multicast mechanism to allow a local end system to send frames to more than one remote end system.
- A mechanism to allow an end system to check the status of a switch (e.g., to see if the switch is congested).

Congestion Control and Quality of Service

One of the nice features of Frame Relay is that it provides **congestion control** and **quality of service (QoS).** We have not discussed these features yet. In Chapter 24, we introduce these two important aspects of networking and discuss how they are implemented in Frame Relay and some other networks.

18.2 ATM

Asynchronous Transfer Mode (ATM) is the **cell relay** protocol designed by the ATM Forum and adopted by the ITU-T. The combination of ATM and SONET will allow high-speed interconnection of all the world's networks. In fact, ATM can be thought of as the "highway" of the information superhighway.

Design Goals

Among the challenges faced by the designers of ATM, six stand out.

1. Foremost is the need for a transmission system to optimize the use of high-data-rate transmission media, in particular optical fiber. In addition to offering large bandwidths, newer transmission media and equipment are dramatically less susceptible to noise degradation. A technology is needed to take advantage of both factors and thereby maximize data rates.
2. The system must interface with existing systems and provide wide-area interconnectivity between them without lowering their effectiveness or requiring their replacement.
3. The design must be implemented inexpensively so that cost would not be a barrier to adoption. If ATM is to become the backbone of international communications, as intended, it must be available at low cost to every user who wants it.
4. The new system must be able to work with and support the existing telecommunications hierarchies (local loops, local providers, long-distance carriers, and so on).
5. The new system must be connection-oriented to ensure accurate and predictable delivery.
6. Last but not least, one objective is to move as many of the functions to hardware as possible (for speed) and eliminate as many software functions as possible (again for speed).

Problems

Before we discuss the solutions to these design requirements, it is useful to examine some of the problems associated with existing systems.

Frame Networks

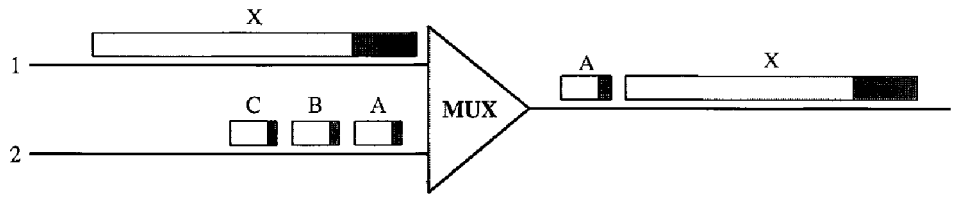
Before ATM, data communications at the data link layer had been based on frame switching and frame networks. Different protocols use frames of varying size and intricacy. As networks become more complex, the information that must be carried in the header becomes more extensive. The result is larger and larger headers relative to the size of the data unit. In response, some protocols have enlarged the size of the data unit to make header use more efficient (sending more data with the same size header). Unfortunately, large data fields create waste. If there is not much information to transmit, much of the field goes unused. To improve utilization, some protocols provide variable frame sizes to users.

Mixed Network Traffic

As you can imagine, the variety of frame sizes makes traffic unpredictable. Switches, multiplexers, and routers must incorporate elaborate software systems to manage the various sizes of frames. A great deal of header information must be read, and each bit counted and evaluated to ensure the integrity of every frame. Internetworking among the different frame networks is slow and expensive at best, and impossible at worst.

Another problem is that of providing consistent data rate delivery when frame sizes are unpredictable and can vary so dramatically. To get the most out of broadband technology, traffic must be time-division multiplexed onto shared paths. Imagine the results of multiplexing frames from two networks with different requirements (and frame designs) onto one link (see Figure 18.6). What happens when line 1 uses large frames (usually data frames) while line 2 uses very small frames (the norm for audio and video information)?

Figure 18.6 *Multiplexing using different frame sizes*



If line 1's gigantic frame X arrives at the multiplexer even a moment earlier than line 2's frames, the multiplexer puts frame X onto the new path first. After all, even if line 2's frames have priority, the multiplexer has no way of knowing to wait for them and so processes the frame that has arrived. Frame A must therefore wait for the entire X bit stream to move into place before it can follow. The sheer size of X creates an unfair delay for frame A. The same imbalance can affect all the frames from line 2.

Because audio and video frames ordinarily are small, mixing them with conventional data traffic often creates unacceptable delays of this type and makes shared frame links unusable for audio and video information. Traffic must travel over different paths, in much the same way that automobile and train traffic does. But to fully utilize broad bandwidth links, we need to be able to send all kinds of traffic over the same links.

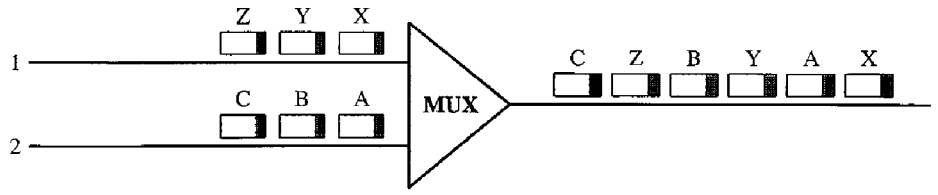
Cell Networks

Many of the problems associated with frame internetworking are solved by adopting a concept called cell networking. A cell is a small data unit of fixed size. In a **cell network,** which uses the **cell** as the basic unit of data exchange, all data are loaded into identical cells that can be transmitted with complete predictability and uniformity. As frames of different sizes and formats reach the cell network from a tributary network, they are split into multiple small data units of equal length and are loaded into cells. The cells are then multiplexed with other cells and routed through the cell network. Because each cell is the same size and all are small, the problems associated with multiplexing different-sized frames are avoided.

A cell network uses the cell as the basic unit of data exchange. A cell is defined as a small, fixed-size block of information.

Figure 18.7 shows the multiplexer from Figure 18.6 with the two lines sending cells instead of frames. Frame X has been segmented into three cells: X, Y, and Z. Only the first cell from line 1 gets put on the link before the first cell from line 2. The cells from the two lines are interleaved so that none suffers a long delay.

Figure 18.7 *Multiplexing using cells*



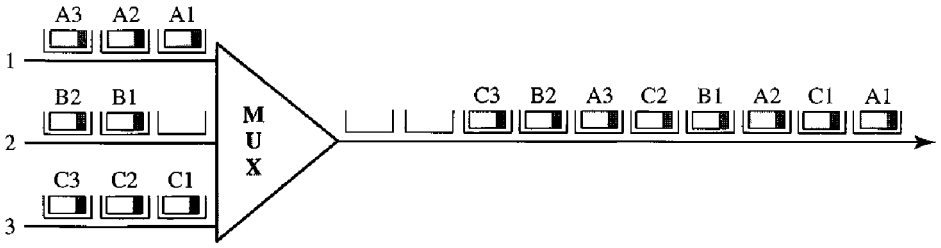
A second point in this same scenario is that the high speed of the links coupled with the small size of the cells means that, despite interleaving, cells from each line arrive at their respective destinations in an approximation of a continuous stream (much as a movie appears to your brain to be continuous action when in fact it is really a series of separate, still photographs). In this way, a cell network can handle real-time transmissions, such as a phone call, without the parties being aware of the segmentation or multiplexing at all.

Asynchronous TDM

ATM uses asynchronous time-division multiplexing—that is why it is called Asynchronous Transfer Mode—to multiplex cells coming from different channels. It uses fixed-size slots (size of a cell). ATM multiplexers fill a slot with a cell from any input channel that has a cell; the slot is empty if none of the channels has a cell to send.

Figure 18.8 shows how cells from three inputs are multiplexed. At the first tick of the clock, channel 2 has no cell (empty input slot), so the multiplexer fills the slot with a cell from the third channel. When all the cells from all the channels are multiplexed, the output slots are empty.

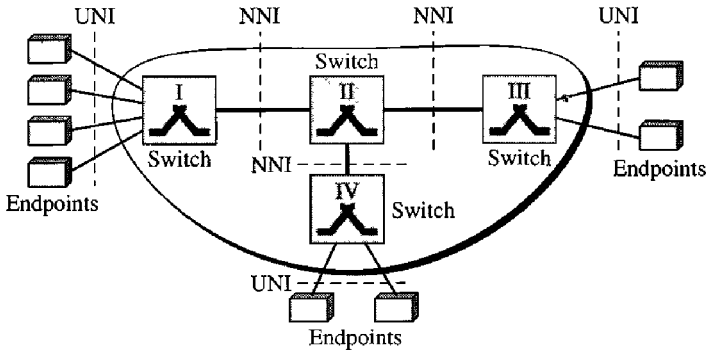
Figure 18.8 *ATM multiplexing*



Architecture

ATM is a cell-switched network. The user access devices, called the endpoints, are connected through a **user-to-network interface (UNI)** to the switches inside the network. The switches are connected through **network-to-network interfaces (NNIs).** Figure 18.9 shows an example of an ATM network.

Figure 18.9 *Architecture of an ATM network*



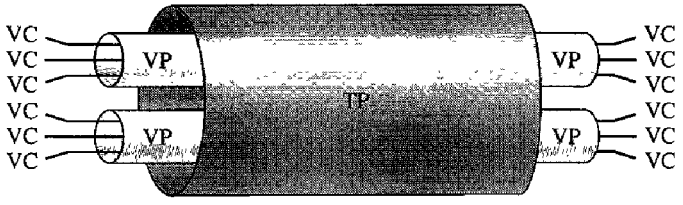
Virtual Connection

Connection between two endpoints is accomplished through transmission paths (TPs), virtual paths (VPs), and virtual circuits (VCs). A **transmission path (TP)** is the physical connection (wire, cable, satellite, and so on) between an endpoint and a switch or between two switches. Think of two switches as two cities. A transmission path is the set of all highways that directly connect the two cities.

A transmission path is divided into several virtual paths. A **virtual path (VP)** provides a connection or a set of connections between two switches. Think of a virtual path as a highway that connects two cities. Each highway is a virtual path; the set of all highways is the transmission path.

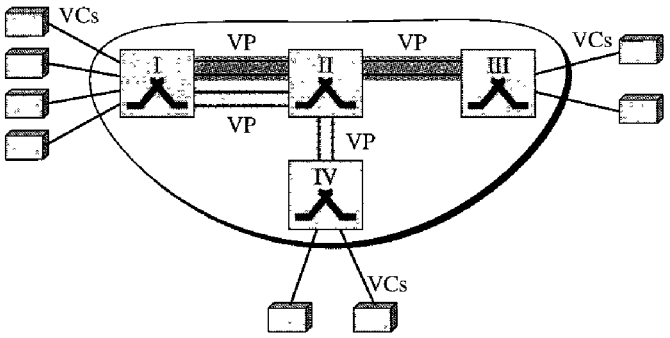
Cell networks are based on **virtual circuits (VCs).** All cells belonging to a single message follow the same virtual circuit and remain in their original order until they reach their destination. Think of a virtual circuit as the lanes of a highway (virtual path). Figure 18.10 shows the relationship between a transmission path (a physical connection), virtual paths (a combination of virtual circuits that are bundled together because parts of their paths are the same), and virtual circuits that logically connect two points.

Figure 18.10 *TP, VPs, and VCs*



To better understand the concept of VPs and VCs, look at Figure 18.11. In this figure, eight endpoints are communicating using four VCs. However, the first two VCs seem to share the same virtual path from switch I to switch III, so it is reasonable to bundle these two VCs together to form one VP. On the other hand, it is clear that the other two VCs share the same path from switch I to switch IV, so it is also reasonable to combine them to form one VP.

Figure 18.11 *Example of VPs and VCs*



Identifiers In a virtual circuit network, to route data from one endpoint to another, the virtual connections need to be identified. For this purpose, the designers of ATM created a hierarchical identifier with two levels: a **virtual path identifier (VPI)** and a **virtual-circuit identifier (VCI).** The VPI defines the specific VP, and the VCI defines a particular VC inside the VP. The VPI is the same for all virtual connections that are bundled (logically) into one VP.

Note that a virtual connection is defined by a pair of numbers: the VPI and the VCI.

Figure 18.12 shows the VPIs and VCIs for a transmission path. The rationale for dividing an identifier into two parts will become clear when we discuss routing in an ATM network.

The lengths of the VPIs for UNIs and NNIs are different. In a UNI, the VPI is 8 bits, whereas in an NNI, the VPI is 12 bits. The length of the VCI is the same in both interfaces (16 bits). We therefore can say that a virtual connection is identified by 24 bits in a UNI and by 28 bits in an NNI (see Figure 18.13).

The whole idea behind dividing a virtual circuit identifier into two parts is to allow hierarchical routing. Most of the switches in a typical ATM network are routed using VPIs. The switches at the boundaries of the network, those that interact directly with the endpoint devices, use both VPIs and VCIs.

Cells

The basic data unit in an ATM network is called a cell. A cell is only 53 bytes long with 5 bytes allocated to the header and 48 bytes carrying the payload (user data may be less

Figure 18.12 *Connection identifiers*

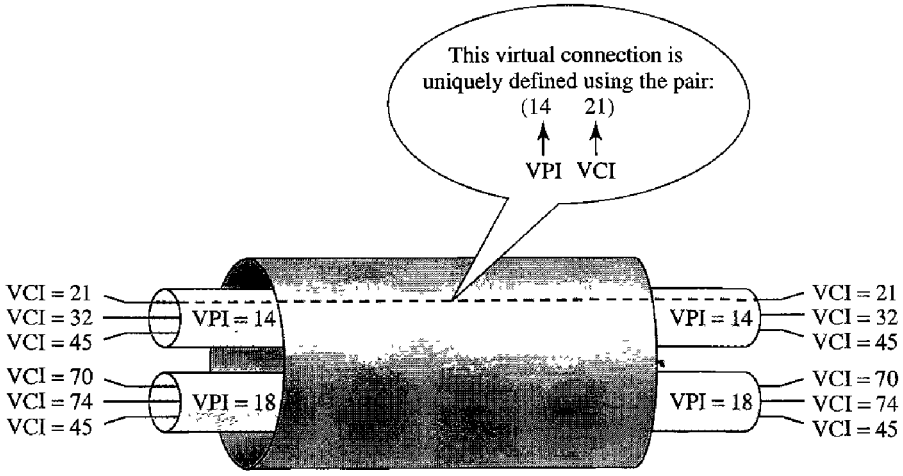
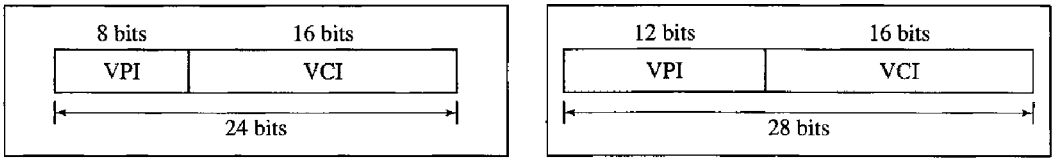


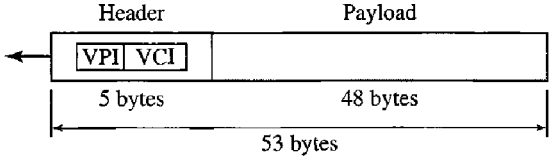
Figure 18.13 *Virtual connection identifiers in UNIs and NNIs*



a. VPI and VCI in a UNI **b. VPI and VCI in an NNI**

than 48 bytes). We will study in detail the fields of a cell, but for the moment it suffices to say that most of the header is occupied by the VPI and VCI that define the virtual connection through which a cell should travel from an endpoint to a switch or from a switch to another switch. Figure 18.14 shows the cell structure.

Figure 18.14 *An ATM cell*



Connection Establishment and Release

Like Frame Relay, ATM uses two types of connections: PVC and SVC.

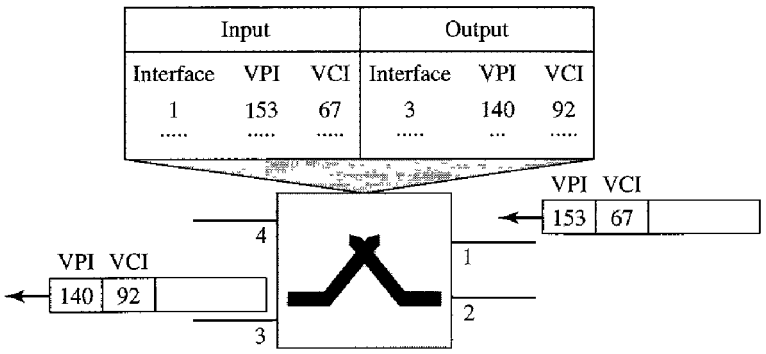
PVC A permanent virtual-circuit connection is established between two endpoints by the network provider. The VPIs and VCIs are defined for the permanent connections, and the values are entered for the tables of each switch.

SVC In a switched virtual-circuit connection, each time an endpoint wants to make a connection with another endpoint, a new virtual circuit must be established. ATM cannot do the job by itself, but needs the network layer addresses and the services of another protocol (such as IP). The signaling mechanism of this other protocol makes a connection request by using the network layer addresses of the two endpoints. The actual mechanism depends on the network layer protocol.

Switching

ATM uses switches to route the cell from a source endpoint to the destination endpoint. A switch routes the cell using both the VPIs and the VCIs. The routing requires the whole identifier. Figure 18.15 shows how a VPC switch routes the cell. A cell with a VPI of 153 and VCI of 67 arrives at switch interface (port) 1. The switch checks its switching table, which stores six pieces of information per row: arrival interface number, incoming VPI, incoming VCI, corresponding outgoing interface number, the new VPI, and the new VCI. The switch finds the entry with the interface 1, VPI 153, and VCI 67 and discovers that the combination corresponds to output interface 3, VPI 140, and VCI 92. It changes the VPI and VCI in the header to 140 and 92, respectively, and sends the cell out through interface 3.

Figure 18.15 *Routing with a switch*



Switching Fabric

The switching technology has created many interesting features to increase the speed of switches to handle data. We discussed switching fabrics in Chapter 8.

ATM Layers

The ATM standard defines three layers. They are, from top to bottom, the application adaptation layer, the ATM layer, and the physical layer (see Figure 18.16).

The endpoints use all three layers while the switches use only the two bottom layers (see Figure 18.17).

Physical Layer

Like Ethernet and wireless LANs, ATM cells can be carried by any physical layer carrier.

Figure 18.16 *ATM layers*

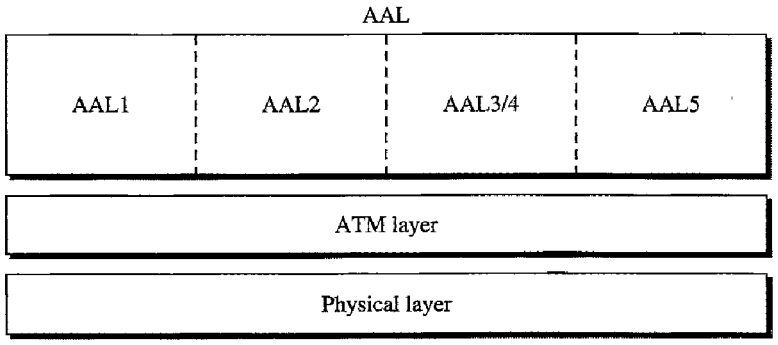
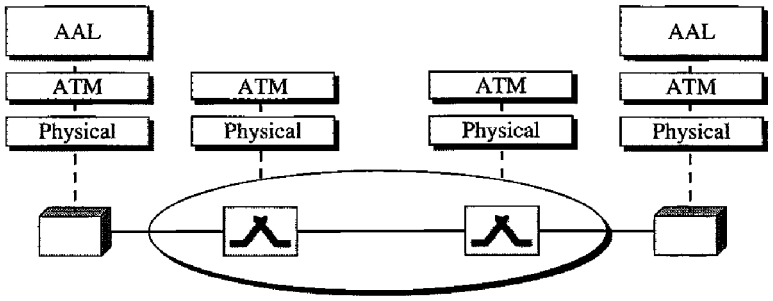


Figure 18.17 *ATM layers in endpoint devices and switches*



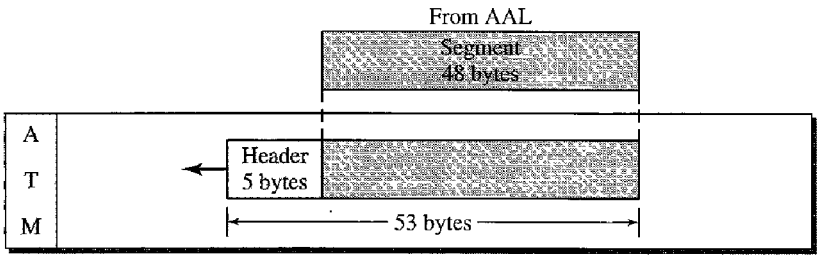
SONET The original design of ATM was based on *SONET* (see Chapter 17) as the physical layer carrier. SONET is preferred for two reasons. First, the high data rate of SONET's carrier reflects the design and philosophy of ATM. Second, in using SONET, the boundaries of cells can be clearly defined. As we saw in Chapter 17, SONET specifies the use of a pointer to define the beginning of a payload. If the beginning of the first ATM cell is defined, the rest of the cells in the same payload can easily be identified because there are no gaps between cells. Just count 53 bytes ahead to find the next cell.

Other Physical Technologies ATM does not limit the physical layer to SONET. Other technologies, even wireless, may be used. However, the problem of cell boundaries must be solved. One solution is for the receiver to guess the end of the cell and apply the CRC to the 5-byte header. If there is no error, the end of the cell is found, with a high probability, correctly. Count 52 bytes back to find the beginning of the cell.

ATM Layer

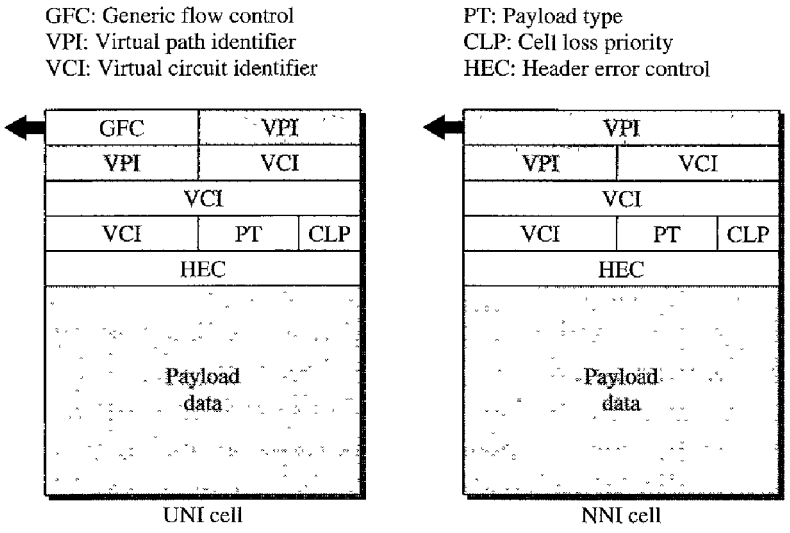
The **ATM layer** provides routing, traffic management, switching, and multiplexing services. It processes outgoing traffic by accepting 48-byte segments from the AAL sublayers and transforming them into 53-byte cells by the addition of a 5-byte header (see Figure 18.18).

Figure 18.18 *ATM layer*



Header Format ATM uses two formats for this header, one for user-to-network interface (UNI) cells and another for network-to-network interface (NNI) cells. Figure 18.19 shows these headers in the byte-by-byte format preferred by the ITU-T (each row represents a byte).

Figure 18.19 *ATM headers*



- **Generic flow control (GFC).** The 4-bit GFC field provides flow control at the UNI level. The ITU-T has determined that this level of flow control is not necessary at the NNI level. In the NNI header, therefore, these bits are added to the VPI. The longer VPI allows more virtual paths to be defined at the NNI level. The format for this additional VPI has not yet been determined.
- **Virtual path identifier (VPI).** The VPI is an 8-bit field in a UNI cell and a 12-bit field in an NNI cell (see above).
- **Virtual circuit identifier (VCI).** The VCI is a 16-bit field in both frames.
- **Payload type (PT).** In the 3-bit PT field, the first bit defines the payload as user data or managerial information. The interpretation of the last 2 bits depends on the first bit.

- **Cell loss priority (CLP).** The 1-bit CLP field is provided for congestion control. A cell with its CLP bit set to 1 must be retained as long as there are cells with a CLP of 0. We discuss congestion control and quality of service in an ATM network in Chapter 24.
- **Header error correction (HEC).** The HEC is a code computed for the first 4 bytes of the header. It is a CRC with the divisor $x^8 + x^2 + x + 1$ that is used to correct single-bit errors and a large class of multiple-bit errors.

Application Adaptation Layer

The **application adaptation layer (AAL)** was designed to enable two ATM concepts. First, ATM must accept any type of payload, both data frames and streams of bits. A data frame can come from an upper-layer protocol that creates a clearly defined frame to be sent to a carrier network such as ATM. A good example is the Internet. ATM must also carry multimedia payload. It can accept continuous bit streams and break them into chunks to be encapsulated into a cell at the ATM layer. AAL uses two sublayers to accomplish these tasks.

Whether the data are a data frame or a stream of bits, the payload must be segmented into 48-byte segments to be carried by a cell. At the destination, these segments need to be reassembled to recreate the original payload. The AAL defines a sublayer, called a **segmentation and reassembly (SAR)** sublayer, to do so. Segmentation is at the source; reassembly, at the destination.

Before data are segmented by SAR, they must be prepared to guarantee the integrity of the data. This is done by a sublayer called the **convergence sublayer (CS).**

ATM defines four versions of the AAL: **AAL1, AAL2, AAL3/4,** and **AAL5.** Although we discuss all these versions, we need to inform the reader that the common versions today are AAL1 and AAL5. The first is used in streaming audio and video communication; the second, in data communications.

AAL1 AAL1 supports applications that transfer information at constant bit rates, such as video and voice. It allows ATM to connect existing digital telephone networks such as voice channels and T lines. Figure 18.20 shows how a bit stream of data is chopped into 47-byte chunks and encapsulated in cells.

The CS sublayer divides the bit stream into 47-byte segments and passes them to the SAR sublayer below. Note that the CS sublayer does not add a header.

The SAR sublayer adds 1 byte of header and passes the 48-byte segment to the ATM layer. The header has two fields:

- **Sequence number (SN).** This 4-bit field defines a sequence number to order the bits. The first bit is sometimes used for timing, which leaves 3 bits for sequencing (modulo 8).
- **Sequence number protection (SNP).** The second 4-bit field protects the first field. The first 3 bits automatically correct the SN field. The last bit is a parity bit that detects error over all 8 bits.

AAL2 Originally AAL2 was intended to support a variable-data-rate bit stream, but it has been redesigned. It is now used for low-bit-rate traffic and short-frame traffic such as audio (compressed or uncompressed), video, or fax. A good example of AAL2 use is in mobile telephony. AAL2 allows the multiplexing of short frames into one cell.

Figure 18.20 *AAL1*

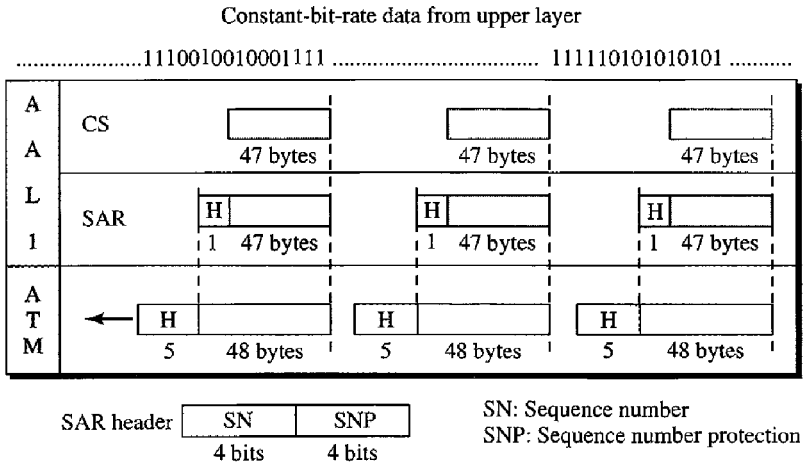
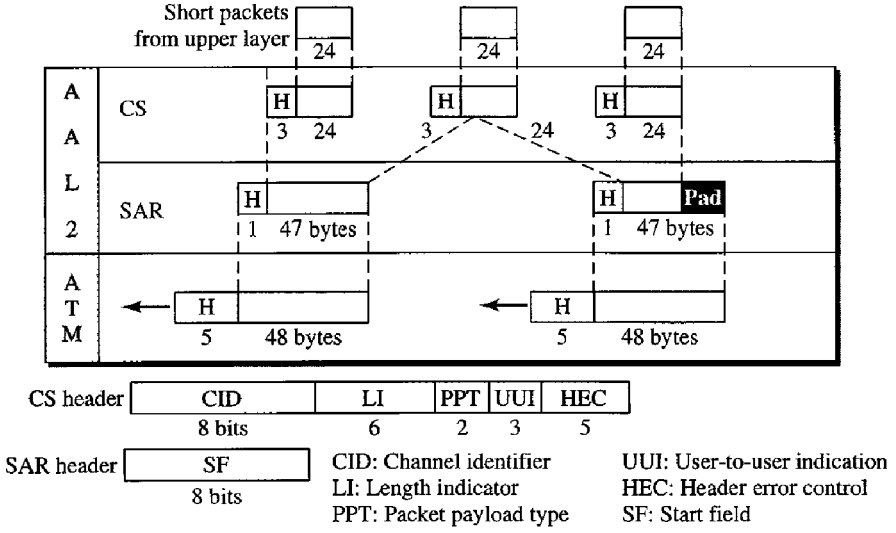


Figure 18.21 shows the process of encapsulating a short frame from the same source (the same user of a mobile phone) or from several sources (several users of mobile telephones) into one cell.

Figure 18.21 *AAL2*



The CS layer overhead consists of five fields:

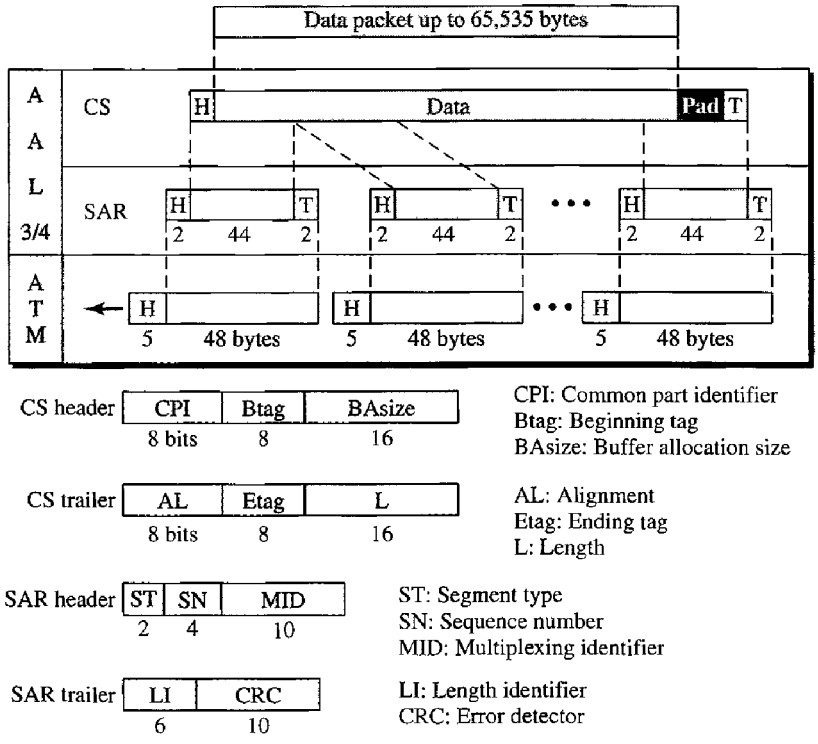
- **Channel identifier (CID).** The 8-bit CID field defines the channel (user) of the short packet.
- **Length indicator (LI).** The 6-bit LI field indicates how much of the final packet is data.
- **Packet payload type (PPT).** The PPT field defines the type of packet.

- **User-to-user indicator (UUI).** The UUI field can be used by end-to-end users.
- **Header error control (HEC).** The last 5 bits is used to correct errors in the header.

The only overhead at the SAR layer is the start field (SF) that defines the offset from the beginning of the packet.

AAL3/4 Initially, AAL3 was intended to support connection-oriented data services and AAL4 to support connectionless services. As they evolved, however, it became evident that the fundamental issues of the two protocols were the same. They have therefore been combined into a single format called **AAL3/4.** Figure 18.22 shows the AAL3/4 sublayer.

Figure 18.22 *AAL3/4*



The CS layer header and trailer consist of six fields:

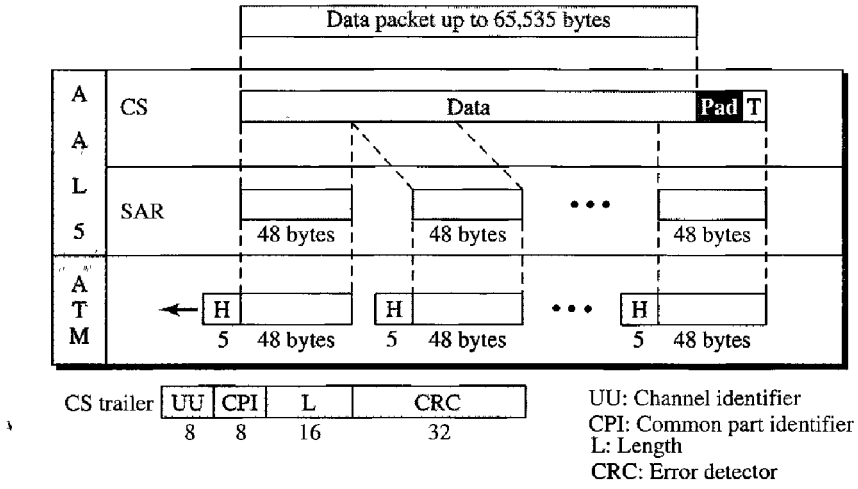
- **Common part identifier (CPI).** The CPI defines how the subsequent fields are to be interpreted. The value at present is 0.
- **Begin tag (Btag).** The value of this field is repeated in each cell to identify all the cells belonging to the same packet. The value is the same as the Etag (see below).
- **Buffer allocation size (BAsize).** The 2-byte BA field tells the receiver what size buffer is needed for the coming data.
- **Alignment (AL).** The 1-byte AL field is included to make the rest of the trailer 4 bytes long.
- **Ending tag (Etag).** The 1-byte ET field serves as an ending flag. Its value is the same as that of the beginning tag.
- **Length (L).** The 2-byte L field indicates the length of the data unit.

The SAR header and trailer consist of five fields:

- **Segment type (ST).** The 2-bit ST identifier specifies the position of the segment in the message: beginning (00), middle (01), or end (10). A single-segment message has an ST of 11.
- **Sequence number (SN).** This field is the same as defined previously.
- **Multiplexing identifier (MID).** The 10-bit MID field identifies cells coming from different data flows and multiplexed on the same virtual connection.
- **Length indicator (LI).** This field defines how much of the packet is data, not padding.
- **CRC.** The last 10 bits of the trailer is a CRC for the entire data unit.

AAL5 AAL3/4 provides comprehensive sequencing and error control mechanisms that are not necessary for every application. For these applications, the designers of ATM have provided a fifth AAL sublayer, called the **simple and efficient adaptation layer (SEAL). AAL5** assumes that all cells belonging to a single message travel sequentially and that control functions are included in the upper layers of the sending application. Figure 18.23 shows the AAL5 sublayer.

Figure 18.23 *AAL5*



The four trailer fields in the CS layer are

- **User-to-user (UU).** This field is used by end users, as described previously.
- **Common part identifier (CPI).** This field is the same as defined previously.
- **Length (L).** The 2-byte L field indicates the length of the original data.
- **CRC.** The last 4 bytes is for error control on the entire data unit.

Congestion Control and Quality of Service

ATM has a very developed congestion control and quality of service that we discuss in Chapter 24.

18.3 ATM LANs

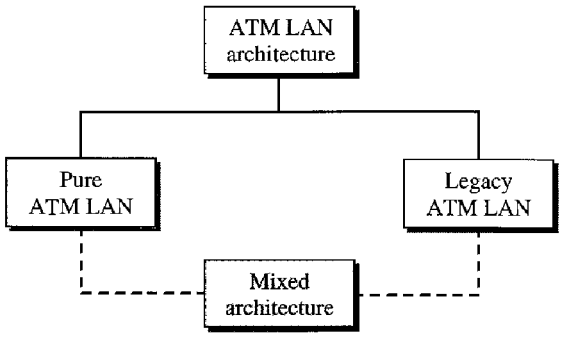
ATM is mainly a wide-area network (WAN ATM); however, the technology can be adapted to local-area networks (ATM LANs). The high data rate of the technology (155 and 622 Mbps) has attracted the attention of designers who are looking for greater and greater speeds in LANs. In addition, ATM technology has several advantages that make it an ideal LAN:

- ATM technology supports different types of connections between two end users. It supports permanent and temporary connections.
- ATM technology supports multimedia communication with a variety of bandwidths for different applications. It can guarantee a bandwidth of several megabits per second for real-time video. It can also provide support for text transfer during off-peak hours.
- An ATM LAN can be easily adapted for expansion in an organization.

ATM LAN Architecture

Today, we have two ways to incorporate ATM technology in a LAN architecture: creating a **pure ATM LAN** or making a **legacy ATM LAN.** Figure 18.24 shows the taxonomy.

Figure 18.24 *ATM LANs*



Pure ATM Architecture

In a pure ATM LAN, an **ATM switch** is used to connect the stations in a LAN, in exactly the same way stations are connected to an Ethernet switch. Figure 18.25 shows the situation.

In this way, stations can exchange data at one of two standard rates of ATM technology (155 and 652 Mbps). However, the station uses a virtual path identifier (VPI) and a virtual circuit identifier (VCI), instead of a source and destination address.

This approach has a major drawback. The system needs to be built from the ground up; existing LANs cannot be upgraded into pure ATM LANs.

Legacy LAN Architecture

A second approach is to use ATM technology as a backbone to connect traditional LANs. Figure 18.26 shows this architecture, a **legacy ATM LAN.**

Figure 18.25 *Pure ATM LAN*

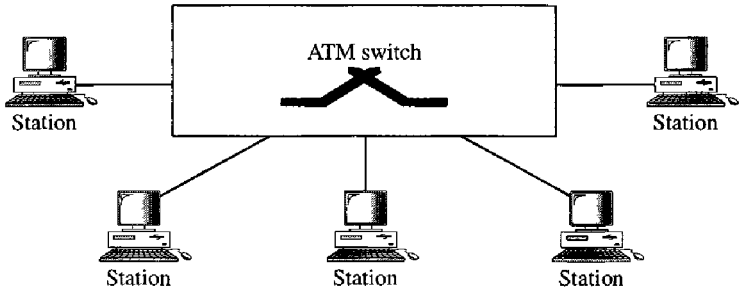
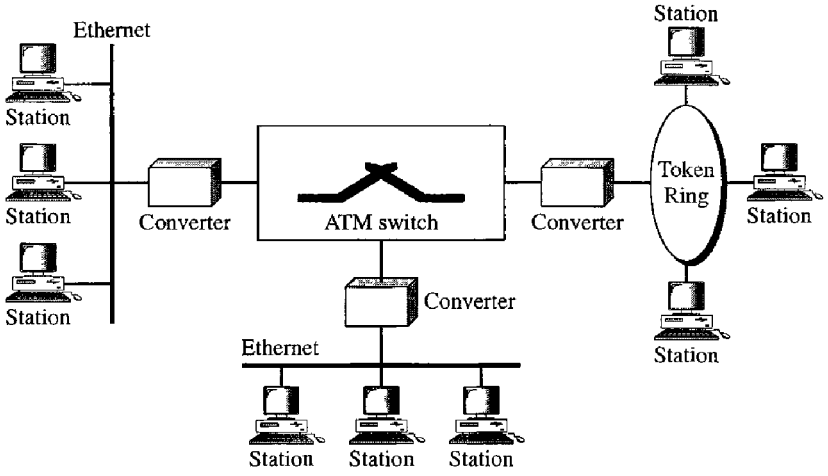


Figure 18.26 *Legacy ATM LAN*



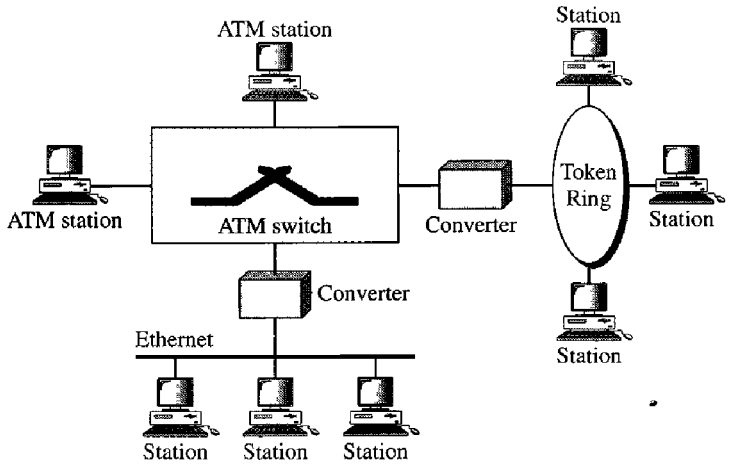
In this way, stations on the same LAN can exchange data at the rate and format of traditional LANs (Ethernet, Token Ring, etc.). But when two stations on two different LANs need to exchange data, they can go through a converting device that changes the frame format. The advantage here is that output from several LANs can be multiplexed together to create a high-data-rate input to the ATM switch. There are several issues that must be resolved first.

Mixed Architecture

Probably the best solution is to mix the two previous architectures. This means keeping the existing LANs and, at the same time, allowing new stations to be directly connected to an ATM switch. The **mixed architecture LAN** allows the gradual migration of legacy LANs onto ATM LANs by adding more and more directly connected stations to the switch. Figure 18.27 shows this architecture.

Again, the stations in one specific LAN can exchange data using the format and data rate of that particular LAN. The stations directly connected to the ATM switch can use an ATM frame to exchange data. However, the problem is, How can a station in a

Figure 18.27 *Mixed architecture ATM LAN*



traditional LAN communicate with a station directly connected to the ATM switch or vice versa? We see how the problem is resolved now.

LAN Emulation (LANE)

At the surface level, the use of ATM technology in LANs seems like a good idea. However, many issues need to be resolved, as summarized below:

- **Connectionless versus connection-oriented.** Traditional LANs, such as Ethernet, are *connectionless protocols.* A station sends data packets to another station whenever the packets are ready. There is no *connection establishment* or *connection termination* phase. On the other hand, ATM is a *connection-oriented protocol;* a station that wishes to send cells to another station must first establish a connection and, after all the cells are sent, terminate the connection.
- **Physical addresses versus virtual-circuit identifiers.** Closely related to the first issue is the difference in addressing. A connectionless protocol, such as Ethernet, defines the route of a packet through *source* and *destination addresses.* However, a connection-oriented protocol, such as ATM, defines the route of a cell through virtual connection identifiers (VPIs and VCIs).
- **Multicasting and broadcasting delivery.** Traditional LANs, such as Ethernet, can both *multicast* and *broadcast* packets; a station can send packets to a group of stations or to all stations. There is no easy way to multicast or broadcast on an ATM network although point-to-multipoint connections are available.
- **Interoperability.** In a mixed architecture, a station connected to a legacy LAN must be able to communicate with a station directly connected to an ATM switch.

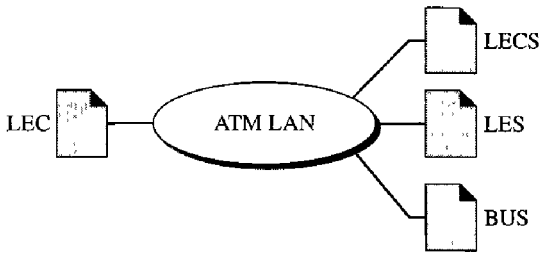
An approach called **local-area network emulation (LANE)** solves the above-mentioned problems and allows stations in a mixed architecture to communicate with one another. The approach uses emulation. Stations can use a connectionless service that emulates a connection-oriented service. Stations use the source and destination addresses for initial

connection and then use VPI and VCI addressing. The approach allows stations to use unicast, multicast, and broadcast addresses. Finally, the approach converts frames using a legacy format to ATM cells before they are sent through the switch.

Client/Server Model

LANE is designed as a **client/server model** to handle the four previously discussed problems. The protocol uses one type of client and three types of servers, as shown in Figure 18.28.

Figure 18.28 *Client and servers in a LANE*



LAN Emulation Client

All ATM stations have **LAN emulation client (LEC)** software installed on top of the three ATM protocols. The upper-layer protocols are unaware of the existence of the ATM technology. These protocols send their requests to LEC for a LAN service such as connectionless delivery using MAC unicast, multicast, or broadcast addresses. The LEC, however, just interprets the request and passes the result on to the servers.

LAN Emulation Configuration Server

The **LAN emulation configuration server (LECS)** is used for the initial connection between the client and LANE. This server is always waiting to receive the initial contact. It has a well-known ATM address that is known to every client in the system.

LAN Emulation Server

LAN emulation server (LES) software is installed on the LES. When a station receives a frame to be sent to another station using a physical address, LEC sends a special frame to the LES. The server creates a virtual circuit between the source and the destination station. The source station can now use this virtual circuit (and the corresponding identifier) to send the frame or frames to the destination.

Broadcast/Unknown Server

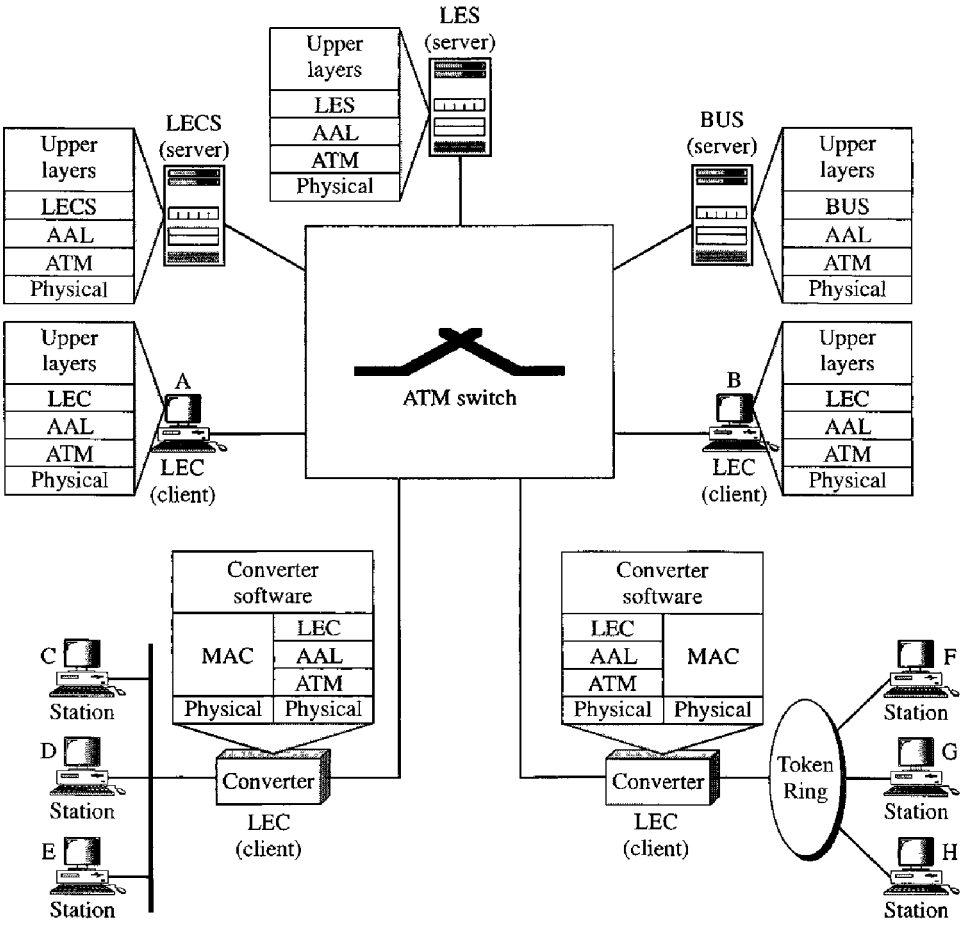
Multicasting and broadcasting require the use of another server called the **broadcast/unknown server (BUS).** If a station needs to send a frame to a group of stations or to every station, the frame first goes to the BUS; this server has permanent virtual connections to every station. The server creates copies of the received frame and sends a copy to a group of stations or to all stations, simulating a multicasting or broadcasting process.

The server can also deliver a unicast frame by sending the frame to every station. In this case the destination address is unknown. This is sometimes more efficient than getting the connection identifier from the LES.

Mixed Architecture with Client/Server

Figure 18.29 shows clients and servers in a mixed architecture ATM LAN. In the figure, three types of servers are connected to the ATM switch (they can actually be part of the switch). Also we show two types of clients. Stations A and B, designed to send and receive LANE communication, are directly connected to the ATM switch. Stations C, D, E, F, G, and H in traditional legacy LANs are connected to the switch via a converter. These converters act as LEC clients and communicate on behalf of their connected stations.

Figure 18.29 *A mixed architecture ATM LAN using LANE*



18.4 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

Frame Relay and ATM are discussed in Chapter 3 of [Sta98]. ATM LAN is discussed in Chapter 14 of [For03]. Chapter 7 of [Kei02] also has a good discussion of ATM LANs.

18.5 KEY TERMS

AAL1
AAL2
AAL3/4
AAL5
application adaptation layer (AAL)
Asynchronous Transfer Mode (ATM)
ATM layer
ATM switch
backward explicit congestion notification (BECN)
bandwidth on demand
broadcast/unknown server (BUS)
bursty data
cell
cell network
cell relay
client/server model
congestion control
convergence sublayer (CS)
data link connection identifier (DLCI)
discard eligibility (DE)
forward explicit congestion notification (FECN)
Frame Relay
Frame Relay assembler/disassembler (FRAD)
LAN emulation client (LEC)
LAN emulation configuration server (LECS)
LAN emulation server (LES)
legacy ATM LAN
local-area network emulation (LANE)
Local Management Information (LMI)
mixed architecture LAN
network-to-network interface (NNI)
permanent virtual circuit (PVC)
pure ATM LAN
quality of service (QoS)
segmentation and reassembly (SAR)
simple and efficient adaptation layer (SEAL)
switched virtual circuit (SVC)
transmission path (TP)
user-to-network interface (UNI)
virtual circuit (VC)
virtual-circuit identifier (VCI)
virtual path (VP)
virtual path identifier (VPI)
Voice Over Frame Relay (VOFR)
X.25

18.6 SUMMARY

- Virtual-circuit switching is a data link layer technology in which links are shared.
- A virtual-circuit identifier (VCI) identifies a frame between two switches.
- Frame Relay is a relatively high-speed, cost-effective technology that can handle bursty data.

- Both PVC and SVC connections are used in Frame Relay.
- The data link connection identifier (DLCI) identifies a virtual circuit in Frame Relay.
- Asynchronous Transfer Mode (ATM) is a cell relay protocol that, in combination with SONET, allows high-speed connections.
- A cell is a small, fixed-size block of information.
- The ATM data packet is a cell composed of 53 bytes (5 bytes of header and 48 bytes of payload).
- ATM eliminates the varying delay times associated with different-size packets.
- ATM can handle real-time transmission.
- A user-to-network interface (UNI) is the interface between a user and an ATM switch.
- A network-to-network interface (NNI) is the interface between two ATM switches.
- In ATM, connection between two endpoints is accomplished through transmission paths (TPs), virtual paths (VPs), and virtual circuits (VCs).
- In ATM, a combination of a virtual path identifier (VPI) and a virtual-circuit identifier identifies a virtual connection.
- The ATM standard defines three layers:
  a. Application adaptation layer (AAL) accepts transmissions from upper-layer services and maps them into ATM cells.
  b. ATM layer provides routing, traffic management, switching, and multiplexing services.
  c. Physical layer defines the transmission medium, bit transmission, encoding, and electrical-to-optical transformation.
- The AAL is divided into two sublayers: convergence sublayer (CS) and segmentation and reassembly (SAR).
- There are four different AALs, each for a specific data type:
  a. AAL1 for constant-bit-rate stream.
  b. AAL2 for short packets.
  c. AAL3/4 for conventional packet switching (virtual-circuit approach or datagram approach).
  d. AAL5 for packets requiring no sequencing and no error control mechanism.
- ATM technology can be adopted for use in a LAN (ATM LAN).
- In a pure ATM LAN, an ATM switch connects stations.
- In a legacy ATM LAN, the backbone that connects traditional LANs uses ATM technology.
- A mixed architecture ATM LAN combines features of a pure ATM LAN and a legacy ATM LAN.
- Local-area network emulation (LANE) is a client/server model that allows the use of ATM technology in LANs.
- LANE software includes LAN emulation client (LECS), LAN emulation configuration server (LECS), LAN emulation server (LES), and broadcast/unknown server (BUS) modules.

18.7 PRACTICE SET

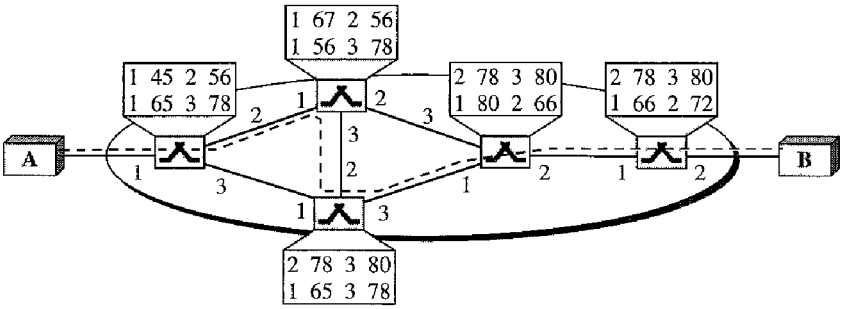
Review Questions

1. There are no sequence numbers in Frame Relay. Why?
2. Can two devices connected to the same Frame Relay network use the same DLCIs?
3. Why is Frame Relay a better solution for connecting LANs than T-1 lines?
4. Compare an SVC with a PVC.
5. Discuss the Frame Relay physical layer.
6. Why is multiplexing more efficient if all the data units are the same size?
7. How does an NNI differ from a UNI?
8. What is the relationship between TPs, VPs, and VCs?
9. How is an ATM virtual connection identified?
10. Name the ATM layers and their functions.
11. How many virtual connections can be defined in a UNI? How many virtual connections can be defined in an NNI?
12. Briefly describe the issues involved in using ATM technology in LANs.

Exercises

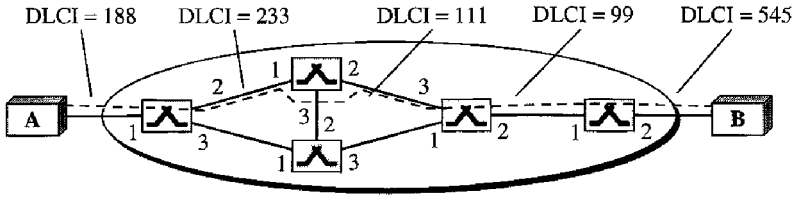
13. The address field of a Frame Relay frame is 1011000000010111. What is the DLCI (in decimal)?
14. The address field of a Frame Relay frame is 101100000101001. Is this valid?
15. Find the DLCI value if the first 3 bytes received is 7C 74 E1 in hexadecimal.
16. Find the value of the 2-byte address field in hexadecimal if the DLCI is 178. Assume no congestion.
17.* In Figure 18.30 a virtual connection is established between A and B. Show the DLCI for each link.

Figure 18.30 *Exercise 17*



18. In Figure 18.31 a virtual connection is established between A and B. Show the corresponding entries in the tables of each switch.

Figure 18.31 *Exercise 18*



19. An AAL1 layer receives data at 2 Mbps. How many cells are created per second by the ATM layer?
20. What is the total efficiency of ATM using AAL1 (the ratio of received bits to sent bits)?
21. If an application uses AAL3/4 and there are 47,787 bytes of data coming into the CS, how many padding bytes are necessary? How many data units get passed from the SAR to the ATM layer? How many cells are produced?
22. Assuming no padding, does the efficiency of ATM using AAL3/4 depend on the size of the packet? Explain your answer.
23. What is the minimum number of cells resulting from an input packet in the AAL3/4 layer? What is the maximum number of cells resulting from an input packet?
24. What is the minimum number of cells resulting from an input packet in the AAL5 layer? What is the maximum number of cells resulting from an input packet?
25. Explain why padding is unnecessary in AAL1, but necessary in other AALs.
26. Using AAL3/4, show the situation where we need _____ of padding.
    a. 0 bytes (no padding)
    b. 40 bytes
    c. 43 bytes
27. Using AAL5, show the situation where we need _____ of padding.
    a. 0 bytes (no padding)
    b. 40 bytes
    c. 47 bytes
28. In a 53-byte cell, how many bytes belong to the user in the following (assume no padding)?
    a. AAL1
    b. AAL2
    c. AAL3/4 (not the first or last cell)
    d. AAL5 (not the first or last cell)

Research Activities

29. Find out about I.150 protocol that provides generic flow control for UNI interface.
30. ATM uses the 8-bit HEC (header error control) field to control errors in the first four bytes (32 bits) of the header. The generating polynomial is $x^8 + x^2 + x + 1$. Find out how this is done.
31. Find the format of the LANE frames and compare it with the format of the Ethernet frame.
32. Find out about different steps involved in the operation of a LANE.